10-17-2003

Energy and Transient Power Minimization During Behavioral Synthesis

Saraju P. Mohanty
University of South Florida

Energy and Transient Power Minimization During Behavioral Synthesis

by

Saraju P. Mohanty

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: N. Ranganathan, Ph.D.
Murali Varanasi, Ph.D.
Srinivas Katkoori, Ph.D.
Wilfredo A. Moreno, Ph.D.
A. N. V. Rao, Ph.D.

Date of Approval:
October 17, 2003

Keywords: peak power, average power, power fluctuation, low power synthesis, datapath scheduling, multiple supply voltages, dynamic frequency clocking, multicycling, digital watermarking

DEDICATION

My state Kalinga (Orissa), World's largest democracy (India), World's oldest democracy (USA), my Parents, my Sisters, Uma, and to every one who has taught me free thinking.

ACKNOWLEDGEMENTS

I would like to express gratitude to my major professor, Dr. N. Ranganathan, for his guidance and support throughout my doctoral degree program. I would sincerely like to thank Dr. K. R. Ramakrishan, Dr. Mohan S. Kanakanhalli, Dr. Chitta Baral, Dr. Rabi N. Mahapatra, Dr. Debasmita Misra, Dr. Srinivas Katkoori and Dr. Sanjukta Bhanja for there support in various phases of my student life. Special thanks to Dr. D. Rundus, Dr. R. Perez, Dr. Goldgof and all the members of my Ph.D. committee. I would also like to thank all members of VCAPP group (such as, Ashok, Sunil, Ravi, Karthik, Suvodeep, Mouli, Bamini, Stelian, Hao, Praveen, etc.) for their help and cooperation. Special thanks to Dr. Austell, ISSS office at USF, the office staffs of CSE department at USF and technical support staff of CSE department at USF (Daniel). Last but not the least, I thank all my friends (Uma, Rupesh, Siddy, Ajaya, Lulu, Pati, Prince, Bhabani, Durga, Amaresh, Krishna, Rajib, Sridhar, Saroj, Jai, Hari, etc.), who have always been a constant source of moral support.

TABLE OF CONTENTS

LIST OF TABLES v

LIST OF FIGURES viii

ABSTRACT xiii

CHAPTER 1 INTRODUCTION 1
- 1.1 Fundamentals of High Level Synthesis 4
  - 1.1.1 Why High-Level Synthesis ? 7
  - 1.1.2 Various Phases of High-Level Synthesis 8
  - 1.1.3 A Synthesis Example 12
- 1.2 Sources of Power Dissipation in a CMOS Circuit 12
- 1.3 Methods for Power Reduction in High-Level Synthesis 16
- 1.4 Why Peak Power Minimization ? 18
- 1.5 Why Average Power and Energy Reduction ? 19
- 1.6 Why Transient Power Minimization ? 20
- 1.7 Why Frequency and Voltage Scaling ? 20
- 1.8 Multiple Supply Voltages, Dynamic Clocking and Multicycling Preliminaries 21
  - 1.8.1 What is Dynamic Frequency Clocking ? 22
  - 1.8.2 Energy or Power Reduction Due to Voltage or Frequency Scaling 22
  - 1.8.3 Issues in Multiple Supply Voltage Based Design 25
  - 1.8.4 Level Converter Design 26
  - 1.8.5 Dynamic Frequency Clocking Unit Design 27
- 1.9 Fundamentals of Digital Watermarking 31
  - 1.9.1 General Framework for Watermarking 32
  - 1.9.2 Types of Watermarking 35
- 1.10 Contributions of this Dissertation 38
- 1.11 Dissertation Outline 40

CHAPTER 2 RELATED WORK 41
- 2.1 Datapath Scheduling for Energy or Average Power Reduction using Voltage Reduction 42
- 2.2 Switching Activity Reduction During High-Level Synthesis 47
- 2.3 Datapath Scheduling for Peak Power Reduction 55
- 2.4 Scheduling for Variable Voltage Processor 57
- 2.5 Design and Synthesis for Low-Power or High-Performance Variable Voltage / Frequency / Latency and Multiple Voltage Based Systems 65

2.6 Hardware Based Digital Watermarking Systems 72
2.7 This Dissertation 73

CHAPTER 3 ENERGY MINIMIZATION 75
3.1 Target Architecture and Datapath Specifications 75
3.2 Time Constrained Scheduling 77
3.2.1 Algorithm Flow 78
3.2.2 Pseudocode Description 80
3.2.3 Time Complexity 82
3.3 Resource Constrained Scheduling 84
3.3.1 Algorithm Flow 86
3.3.2 Pseudocode of the Resource Constrained Algorithm 87
3.3.3 Time Complexity 90
3.4 Experimental Results 91
3.5 Conclusions 96

CHAPTER 4 ENERGY DELAY PRODUCT MINIMIZATION 98
4.1 Energy Delay Product of a Datapath Circuit 98
4.2 ILP Formulations 102
4.2.1 ILP Formulations : Dynamic Frequency Clocking 102
4.2.2 ILP Formulations : Multicycling 103
4.3 Datapath Scheduling Algorithm 105
4.3.1 Scheduling for MVDFC 105
4.3.2 Scheduling for MVMC 106
4.4 Experimental Results 110
4.5 Conclusions 113

CHAPTER 5 PEAK POWER AND AVERAGE POWER MINIMIZATION 114
5.1 Peak and Average Power Consumption of a Datapath Circuit 114
5.2 ILP Formulations 117
5.2.1 ILP Formulations for DFC 117
5.2.2 ILP Formulations for Multicycling 119
5.3 ILP-Based Scheduler 120
5.3.1 Scheduler using Multiple Voltages and Dynamic Frequency Clocking 121
5.3.2 Scheduler using Multiple Supply Voltages and Multicycling 124
5.4 Experimental Results 126
5.5 Peak Power Minimization 128
5.5.1 ILP Formulations 128
5.5.1.1 Multiple Supply Voltages and Dynamic Frequency Clocking (MVDFC) 130
5.5.1.2 Multiple Supply Voltages and Multicycling (MVMC) 131
5.5.2 ILP-Based Scheduler 132
5.5.2.1 Scheduling for MVDFC 132
5.5.2.2 Scheduling for MVMC 133
5.5.3 Experimental Results 139

5.6 Conclusions 142

CHAPTER 6 ENERGY AND TRANSIENT POWER MINIMIZATION 143
6.1 Cycle Power Function (CPF) 144
6.1.1 Model 1 : CPF using Mean Deviation 145
6.1.2 Model 2 : CPF using Cycle-to-Cycle Gradient 148
6.2 CPF-Scheduler Algorithm 150
6.3 Experimental Results 157
6.4 Conclusions 164

CHAPTER 7 TRANSIENT POWER MINIMIZATION 166
7.1 Modified Cycle Power Function 167
7.2 Modeling of Non-linearities 170
7.2.1 LP Formulation Involving Sum of Absolute Deviations 170
7.2.2 LP Formulation Involving Fraction 171
7.3 ILP Formulations to Minimize Cycle Power Function 172
7.3.1 Multiple Voltages and Dynamic Frequency Clocking (MVDFC) 173
7.3.2 Multiple Voltages and Multicycling (MVMC) 176
7.4 ILP-Based Scheduling Algorithm 179
7.4.1 CPF-MVDFC Scheduling Scheme 181
7.4.2 CPF-MVMC Scheduling Scheme 182
7.5 Experimental Results 183
7.6 Conclusions 189

CHAPTER 8 POWER FLUCTUATION MINIMIZATION 193
8.1 Power Fluctuation Modeling 194
8.2 Modeling of Non-linearities 197
8.3 ILP Formulations to Minimize Mean Power Gradient 199
8.3.1 Formulations using Multiple Voltages and Dynamic Frequency 199
8.3.2 Formulations using Multiple Supply Voltages and Multicycling 201
8.4 Scheduling Algorithm 204
8.5 Experimental Results 207
8.6 Conclusions 213

CHAPTER 9 VLSI DESIGN FOR DIGITAL WATERMARKING OF IMAGES 214
9.1 Invisible Watermarking in Spatial Domain 214
9.1.1 Spatial Domain Invisible Watermarking Algorithms 216
9.1.1.1 Invisible Robust Algorithm 216
9.1.1.2 Invisible Fragile Algorithm 218
9.1.2 VLSI Architecture for Invisible Spatial Domain Watermarking 220
9.1.2.1 Architecture for Robust Watermarking 220
9.1.2.2 Architecture for Fragile Watermarking 222
9.1.2.3 Overall Chip Architecture 222
9.1.3 Implementation of Spatial Domain Invisible Watermarking Chip 223
9.1.4 Results and Conclusions 227

9.2 Visible Watermarking in Spatial Domain 229
9.2.1 Watermarking Algorithms 229
9.2.1.1 Visible Watermarking Algorithm 1 : 229
9.2.1.2 Visible Watermarking Algorithm 2 : 231
9.2.2 VLSI Architecture 234
9.2.2.1 Architecture for Algorithm 1 : 234
9.2.2.2 Architecture for Algorithm 2 : 236
9.2.2.3 Architecture for the Watermarking Processor : 238
9.2.3 Chip Implementation 239
9.2.4 Results and Conclusions 243
9.3 Invisible and Visible Watermarking in DCT Domain 245
9.3.1 Watermarking Algorithms 246
9.3.1.1 Spread Spectrum Invisible Watermarking Insertion Algorithm 246
9.3.1.2 Visible Watermarking Insertion Algorithm 248
9.3.1.3 Algorithm Modification for Hardware Implementations 249
9.3.2 VLSI Architecture 250

CHAPTER 10 CONCLUSIONS AND FUTURE WORK 256

REFERENCES 258

ABOUT THE AUTHOR End Page

LIST OF TABLES

Table 2.1	Datapath Scheduling Schemes using Multiple Supply Voltages	45
Table 2.2	High-Level Synthesis Schemes using Switching Activity Reduction	51
Table 2.3	Relative Performance of Various Schemes Proposed for Peak Power Minimization	55
Table 2.4	Scheduling Algorithms for Variable Voltage Processor	60
Table 2.5	Design and Synthesis Works on Variable Frequency or Multiple Frequency	67
Table 2.6	Watermarking Chips Proposed in Current Literature	73
Table 3.1	List of Functions used in the TC-DFC Algorithm	79
Table 3.2	List of Variables and Data Structures used in the TC-DFC Algorithm Description	80
Table 3.3	TC-DFC Freqeuncy Selection : from left $\rightarrow$ right	80
Table 3.4	Vertex Priority List	80
Table 3.5	Cycle Priority List : $T_c \approx 2 * T_{cp} or 1.75 * T_{cp}$	82
Table 3.6	Cycle Priority List : $T_c \approx 1.5 * T_{cp}$	82
Table 3.7	Frequency Selection (From Left to Right in Each Step)	85
Table 3.8	Resource Look-up Table (order, From Left to Right)	85
Table 3.9	List of Functions used in the RC-DFC Algorithm	87
Table 3.10	List of Variables and Data Structures used in the RC-DFC Algorithm Description	89
Table 3.11	Resource Constraints used in our Experiements	93
Table 3.12	Energy Details for Different Benchmarks (for $\alpha = 0.5$) using RC-DFC Scheduler	94
Table 3.13	Configurations for Minimum EDP using RC-DFC	95

Table 3.14	Energy Savings using TC-DFC Scheduler	95
Table 3.15	Savings for Various Resource Constrained Schedulings	97
Table 3.16	Savings for Various Time Constrained Schedulings	97
Table 4.1	Notations used in Description	100
Table 4.2	Notations used in ILP Formulations	102
Table 4.3	Energy and EDP Estimates for Benchmarks for MVDFC and MVMC Schemes	111
Table 4.4	Savings for Various Schedulings Schemes	113
Table 5.1	Notations used in Description	115
Table 5.2	Notations used in ILP Formulations	117
Table 5.3	Notations used in Expressing Results	127
Table 5.4	Resource Constraints used for our Experiement	128
Table 5.5	Peak Power, Average Power and PDP Estimates for Benchmarks using Scheduling Schemes	129
Table 5.6	Peak and Average Power Reduction for Various Scheduling Schemes	131
Table 5.7	Resource Constraints used for our Experiment	139
Table 5.8	Power Estimates for MVDFC and MVMC Scheduling Schemes	140
Table 5.9	Power Reduction for Various Scheduling Schemes	141
Table 6.1	List of Notataions and Terminology used in CPF Modeling	144
Table 6.2	Notations used to Express the Results	158
Table 6.3	Power Estimates for Different Benchmarks (using Model 1)	159
Table 6.4	Power Estimates for Different Benchmarks (using Model 2)	163
Table 7.1	List of Variables used in ILP Formulations	173
Table 7.2	List of Variables used to Express the Results	184
Table 7.3	Power, Energy and EDP Estimates for Benchmarks using MVDFC	186
Table 7.4	Power, energy and EDP Estimates for Benchmarks using MVMC	187
Table 8.1	Notations used in the Description	195

Table 8.2	Notations used in ILP formulations	199
Table 8.3	Notations used in Describing the Results	208
Table 8.4	Power Estimates for Benchmarks	209
Table 9.1	Notations used to Explain Spatial Domain Watermarking Algorithms	216
Table 9.2	Control Signals for Spatial Domain Invisible Watermarking Chip	224
Table 9.3	Power, Area Details for Individual Units	225
Table 9.4	Overall Chip Statistics	226
Table 9.5	List of Variables used in Algorithm Explanation	230
Table 9.6	Power and Area of Different Units	242
Table 9.7	Overall Statistics of the Watermarking Chip	243
Table 9.8	Notations used in the Description of the Algorithm	247
Table 9.9	Overall Statistics of the DCT Domain Watermarking Chip [85]	255

LIST OF FIGURES

Figure 1.1	Chronological Change in Power, Power Density, Transistor Count, Gate Count, Operating Frequency and Feature Size of CMOS Integrated Circuits	2
Figure 1.2	Desription of Hardware in Different Domains and Abstractions [4]	5
Figure 1.3	Synthesis Flow	6
Figure 1.4	Various Phases of High-Level Synthesis	8
Figure 1.5	Data Flow Graph and Control Flow Graph of a Square Root Algorithm [3]	10
Figure 1.6	Different Types of Scheduling Algorithms	11
Figure 1.7	A Synthesis Example : Step 1 to Step 3	13
Figure 1.8	The Synthesis Example : Step 4 to Step 6	14
Figure 1.9	Sources of Power Dissipation in a CMOS Circuit	15
Figure 1.10	Static Vs Dynamic Power Dissipation for Different Switching Activity [6, 7]	17
Figure 1.11	Dynamic Frequency Generation using Dynamic Clocking Unit [54]	23
Figure 1.12	Data Flow Graph in Three Modes of Operation	24
Figure 1.13	Level Converter Schematic Diagram [65, 66]	27
Figure 1.14	Level Converter Layout and Simulation	28
Figure 1.15	Dynamic Clocking Unit : Ranganathan, et. al. [59]	29
Figure 1.16	Dynamic Clocking Unit and Output Clock : Byrnjolfson and Zilic [61]	30
Figure 1.17	Visible Watermarked Image [71]	32
Figure 1.18	General Framework of Digital Watermarking	34
Figure 1.19	Different Types of Watermarks and Watermarking Techniques	36

Figure 1.20	Contributions of this Dissertation	38
Figure 1.21	Energy Vs Peak Power Efficient Schedule	39
Figure 2.1	Variable Voltage Processor Operation : Voltage Vs Frequency [122]	58
Figure 3.1	Level Converters Needed for Stepping up Signal	76
Figure 3.2	HAL Differential Equation Solver (with ASAP labels)	77
Figure 3.3	TC-DFC Scheduling Algorithm Flow	78
Figure 3.4	Pseudo-code for TC-DFC Scheduling Algorithm	81
Figure 3.5	Schedules Obtained for HAL Benchmark for Different Time Constraints using TC-DFC	83
Figure 3.6	RC-DFC Scheduling Algorithm Flow	86
Figure 3.7	Pseudo-code for RC-DFC Scheduler	88
Figure 3.8	Final Schedule of FIR Filter DFG (using RC-DFC)	91
Figure 3.9	Average Energy and EDP Reduction for Benchmarks	96
Figure 4.1	ILP Based Scheduling for Low EDP	105
Figure 4.2	Example Data Flow Graph for Multiple Supply Voltages and Dynamic Frequency Clocking	106
Figure 4.3	ILP Formulation for Example DFG for Multiple Supply Voltages and Dynamic Frequency Clocking	107
Figure 4.4	Example DFG (for RC2) (MVMC)	108
Figure 4.5	ILP Formulation for Example DFG for Multiple Supply Voltages and Multicycling	109
Figure 4.6	Reduction for Different Benchmarks Expressed as Percentage in Average	112
Figure 5.1	ILP-Based Scheduler	121
Figure 5.2	Example DFG for Resource Constraint RC3; using Multiple Supply Voltages and Dynamic Frequency Clocking	122
Figure 5.3	ILP Formulation for Example DFG using DFC, for RC3 and Switching Activity = 0.5	123
Figure 5.4	Example DFG for Resource Constraint RC3; using Multiple Supply Voltages and Multicycling	124

Figure 5.5	ILP Formulation for Example DFG using Multicycling, for RC3 and Switching Activity = 0.5	125
Figure 5.6	Average Reduction for Different Bechmarks	130
Figure 5.7	Example DFG (for RC1) (MVDFC)	133
Figure 5.8	ILP Formulation for Example DFG (MVDFC)	134
Figure 5.9	ILP Formulation for Example DFG (MVDFC) in AMPL	135
Figure 5.10	Example DFG (for RC1) (MVMC)	136
Figure 5.11	ILP Formulation for Example DFG (MVMC)	137
Figure 5.12	ILP Formulation for Example DFG (MVMC) in AMPL	138
Figure 5.13	Average Reductions for Benchmarks	141
Figure 6.1	The CPF-Scheduler Algorithm Flow	152
Figure 6.2	The CPF-Scheduler Algorithm Heuristic	153
Figure 6.3	Cycle Power Consumptions for Resource Constraint RC1	161
Figure 6.4	Cycle Power Consumptions for Resource Constraint RC2	161
Figure 6.5	Cycle Power Consumptions for Resource Constraint RC3	162
Figure 6.6	Cycle Power Consumptions for Resource Constraint RC4	162
Figure 6.7	Percentage Average Reduction for Benchmarks using Model1	164
Figure 6.8	Percentage Average Reduction for Benchmarks using Model2	165
Figure 7.1	Scheduling for CPF^* Minimization	180
Figure 7.2	ASAP and ALAP Schedule for Example DFG (used to find Mobility Graph)	181
Figure 7.3	Mobility Graph and Final Schedule for Example DFG for RC5 using MVDFC	182
Figure 7.4	Mobility Graph and Final Schedule for Example DFG for RC5 using MVMC	183
Figure 7.5	Average Reductions in Power or Energy for Benchmarks using CPF-MVDFC	188
Figure 7.6	Average Reductions for Benchmarks using CPF-MVMC	189

Figure 7.7	Power Profile for Benchmark for Resource Constraint RC1	190
Figure 7.8	Power Profile for Benchmark for Resource Constraint RC2	191
Figure 7.9	Power Profile for Benchmark for Resource Constraint RC3	191
Figure 7.10	Power Profile for Benchmark for Resource Constraint RC4	192
Figure 7.11	Power Profile for Benchmark for Resource Constraint RC5	192
Figure 8.1	Scheduling for MPG Minimization	205
Figure 8.2	Example Data Flow Graph (DFG)	206
Figure 8.3	Average Reductions using DFC Scheme	210
Figure 8.4	Average Reductions using Multicycling Scheme	211
Figure 8.5	Power Profiles for Benchmarks (for RC2)	212
Figure 8.6	Power Profiles for Benchmarks (for RC3)	212
Figure 8.7	Power Profiles for Benchmarks (for RC5)	213
Figure 9.1	Secure JPEG Encoder : Block Level View [176]	215
Figure 9.2	Secure Digital Still Camera : Schematic View	215
Figure 9.3	Invisible Robust Watermarking in Spatial Domain [177, 178]	217
Figure 9.4	Invisible Fragile Watermarking in Spatial Domain [83, 72]	219
Figure 9.5	Datapath for Robust Watermarking	220
Figure 9.6	Datapath for Fragile Watermarking	221
Figure 9.7	Datapath For Combined Spatial Domain Invisible Robust / Fragile Watermarking	222
Figure 9.8	Controller For Combined Spatial Domain Invisible Robust / Fragile Watermarking	223
Figure 9.9	Layout of the Invisible Spatial Domain Watermarking Datapath and Controller	225
Figure 9.10	Layout of RAM (Zoomed view of a portion is shown)	226
Figure 9.11	Layout of the Proposed Spatial Domain Invisible Watermarking Chip	227
Figure 9.12	Pin Diagram for the Proposed Spatial Domain Invisible Watermarking Chip	227

Figure 9.13	Spatial Domain Invisible Watermarked Shuttle	228
Figure 9.14	Spatial Domain Invisible Watermarked Bird	228
Figure 9.15	Datapath Architectures for the Visible Watermarking Algorithms	235
Figure 9.16	Individual Datapath Units for Algorithm 2	237
Figure 9.17	Architecture for the Proposed Watermarking Processor	239
Figure 9.18	Layout of Datapath and Controller of the Proposed Chip	241
Figure 9.19	Layout and Floor Plan of the Proposed Watermarking Chip	242
Figure 9.20	Pin Diagram for the Proposed Watermarking Chip	243
Figure 9.21	Original Host Images (a, b, and c) and Watermark Image (d)	244
Figure 9.22	Watermarked Images for the First Algorithm	245
Figure 9.23	Watermarked Images for the Second Algorithm	245
Figure 9.24	Combined Architecture for DCT domain Invisible and Visible Watermarking Chip	251
Figure 9.25	Architecture of the Different Units used for Invisible Watermarking	252
Figure 9.26	Architecture of the Different Units used for Visible Watermarking	253
Figure 9.27	Dual Voltage and Dual Frequency Operation of the Datapath	254
Figure 9.28	Layout of the DCT Domain Invisible and Visible Watermarking Chip [85]	255
Figure 9.29	Floorplan of the DCT Domain Invisible and Visible Watermarking Chip [85]	255

ENERGY AND TRANSIENT POWER MINIMIZATION DURING BEHAVIORAL SYNTHESIS

Saraju P. Mohanty

ABSTRACT

The proliferation of portable systems and mobile computing platforms has increased the need for the design of low power consuming integrated circuits. The increase in chip density and clock frequencies due to technology advances has made low power design a critical issue. Low power design is further driven by several other factors such as thermal considerations and environmental concerns. In low-power design for battery driven portable applications, the reduction of peak power, peak power differential, average power and energy are equally important. In this dissertation, we propose a framework for the reduction of these parameters through datapath scheduling at behavioral level. Several ILP based and heuristic based scheduling schemes are developed for datapath synthesis assuming : (i) single supply voltage and single frequency (SVSF), (ii) multiple supply voltages and dynamic frequency clocking (MVDFC), and (iii) multiple supply voltages and multicycling (MVMC). The scheduling schemes attempt to minimize : (i) energy, (ii) energy delay product, (iii) peak power, (iv) simultaneous peak power and average power, (v) simultaneous peak power, average power, peak power differential and energy, and (vi) power fluctuation.

A new parameter called "Cycle Power Function" (CPF) is defined which captures the transient power characteristics as the equally weighted sum of normalized mean cycle power and normalized mean cycle differential power. Minimizing this parameter using multiple supply voltages and dynamic frequency clocking results in the reduction of both energy and transient power. The cycle differential power can be modeled as either the absolute deviation from the average power or as the cycle-to-cycle power gradient. The switching activity information is obtained from behavioral simulations. Power fluctuation is modeled as the cycle-to-cycle power gradient and to reduce fluc-

tuation the mean power gradient (MPG) is minimized. The power models take into consideration the effect of switching activity on the power consumption of the functional units.

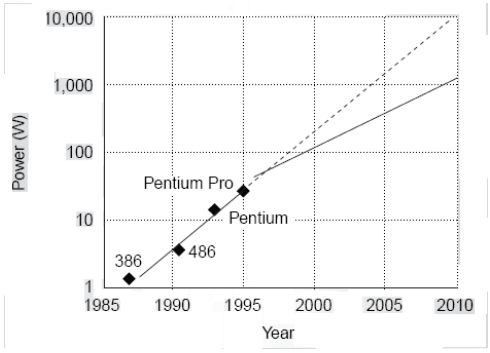
Experimental results for selected high-level synthesis benchmark circuits under different constraints indicate that significant reductions in power, energy and energy delay product can be obtained and that the MVDFC and MVMC schemes yield better power reduction compared to the SVSF scheme. Several application specific VLSI circuits were designed and implemented for digital watermarking of images. Digital watermarking is the process that embeds data called a watermark into a multimedia object such that the watermark can be detected or extracted later to make an assertion about the object. A class of VLSI architectures were proposed for various watermarking algorithms : (i) spatial domain invisible-robust watermarking scheme, (ii) spatial domain invisible-fragile watermarking scheme, (iii) spatial domain visible watermarking scheme, (iv) DCT domain invisible-robust watermarking scheme, and (v) DCT domain visible watermarking scheme. Prototype implementation of (i), (ii) and (iii) are given. The hardware modules can be incorporated in a ”JPEG encoder” or in a ”digital still camera”.

CHAPTER 1

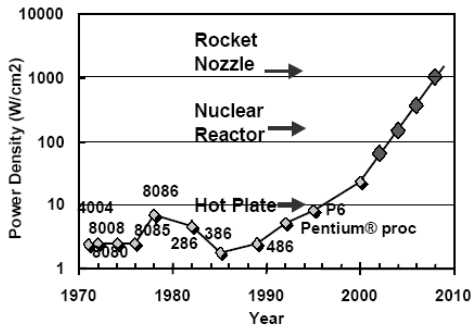
INTRODUCTION

Low power circuit design is a three dimensional problem involving area, performance and power trade-offs. Because of the decreasing feature size and increasing packing density, it may be possible to trade area against power [1]. The trend of decreasing device size and increasing chip densities involving several hundred millions of transistors per chip has resulted in tremendous increase in design complexity. Designing chips of such complexity using traditional *capture and simulate* methodology is time consuming and difficult. The industry has started looking at the development cycle to reduce design time and to gain a competitive edge. High-level synthesis of digital circuits has become necessary due to several advantages such as, reduction of design time, exploration of different design styles, meeting design constraints and requirements [2, 3, 4]. Additionally, this trend of reducing the feature size with increasing the clock frequency has made reliability a big challenge for the designers, mainly because of high on-chip electric fields [1, 5, 6, 7, 8]. Fig. 1.1 shows the chronologcal change in power, power density, transistor count, gate count, operating frequency and feature size of CMOS ICs.

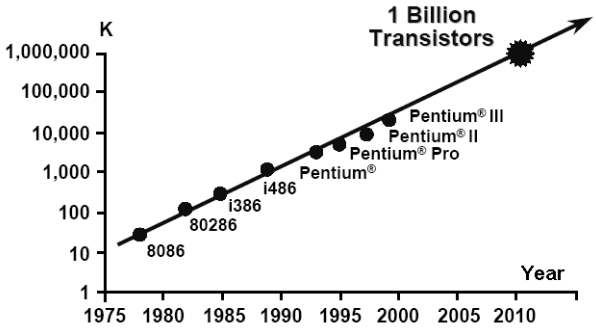
High-level synthesis process can be defined as the translation process from behavioral description to its structural description [3, 14, 4, 15]. This is analogous to a ”compiler” that translates a high-level language program in C/Pascal to an assembly language program. High-level synthesis is also known as behavioral-level synthesis or algorithm-level synthesis. The constraints which are to be considered in high-level synthesis are area, performance, power consumption, reliability, testability and cost. With the increasing demand for personal computing devices and wireless communications equipment, the demand for designing low power consuming circuits has increased. ”Power” has become an important parameter alongwith area and throughput. The need for low power synthesis is driven by several factors [16, 17, 18, 19, 20]:



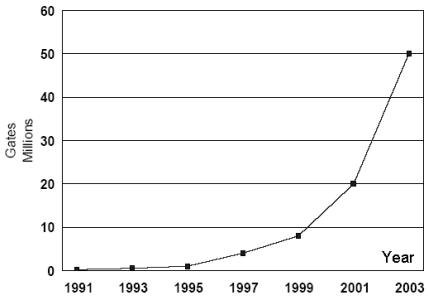
(a) Increase in Power [8, 9, 10]



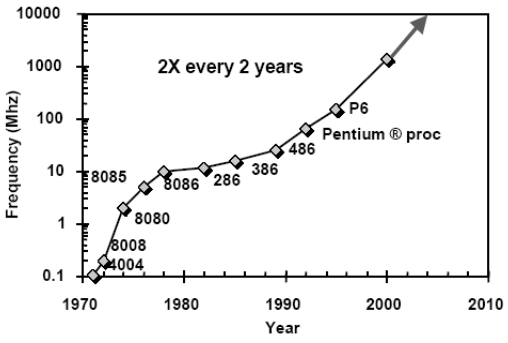
(b) Increase in Power Density [9, 11, 10]



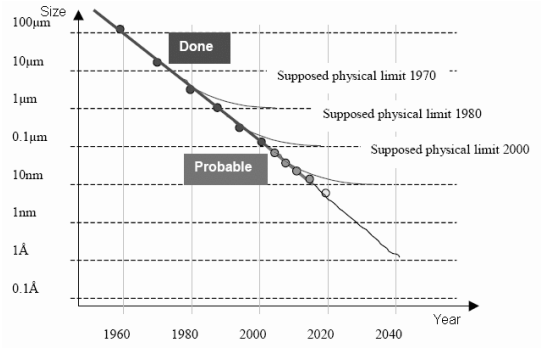
(c) Increase in Transistor Count [11, 10]



(d) Increase in Gate Count [12]



(e) Increase in Frequency [11, 10]



(f) Decrease in Feature Size [11, 10, 13]

Figure 1.1. Chronological Change in Power, Power Density, Transistor Count, Gate Count, Operating Frequency and Feature Size of CMOS Integrated Circuits

- Increased demand for portable systems: Emergence of portable devices like laptop computers, mobile phones etc. for which battery life is an important factor
- Thermal considerations: If power dissipation can be reduced, the cost of cooling and packaging would be reduced.
- Environmental concerns: The smaller the power dissipation in a circuit, lesser the heat pumped into the rooms. So, the electricity consumption will be lower and impact on the environment will be less.
- Reliability issues: If the power consumption is higher, the temperature in the circuit is increased. This may lead to phenomenon like electromigration and hot-electron effects. This causes reduction in the reliability of the system. In fact, it is seen that for every 10^oC rise in operating temperature, roughly doubles the failure rate of the components.

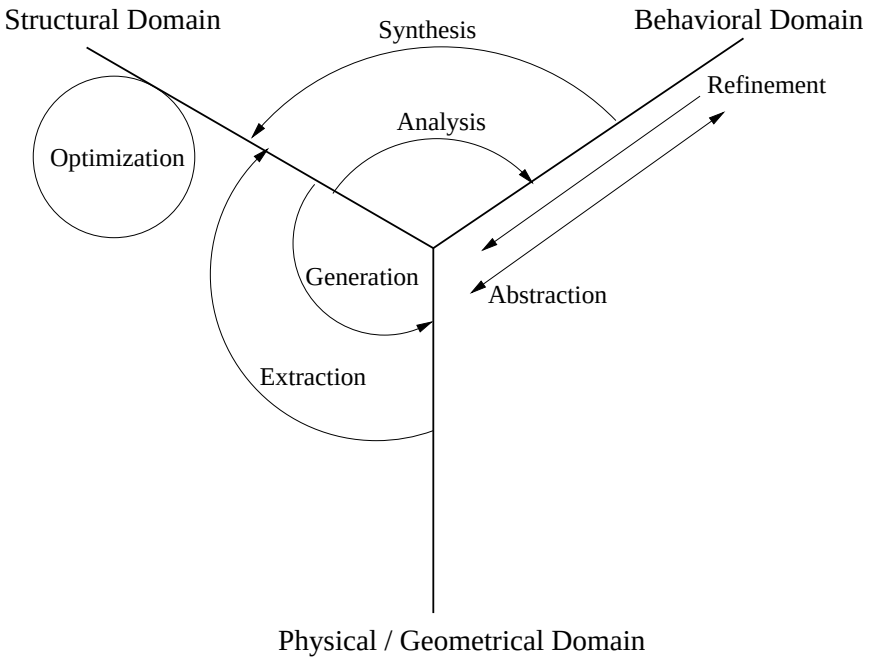
The growth of high speed computer networks and that of the internet, in particular, has explored means of new business, scientific, entertainment, and social opportunities. Ironically, the cause for the growth is also of the apprehension - use of digital formatted data. Digital media offer several distinct advantages over analog media, such as high quality, easy editing, high fidelity copying. The ease by which a digital information can be duplicated and distributed has led to the need for effective copyright protection tools. Various software products have been recently introduced in attempt to address these growing concerns. It is done by hiding metadata (information) within digital audio, images and video files. One way of such data hiding is *digital signature*, *copyright label* or *digital watermark*, that completely characterizes the person who applies it and, therefore, marks it as being his intellectual property. *Digital Watermarking* is the process that embeds data called a watermark into a multimedia object such that watermark can be detected or extracted later to make an assertion about the object. While the software implementation of digital watermarking techniques are enormously large, the hardware of the same is negligibly small. The hardware implementation has advantages over the software implementation in terms of low power, high performance and reliability. Also, the hardware implementation of watermarking techniques is absolutely essential for real-time watermarking applications, such as of digital TV broadcasting.

This chapter presents a general overview of high-level synthesis and power minimization in VLSI circuits. The chapter is organized as follows. Section 1.1 discusses high-level synthesis in general and motivation behind high level synthesis. The various sources of power consumption are discussed in Section 1.2. The possible methods of power reduction are described in Section 1.3. Section 1.4 discusses why we need to minimize peak power. The need for average power and energy reduction is listed in Section 1.5 and that of transient power is in Section 1.6. Section 1.7 discusses how frequency and voltage scaling can reduce energy / power in a circuit. The fundamentals of digital watermarking is discussed in Section 1.9. The design issues for multiple supply voltage and dynamic frequency clocking based circuits are discussed in Section 1.8. Section 1.10 discusses the contribution of this dissertation. The dissertation outline is given in Section 1.11.

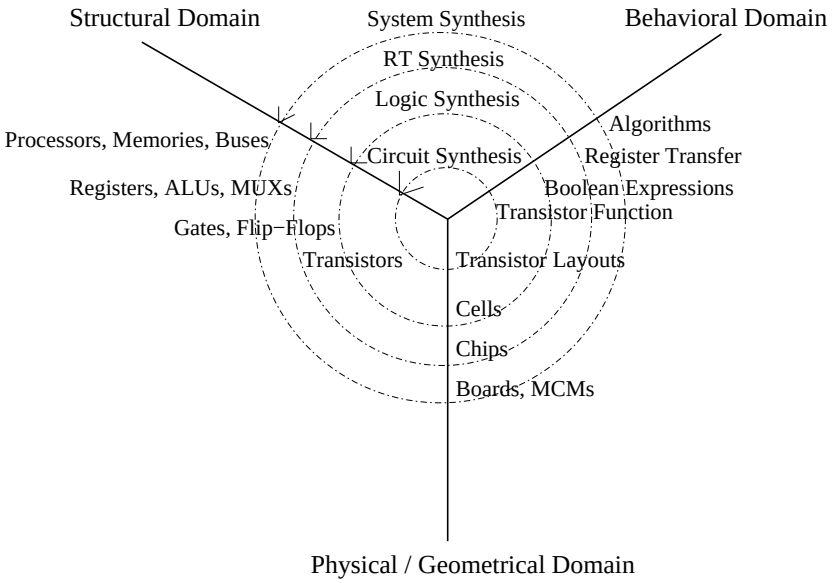
1.1 Fundamentals of High Level Synthesis

In circuit analysis, we study the behavior or characterisitcs of a circuit. Synthesis process is the reverse of analysis process. The task of synthesis process is to take the specifications of the behavior required for a system and a set of constraints and goals to be satisfied, and to find a structure that implements the behavior while satisfying the goals and constraints [3, 4, 15, 21]. The ”behavior” of the system refers to the ways in which the system or its components interact with their environment (mapping from inputs to outputs). The ”structure” refers to the set of interconnected components that constitute the system (described by a netlist). Finally, the structure must be mapped into a ”physical” design. Behavior, structure and physical design are considered as three domains in which a hardware can be described (Fig. 1.2(a) and 1.2(b)). In behavioral domain, we are interested in what a design does, not in how it is built. The physical domain ingnores what the design is supposed to do and binds its structure in space or to silicon. A structual representation bridges the behavioral and physical representation. It is one-to-one mapping of a behavioral representation onto a set of components and connections under constraints, such as area, cost and delay.

Fig. 1.2(a) describes the design automation terminologies, such as optimization, synthesis, analysis, and optimization in the hardware representation domain. The axes in Y-chart (Fig. 1.2(b))



(a) Y-chart : Anaylsis, Optimization or Synthesis



(b) Y-chart : Detailed Hardware Description

Figure 1.2. Desription of Hardware in Different Domains and Abstractions [4]

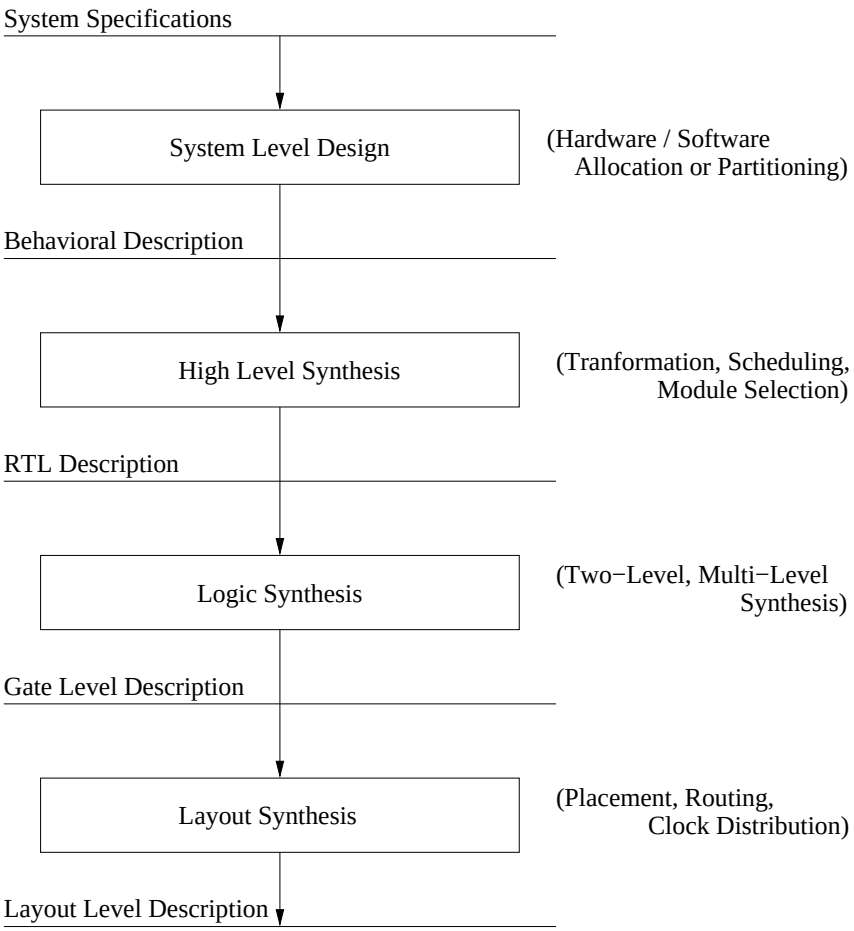


Figure 1.3. Synthesis Flow

represent three different domains of description, such as behvaioral, structural and physical. Each concentric circle intersects the axes at a particular level of representation within a domain. It may be noted that the synthesis process is a transformation from the behavioral domain to the structual domain, which is represented as an arc in Fig. 1.2(a).

The digital circuits are designed and synthesised at several levels of abstraction as shown in Fig. 1.3.

- System Level: The system level is concerned with the overall system structure and information flow. Computer systems are described as interconnected set of processors, memories and switches in this level.

- Behavioral Level: This level is also called as Instruction Set Level or Algorithmic Level. At this level the focus is on the computations performed by an individual processor, the way it maps sequences of inputs to sequences of ouputs.
- Register Transfer Level: The system is viewed as a set of interconnected storage elements and functional blocks in this level. The behavior of system is described as a series of data transfers and transformations between the storage elements.
- Logic Level: Below the register transfer level is the logic level. The system is described as a network of gates and flip-flops and the behavior is specified by logic equations at this level.
- Layout Level: In this level, the system is specified in terms of the individual transistors of which it is composed. The behavior of the system can be described in terms of the network equations.

1.1.1 Why High-Level Synthesis ?

High-level synthesis is popular for the following reasons [3]:

- Shorter design cycle: If more of the design process is automated, faster products can be made available at cheaper prices.
- Fewer errors: Since the synthesis process can be verified easily, the chances of getting errors will be less.
- Ability to search the design space: As synthesis system can produce several designs in a small time, the designer has more flexibity to choose proper design considering different trade-offs.
- Documenting the design process: An automated system can keep track of design decisions and effect of those decisions.
- Availability of IC technology to more people: As design expertise is moved into synthesis system, it becomes easier for a non-expert to produce a chip that meets a given set of specifications.

1.1.2 Various Phases of High-Level Synthesis

The various phases of high-level synthesis include, compilation, transformation, scheduling, allocation, binding as detailed in Fig. 1.4.

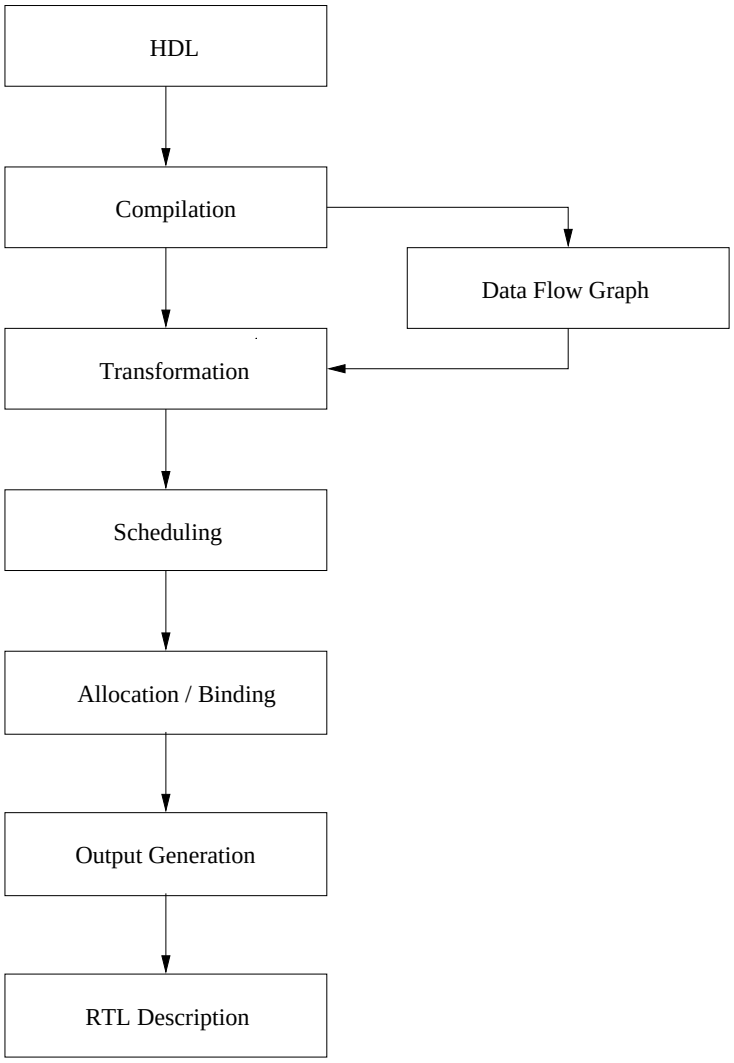


Figure 1.4. Various Phases of High-Level Synthesis

The behavior of a system to be synthesized is usually specified at the algorithmic level using a high-level programming language like Pascal, C or a hardware description language such as VHDL and Verilog [3, 22]. The behavior of the system is then compiled into internal representations, which are usually data flow graphs (DFGs) and control flow graphs (CFGs). Each behavioral specification is transformed into an unique graphical representation. The data flow graph is a

directed graph which represents the data moves, while the control flow graph is a directed graph which indicates the sequence of operations. The formal definitions of data flow graph and control flow are given below [3].

A data flow graph (DFG) is a directed graph $G = (V, E)$, where:
(i) $V = v_1, v_2, ..., v_n$ is a finite set whose elements are "nodes", and
(ii) $E = V \times V$ is an asymmetric "data flow relation",
whose elements are directed "data edges".

A control flow graph (CFG) is a directed graph $G = (V, E)$, where:
(i) $V = v_1, v_2, ..., v_n$ is a finite set whose elements are "nodes", and
(ii) $E = V \times V$ is a "control flow relation",
whose elements are directed "sequence edges".

Lets consider the following algorithm that computes the square root of X using Newton's method [3].

Algorithm : Square Root Calculations
{
$Y := 0.22 + 0.89 * X;$
$I := 0;$
Do until $I > 3$ loop
$Y := 0.5 * (Y + \frac{X}{Y});$
$I := I + 1;$
End do
}

The above algorithm can be represented using the following data flow graph and control flow graph (Fig. 1.5).

In the transformation step, the initial data flow graph is transformed so that the resultant data flow graph is more suitable for scheduling and allocation. These transformations include compiler-like optimizations such as dead code elimination, common subexpression elimination, loop un-

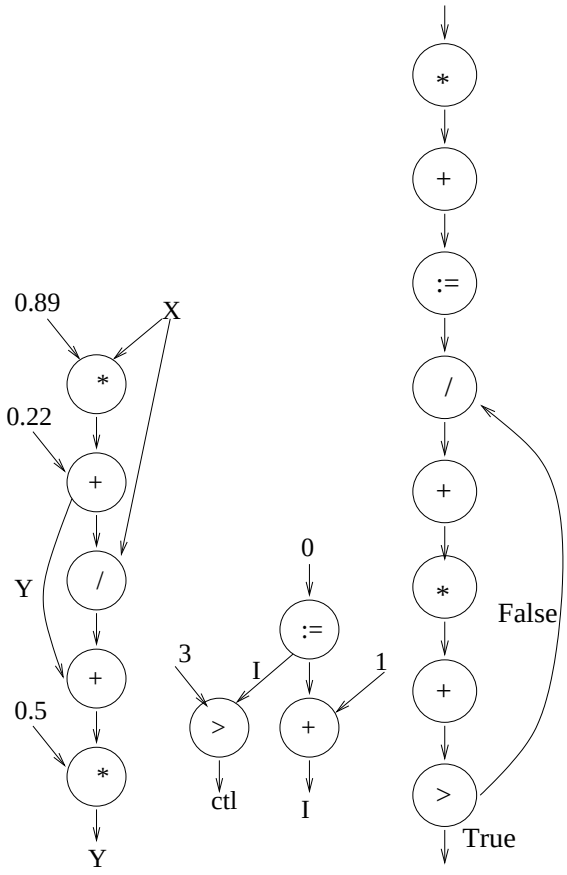


Figure 1.5. Data Flow Graph and Control Flow Graph of a Square Root Algorithm [3]

rolling, constant propagation and code motion. In addition to this, some hardware-specific transformations like syntactic variances minimization, retiming may be applied to to take advantage of the associativity and commutativity of certain operations.

Scheduling is the process of partitioning the set of arithmetic and logical operations in the data flow graph into groups of operations so that the operations in the same group can be executed concurrently, while taking into consideration possible trade-offs between the total execution cost and hardware cost. A group of concurrent computations to be executed simultaneously is referred to as control step. The total number of control steps needed to execute all operations in the data

flow graph, the minimum number of functional units of each type to be used in the design, and the lifetimes of the variables generated during the computation of operations are determined in the scheduling step. Datapath scheduling algorithms may be of various types based on the constraints and optimization schemes as shown in Fig. 1.6. Various scheduling algorithms are described in [4, 21, 22, 3, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 2, 34, 35, 36, 37, 38]. The commonly used scheduling techniques are integer linear programming, as-soon-as possible, as-late-as possible, list-based scheduling, force directed scheduling and freedom-based scheduling, etc.

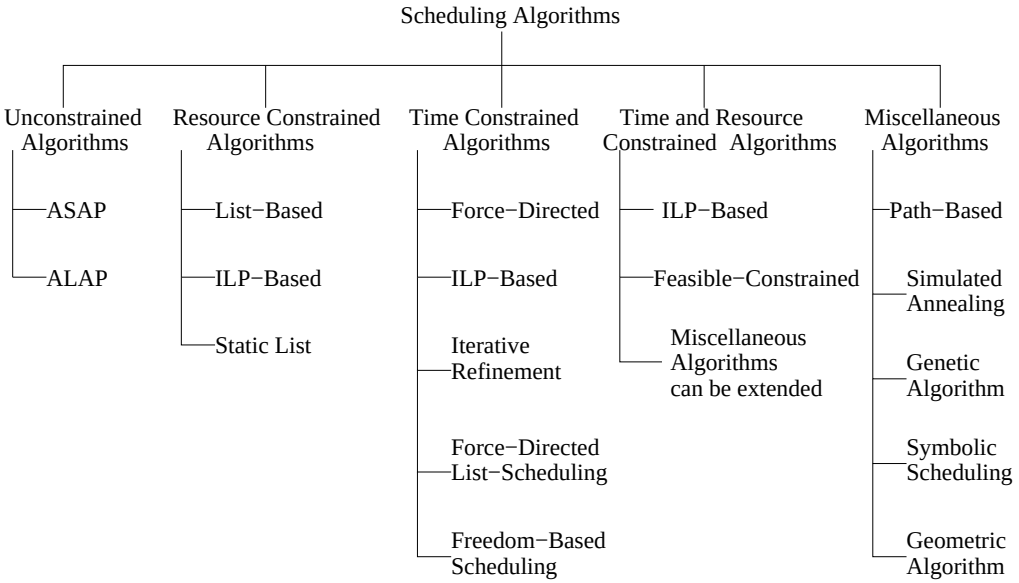


Figure 1.6. Different Types of Scheduling Algorithms

Allocation is the process of determining functional units of each type for performing operations, memory units(registers) for storing data values, and interconnects for data transportation. Binding is the process of assigning variables to memory units, and data transfers to interconnections. Allocation / binding is further divided into tasks, such as functional unit allocation / binding, memory unit allocation/binding and interconnect allocation / binding. The functional unit allocation / binding involves the mapping of operations in the behavioral description into a set of selected functional units. The memory unit allocation / binding maps data carriers(constants, variables, arrays) in the behavioral description onto storage elements(ROMs, registers, memory units) in the

datapath. The interconnect allocation / binding task maps every data transfer in the behavior into a set of interconnection units for data routing.

In the output generation phase, design output is generated. The output should be in a form, so that logic-level synthesis tools can optimize the combinational logic, and layout synthesis tools can design the chip geometry . The generated output is generally in a low level hardware description language, such as structural VHDL or EDIF [22].

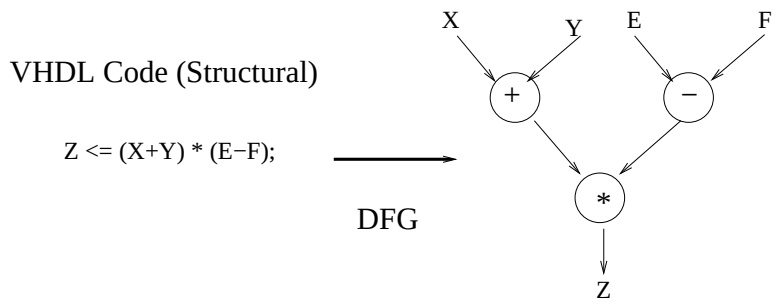
1.1.3 A Synthesis Example

Let us consider a small synthesis example to learn the various phases of synthesis in detail. Suppose, we want to synthesize hardware to perform the operation : $Z = (X + Y) * (E - F)$. The following self explanatory Figs. (1.7 -1.8) illustrate the steps.

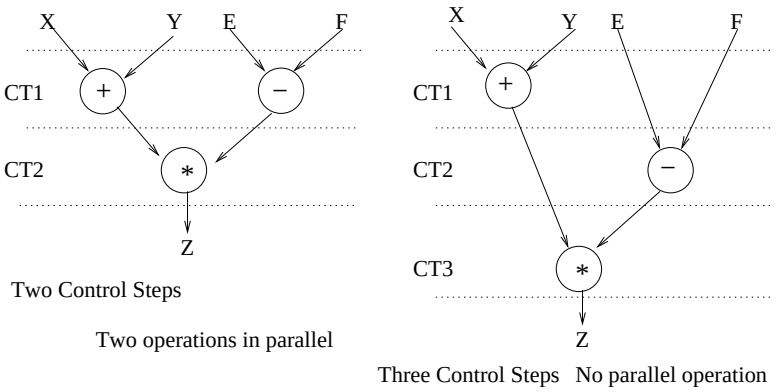
1.2 Sources of Power Dissipation in a CMOS Circuit

The details of power dissipations are shown in Fig. 1.9. Power dissipation in a CMOS circuit is caused by four sources [17] :

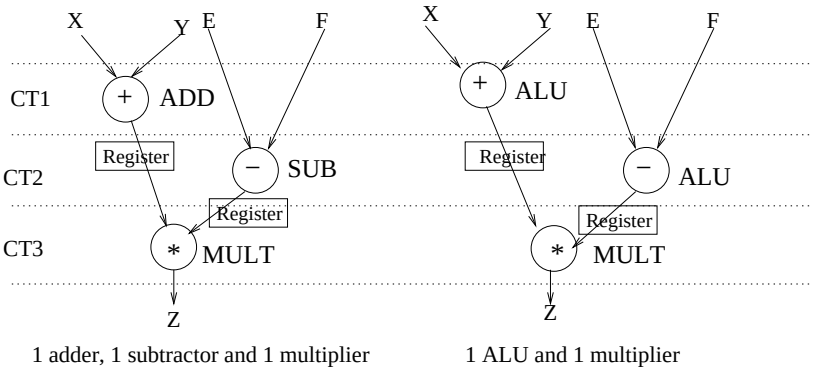
- Leakage current: It is determined by the fabrication process technology and consists of two components: (1) reverse bias current in the parasitic diodes formed between source and drain diffusions and the bulk region in the transistor, and (2) the subthreshold current that arises from the inversion charge that exists at the gate voltages below the threshold voltage.
- Standy current: It is the DC current drawn continuously form V_{dd} to ground.
- Short-circuit current: This is the current due to the DC path between the supply and ground during output transitions.
- Capacitance current: This curent flows to charge and discharge capacitance loads during logic changes.



(a) Step1: Compilation and Transformation

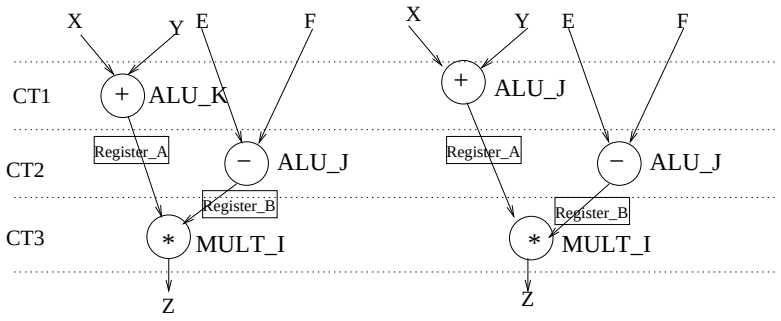


(b) Step2: Scheduling (Time or Resource Constraints)

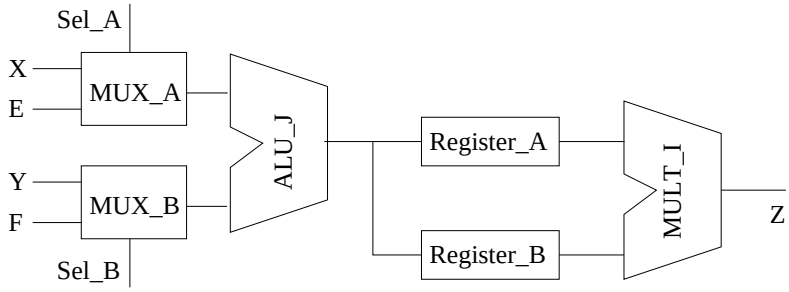


(c) Step3: Allocation (Fixes Amount and Types of Resources)

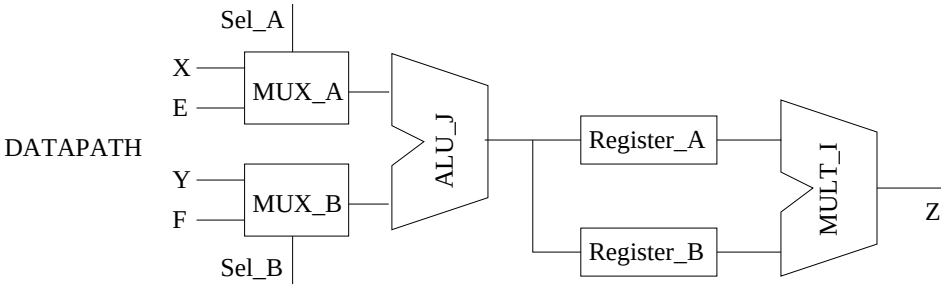
Figure 1.7. A Synthesis Example : Step 1 to Step 3



(a) Step4: Binding (which Resource will be used by which Operation)



(b) Step5: Connection Allocation (Communication between Resources: Bus, Buffer or MUX)



CONTROL

CT1 − Action A = X + Y
Signals : Sel_A, Sel_B, load(Reg_A)

CT2 − Action : B = E −F
Signals : Sel_A, Sel_B, load(Reg_B)

CT3 − Action : Z = A * B
Signals : load(Reg_Z)

(c) Step6: Architecture Generation (Datapath and Control)

Figure 1.8. The Synthesis Example : Step 4 to Step 6

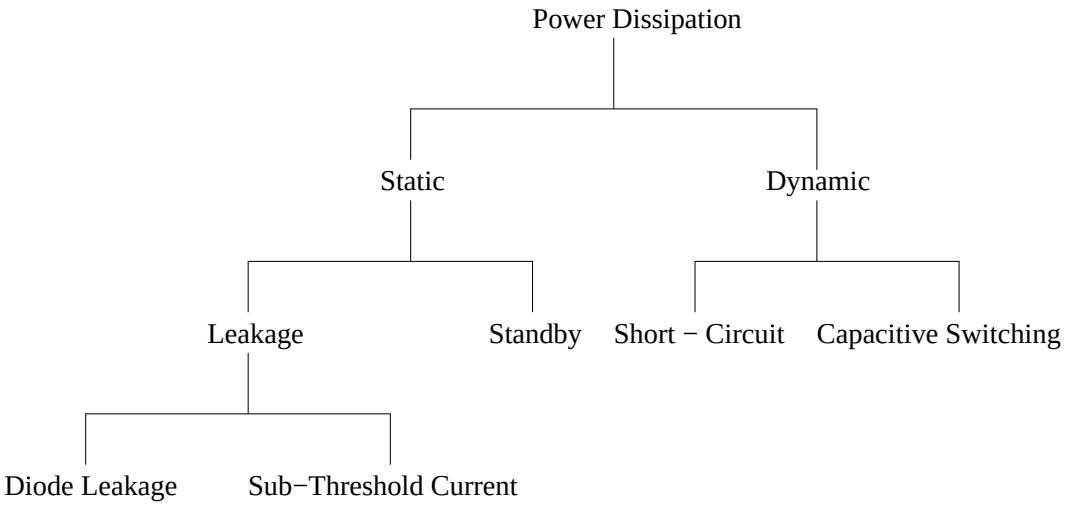


Figure 1.9. Sources of Power Dissipation in a CMOS Circuit

Capacitive switching power dissipation is caused by charging and discharging of parasitic capacitance in the circuit and is given by Eqn. 1.1,

$$P_{dynamic} = \frac{1}{2} C_L V_{dd}^2 N f \tag{1.1}$$

where, C_L is load capacitor, V_{dd} is supply voltage, N is average or expected number of transitions per clock cycle (switching activity), and f is the clock frequency. During transition from either 0 to 1 or 1 to 0, both NMOS and PMOS are ON for a short period of time. Because of this there is flow of current from V_{dd} to V_{ss} (short current pulse). The power dissipation corresponding to this is called *short-circuit power* dissipation which is quantified as in Eqn. 1.2

$$P_{short} = \frac{\beta}{12} (V_{dd} - 2V_t)^3 \frac{t_{rf}}{t_p} \tag{1.2}$$

where, β is the transistor gain factor, V_{dd} is supply voltage, V_t is the threshold voltage, $t_{rf}$is the rise/fall time (under the assumption that $t_r = t_f$) and t_p is the period of the input waveform. The *dynamic power dissipation* is the sum of the short-circuit and capacitative power dissipations.

The *leakage power* dissipation occurs because of reverse-biased diode (formed between diffusion regions and substrate) current and subthreshold current. Leakage currents in CMOS circuits

can be made small with the proper choice of device technology. *Standby power dissipation* happens when both the nMOS and pMOS transistors are continuously on in a psuedo-nMOS inverter, when the drain of an nMOS transistor is driving the gate of another nMOS transistor in a pass-transistor logic, or when the tristated input of a CMOS gate leaks away to a value between V_{dd} and ground. The *static-circuit power dissipation* is the sum of the leakage and standby power dissipations. The total static power of a CMOS circuit is obtained using the Eqn. 1.3 as given below (assuming n number of transistors). In practice, standby power is neglected compared to the leakage power and static power is assumed to be the leakage power.

$$\begin{aligned} P_{static} &= \sum_{i=1}^{n} \text{leakage current} * \text{supply voltage} \\ &= \sum_{i=1}^{n} \left(I_{\text{diode}} + I_{\text{subthreshold}}\right) * \text{supply voltage} \end{aligned} \tag{1.3}$$

1.3 Methods for Power Reduction in High-Level Synthesis

Leakage power dissipation is small in comparison to other components. In a well designed circuit, short-circuit power dissipation is less than 20% of dynamic power [39]. It is also evident from Fig. 1.10 [6, 7] that at larger switching activity the static power is negligible compared to the dynamic power dissipation. This shows that the dynamic power dissipation is the the main power dissipation that needs to be taken care of. From the dynamic power dissipation expression given in Eqn. 1.1, we can conclude that the parameters that can be varied to affect power as well as energy consumption are :

- supply voltage,
- the clock frequency,
- the switching activity per clock cycle at various signals in the circuit,
- the parasitic capacitance.

It is important to note that these parameters are not independent. It is necessary to take into account the interactions and trade-offs among these parameters to minimize power consumption [17]. The key principles used for low-power design are as follows [20, 40] :

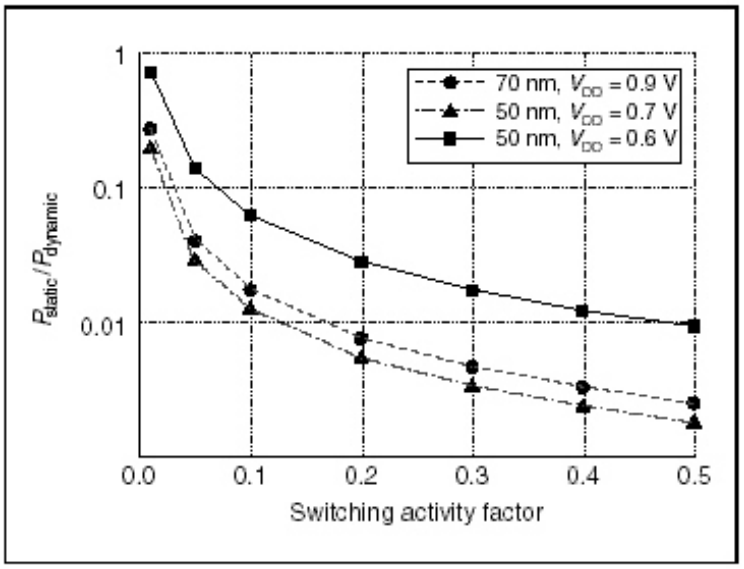


Figure 1.10. Static Vs Dynamic Power Dissipation for Different Switching Activity [6, 7]

- using the lowest possible supply voltage
- using the smallest geometry, highest frequency devices, but operating them at lowest possible frequency,
- using parallelism and pipelining to lower required frequency of operation,
- power management by disconnecting the power source when the system is idle, and
- designing systems to have lowest requirements on subsystem performance for the given level functionality.

Based on the above observations, following are the some techniques used to reduce power consumption in high-level synthesis [41, 22, 1, 9, 42, 40].

- Transformation: The basic approach is to scan the design space by utilizing various flow graph transformations with high-level power estimation techniques, and transform data flow graphs into less power consuming data flow graphs.

- Operator shutdown: The massive switching in large components, such as adders, multipliers and registers, consume a large amount of power. By disabling the clock signal the internal nodes remain at static voltage levels and do not consume power.

- Lower supply voltages: In a CMOS circuit, power consumption decreases quadratically with voltage while the speed reduction is linear. When intensive computation is not needed, the supply voltage is lowered and consequently can save power consumption.

- Mixed voltage circuit: Dual voltages on one IC are attractive enough for commercial consideration. Although such an approach is viable, designers must carefully consider cross-talk and latch-up issues among others.

- Increased parallelism: Slower operations can be used on non-time critical paths, while parallelism can be increased to compensate for slower components. The parallel option consumes less power and has a shorter total delay. However, extra area might be needed to achieve the parallelism.

1.4 Why Peak Power Minimization ?

With the increase in chip densities and clock frequencies the demand for design of low power integrated circuits has increased. The literature is rich on efforts to reduce total energy consumption and average power consumption of the CMOS circuits. At the same time, the reduction of peak power consumption is essential for the following reasons [43, 5, 8, 44, 45, 46] :

- to maintain supply voltage levels,

- to increase reliability and

- smaller heat sinks and cheaper packaging.

The peak power is the maximum power consumption of the integrated circuit (IC) at any instance during its execution. If the current flow is large, then the IR drop of the interconnects becomes large which can reduce the supply voltage levels at different parts of a IC. High current flow can

reduce reliability because of hot electron effects and high current density. The hot electron effects may lead to runaway current failures and electrostatic discharge failures. High current density can cause electromigration failure. It is observed that the mean time to failure (MTF) of CMOS circuit is inversely proportional to current density (or power density). If the current (power) dissipation is large, then the heat generated out of the system is large. This in turn, needs bigger sink and costlier heat dissipation mechanism in order to maintain the operating temperature of the ICs in its tolerance limit.

1.5 Why Average Power and Energy Reduction ?

Energy and average power reduction is essential for the following reasons [17, 8, 5, 46]:

- to increase battery life time,
- to enhance noise margin,
- to reduce cooling and energy costs,
- to reduce use of natural resources, and
- to increase system reliability.

The battery life time is determined by the Ah (ampere hour) rating of the battery. If the average power (and/or energy) consumption is high, then battery life time may reduce because of high ampere consumption. This factor is important for portable applications. The reduction of average power is essential to enhance noise margin (to decrease functional failure). The cost of packaging and cooling is determined by average current flow and hence, the average power and energy. The high energy consumption of the computer systems leads to environment concerns due to the need for more power generation. If the average power is large, the operating temperature of the chip increases, which may lead to failures. It is estimated that for each 10^oC increase in the operating temperature, the failure rates of the components is roughly doubled.

1.6 Why Transient Power Minimization ?

Both the peak power and peak power differential describe the transient power characteristics of a CMOS circuit. In the above section we discussed the needs for peak power reduction. The peak power differential needs to be reduced for the following reasons [8, 5, 47, 48]:

- to reduce power supply noise,
- to reduce cross talk and electromagnetic noise,
- to increase battery efficiency and
- to increase reliability.

Power fluctuation leads to larger $\frac{di}{dt}$ causing power supply noise, (similar to IR drop), because of self inductance of power supply lines. Crosstalk is the noise voltage induced in signal line due to the switching in another signal line [5]. The voltage induced by the mutual inductance is expressed as $L\frac{di}{dt}$ and that induced by the mutual capacitance as $C\frac{dv}{dt}$. If the power fluctuation is high, then large $\frac{di}{dt}$ and $\frac{dv}{dt}$ can introduce significant noise in the signal lines. As the power fluctuation increases, it reduces the electrochemical conversion and hence there is decrease in battery life [49]. High current peaks (power fluctuation) in short time spans can cause high heat dissipation in a localised area of silicon die which may lead to permanent failure of the integrated circuit.

1.7 Why Frequency and Voltage Scaling ?

With the increasing demand for portable electronic devices, power reduction has emerged as a major design goal in VLSI circuits. Let us consider the following equations for a CMOS circuit [50, 51, 52, 53, 54, 55, 56] :

- Energy dissipation per operation is

$$E_{dynamic} = C_{eff}V_{dd}^2 \tag{1.4}$$

where, C_{eff} is the effective switched capacitance and V_{dd} is the supply voltage,

- Power dissipation for the operation is

$$P_{dynamic} = C_{eff} V_{dd}^2 f \tag{1.5}$$

  where, f is the frequency.

- Further, the critical delay (t_d) in a device that determines the maximum frequency (f_{max}) is

$$t_d = k \frac{V_{dd}}{(V_{dd} - V_t)^\alpha} \tag{1.6}$$

  where, V_T is the threshold voltage, α is a technology dependent factor and k is a constant.

From the above three equations, the following can be deduced [50, 52, 57, 9, 54, 55, 58] :

- By reducing only V_{dd}, both energy and power can be saved at the cost of performance (speed / time).
- Slowing down CPU by reducing only f will save power but not energy.
- However, by scaling frequency and voltage in a coordinated manner, both energy and power can be saved while maintaining performance.

The third factor above forms the major motivation for this work. The objective is to generate a datapath schedule that attempt at energy and power reduction without degrading the performance by using multiple voltages and dynamic frequency clocking in a co-ordinated manner. Moreover, simultaneous voltage and frequency reduction opens oppurtunity for power reduction in three folds. In this dissertation, we investigate the power and energy reduction due to combined use of multiple supply voltages, dynamic frequency clocking, and multicycling.

1.8 Multiple Supply Voltages, Dynamic Clocking and Multicycling Preliminaries

In Section 1.7, we have seen that voltage and frequency need to be varied in a co-ordinated manner to get better results in terms of power, energy or performance. Dynamic frequency clocking is a mechanism to vary clock frequency on the fly depending on the computation. In multiple supply

voltage scheme, different modules or functional units are operated at different supply voltages. Similarly, variable voltage scheme is a technique in which the operating voltage is valid from time to time. This chapter discusses how energy and power reduction can be achieved through the use of dynamic frequency clocking, voltage scaling multicyling. Further, the design related issues of having multiple supply voltages in a processor are discussed. Design of level converters and dynamic frequency clocking units are also presented.

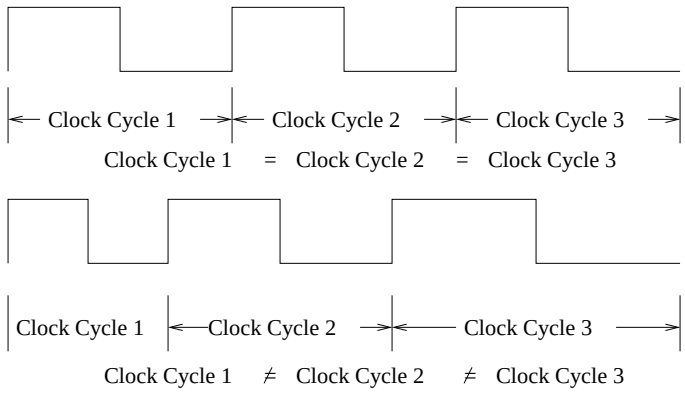
1.8.1 What is Dynamic Frequency Clocking ?

In dynamic frequency clocking, the functional units can be operated at different frequencies depending on the computations occuring within the datapath during a given clock cycle. The strategy is to schedule high energy units, such as multipliers at lower frequencies such that they can be operated at lower voltages to reduce energy consumption and the low energy units, such as adders at higher frequencies, to compensate for speed. In this clocking scheme, all the units are clocked by a single clock line which switches at run-time. A clocking mechanism that varies the clock frequency dynamically has been shown to improve the execution time as compared to using a uni-frequency global clock [59]. Generation of such types of clocks have been studied extensively in [60, 61, 62, 63]. Fig. 1.11(a) shows the uni-frequency and dynamic frequency diagrams.

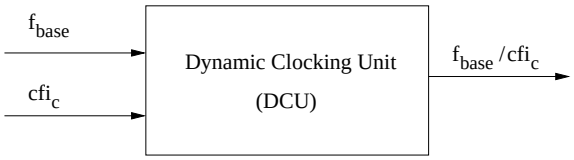
The dynamic clocking unit (DCU) which generates the required clock frequency uses a clock divider strategy to generate frequency which are submultiples of the base frequency. Base frequency f_{base} is the maximum frequency (or multiple of maximum) of any functional unit (FU) at the maximum supply voltage. A value cfi_c (cycle frequency index for control step c) is loaded as an input to the DCU which comes from controller. The scheme for dynamic frequency generation is shown in Fig. 1.11(b). Loading a value of cfi_c into the counters provide a divided output clock of frequency $\frac{f_{base}}{cfi_c}$.

1.8.2 Energy or Power Reduction Due to Voltage or Frequency Scaling

To understand how multiple supply voltage, variable frequency and multicycling can be helpful in energy or power reductions, let us consider the small data flow graph shown in Fig. 1.12(a).



(a) Single Frequency Vs Dynamic Frequency



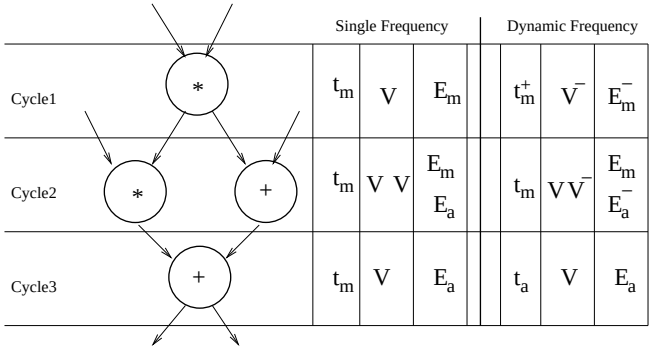
(b) Dynamic Frequency Generation

Figure 1.11. Dynamic Frequency Generation using Dynamic Clocking Unit [54]

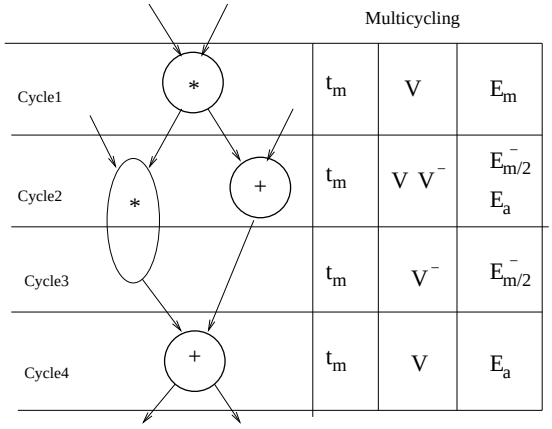
Let us analyse the power, energy consumption for this data flow graph in three possible modes of datapath operation, such as (i) single supply voltage and single frequency, (ii) multiple supply voltage and variable or dynamic frequency, and (iii) multiple supply voltage and multicycling [54, 55, 64]. Let t_a and t_m be the delays of the adder and the multiplier respectively at the maximum supply voltage V. The DFG is scheduled to three control steps.

Single supply voltage and single frequency (SVSF) : Each cycle has clock width determined by the slowest operator delay t_m. The total energy consumption is given by $E_{sf} = 2E_m + 2E_a$ and the total delay is $T_{sf} = 3t_m$. In this case, the peak power consumption is given by, $P_{peak,sf} = \frac{E_m+E_a}{t_m}$.

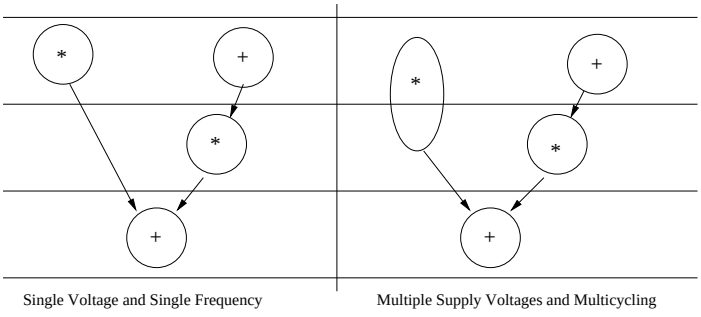
Multiple supply voltages and dynamic frequency (MVDFC) : Let, E_m^- and E_a^- are some energy values less than E_m and E_a respectively and t_m^+ be the delay of the multiplier at lower voltage V^-. In data flow graph shown in Fig. 1.12(a), assuming that, the clock cycle width for the 3^{rd} cycle is t_a which is smaller than t_m. This allows us to increase the clock width of some other cycles from t_m to some t_m^+ without violating the time constraints (or without time penalty). In this case, the total



(a) Data Flow Graph : Variable Frequency Vs Single Frequency



(b) Data Flow Graph : Multicycling $\rightarrow$ Performance Degradation



(c) Data Flow Graph : Multicycling $\rightarrow$ No Performance Degradation

Figure 1.12. Data Flow Graph in Three Modes of Operation

delay $T_{dfc} = t_m^+ + t_m + t_a$ and the energy consumption is given by $E_{dfc} = E_m + E_a + E_m^- + E_a^-$. Since, $T_{dfc} \approx T_{sf}$ and $E_{dfc} < E_{sf}$, energy reduction is achieved without degrading performance. Energy overhead of level converters have to be considered for this case. The peak power consumption is given by, $P_{peak,dfc} = \frac{E_m + E_a^-}{t_m}$.

Multiple supply voltages and multicycling (MVMC) : In this mode of operation, the functional units are operated at multiple supply voltages. The functional units operating at low voltage are made to run in more than one consecutive control steps. Let us assume that multiplier takes two control steps, when it is operated at a lower supply voltage. The example data flowgraph for the multicycling case in shown in Fig. 1.12(b). In this case, the total energy consumption $E_{mc} = E_m + E_m^- + 2E_a$ and total delay $T_{mc} = 4t_m$. Since, $T_{mc} > T_{sf}$ and $E_{mc} < E_{sf}$, energy reduction is obtained with a degradation in performance of the circuit. For the multicycling case, level converters are the only overheads. The peak power consumption of the DFG will be determined by the multiplication operation in control step 1, $P_{peak,mc} = \frac{E_m}{t_m}$. This is based on the observation that the power consumption of the multipliers are much higher than that of the adders. It may be noted the above mentioned performance degradation may not always happen. For example, consider a DFG such as the one shown in Fig. 1.12(c); although the multiplier is scheduled in two control steps there is no change is the critical path delay. The delay is $3t_m$ for both SVSF and MVMC cases.

1.8.3 Issues in Multiple Supply Voltage Based Design

A designer needs to take into consideration several design issues when a multiple voltage design is targeted for fabrication. The effects of multiple voltage operation on IC layout and power supply requirements should be considered [65, 66, 67]. Multiple voltage design may affect IC design in the following ways :

- If the multiple supplies are generated off-chip, additional power and ground pins will be required.

- It may be necessary to partition the chip into separate regions, where all modules in a region operate at the same voltage.
- Some kind of isolation will be required between the regions operated at different voltages.
- There may be some limit on the voltage difference that can be tolerated between the regions.
- Protection against latch-up may be needed at the logic interfaces between regions of different voltages.
- New design rules for routing may be needed to deal with signals at one voltage passing through a region at another voltage.
- Choice between generating the voltage on-chip or off-chip has to be made depending on the application.
- Clocking scheme needs to be modified.

1.8.4 Level Converter Design

We already know that whenever one resource has to drive an input of another resource operating at a different voltage, a level conversion is needed. Thus, level-converter or level-shifter is the most essential component for multiple supply voltage designs. This results in overheads in the form of area and power for multiple supply voltage designs as compared to single supply voltage designs. Four possible alternatives are used by various researchers as listed below [65].

- The level conveters can be omitted.
- A chain of inverters can be used at successive higher voltages.
- An active or passive pullup can be used.
- A differential cascode voltage switch (DVCS) can be used.

Various level converter designs have been discussed in [66, 68, 69, 67, 65]. We implemented the level converter design proposed in [65, 66] to get better understanding. The schematic diagram,

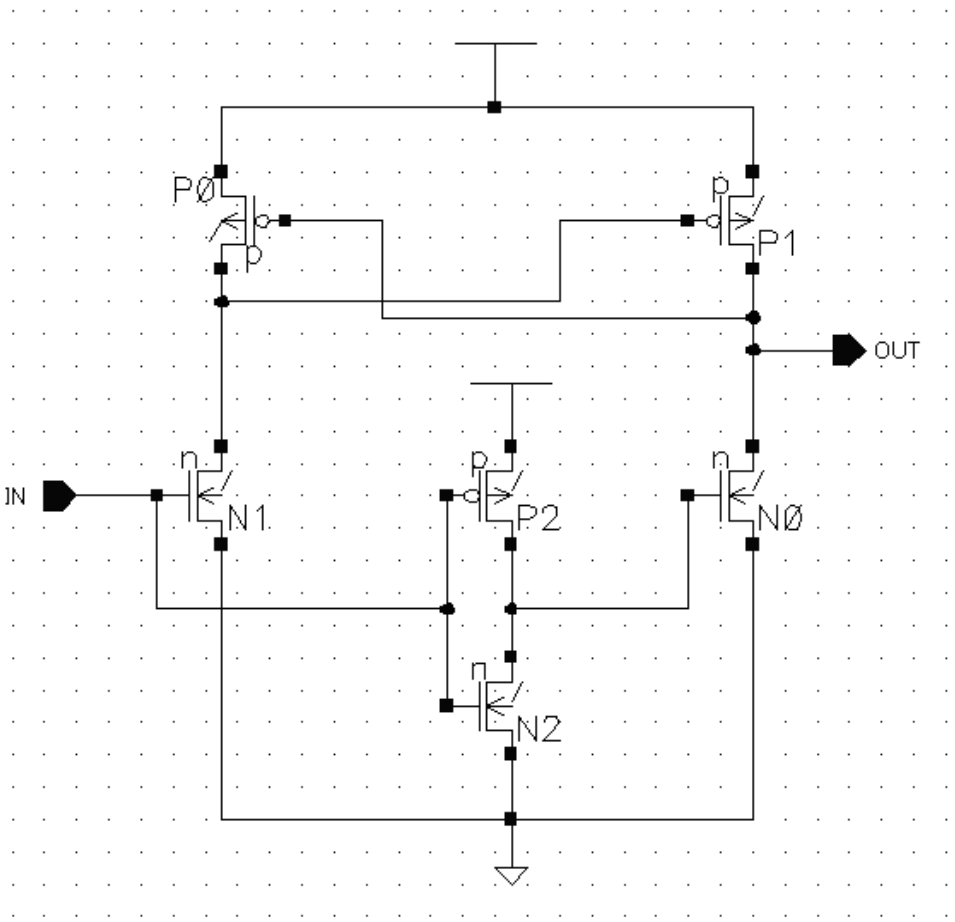
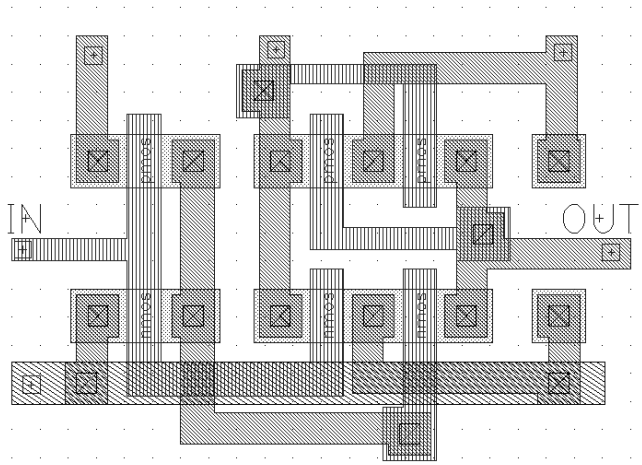


Figure 1.13. Level Converter Schematic Diagram [65, 66]

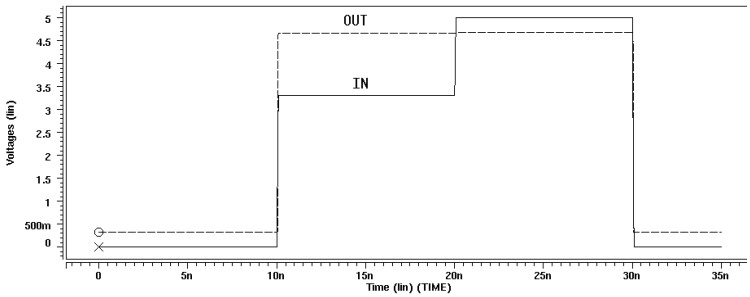
the layout and the simulation waveform is given in Fig. 1.13, 1.14(a) and 1.14(b) respectively. The constant output voltage indicates that the level converter can step up or step down the voltage to produce a constant supply voltage.

1.8.5 Dynamic Frequency Clocking Unit Design

Dynamic frequency scaling is an efficient power reduction method with large potential power savings. In order to exploit dynamic frequency scaling for energy or power reduction, a clock divider is needed to safely change the clock rates. In this section, the design of two such dynamic frequency clocking units present in the existing literature [59, 61] are described.



(a) Level Converter Layout



(b) Level Converter Simulation Waveform

Figure 1.14. Level Converter Layout and Simulation

Ranganathan, Vijaykrishnan and Bhavanishankar [59] introduce the concept of dynamic frequency clocking. The DFC scheme is more suitable for data flow intensive application (such as DSP and image processing). In dynamic frequency clocking scheme, frequency switching occurs based on the units being used and on single clock which drives all the units. The dynamic clocking unit (DCU) generates different clock frequencies based on instruction words. The block diagram of the DCU is shown in Fig. 1.15. The DCU is a series of cascaded clock divider stages whose inputs are controlled by the pass logic blocks. The output of one clock divider is presented at the input of the next stage when the pass logic is enabled. The pass logic block is controlled by a set of signals generated by the enable encoder. Based on the instruction class, the appropriate pass

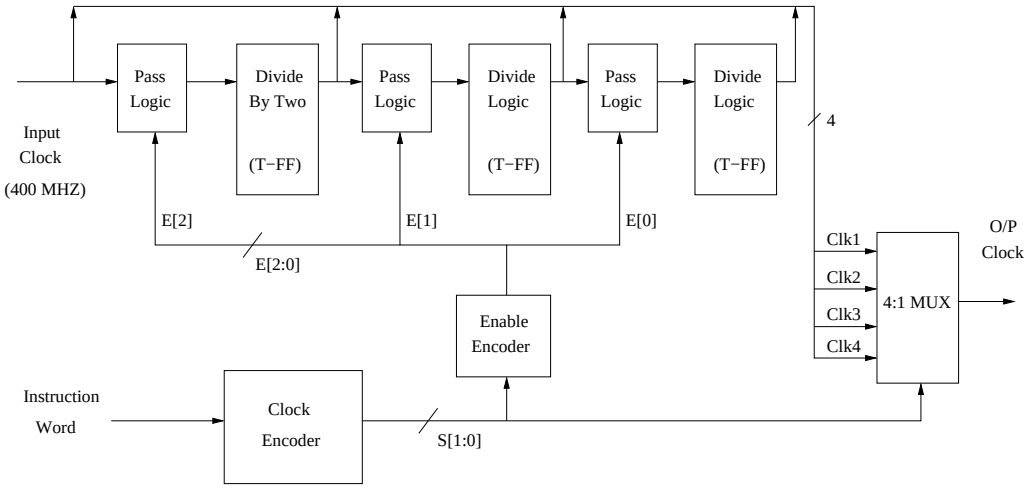
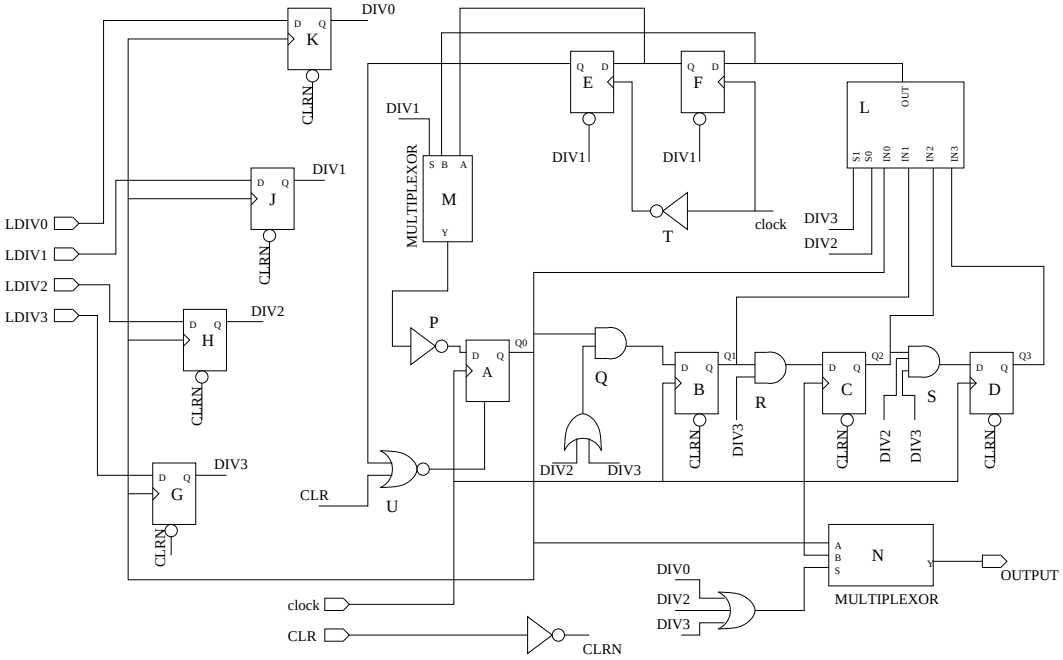


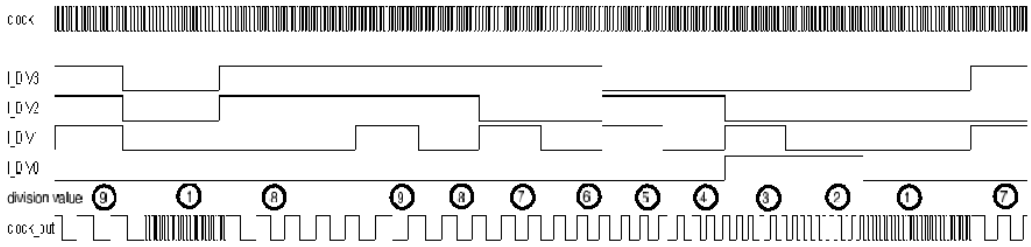
Figure 1.15. Dynamic Clocking Unit : Ranganathan, et. al. [59]

logic blocks are activated by the enable encoder. The master clock is accordingly divided by clock divider circuit to generate the resultant output clock.

Brynjolfson and Zilic [61] propose a dynamic programmable clock divider (DPCD) to use in conjugation with FPGA clock managers. Clock division by ordinary clock dividers can lead to glitches or distortions of the output clock. Distortions at the output clock can result in metastability and latching errors. The DPCD is capable of performing dynamic frequency division without undesired effects at the output. The circuit is shown in Fig. 1.16(a). Division of the input clock is performed by creating a loop of D-flip-flops {A-D} driven by the input clock, and feeding the signal back into the loop thorugh an inverter {D} to create the necessary clock inversion. To expand the length of the output clock, the number of D-flip-flops in the loop is increased by multiplexor {L}. In order to perform an odd division, flip-flops {E, F} extend the loop, by half a period, with an asynchronous clear of flip-flop {A} on the falling edge of the input clock. For the divider output, multiplexer {N} chooses between the original input clock, for a divison of one, and the output of {A}. The output generated by the DPCD is shown in Fig. 1.16(b). To prevent output glitching, D-flip-flops {G,H,J,K} latch the new program value on the rising edge of the output from {A}. Combinational logic {Q,R,S} also help to prevent glitching, but also prevent transient patterns from being captured and fed back, thus causing irregular oscillation in the circuit.



(a) Dynamic Clocking Unit



(b) Output Clock Generated

Figure 1.16. Dynamic Clocking Unit and Output Clock : Byrnjolfson and Zilic [61]

1.9 Fundamentals of Digital Watermarking

Digital watermarking technology is an emerging field in computer science, cryptography, signal processing and communications. Digital Watermarking is intended by its developers as the solution to the need to provide value added protection on top of data encryption and scrambling for content protection. Like other technology under development, digital watermarking raises a number of essential questions as follows.

- What is it?
- How can a digital watermark be inserted or detected?
- How robust does it need to be?
- Why and when are digital watermarks necessary?
- What can watermarks achieve or fail to achieve?
- How should digital watermarks be used?
- How might they be abused?
- How can we evaluate the technology?
- How useful are they, that is, what can they do for content protection in addition to or in conjunction with current copyright laws or the legal and judicial means used to resolve copyright grievances?
- What are the business opportunities?
- What roles can digital watermarking play in the content protection infrastructure ?
- And many more ...

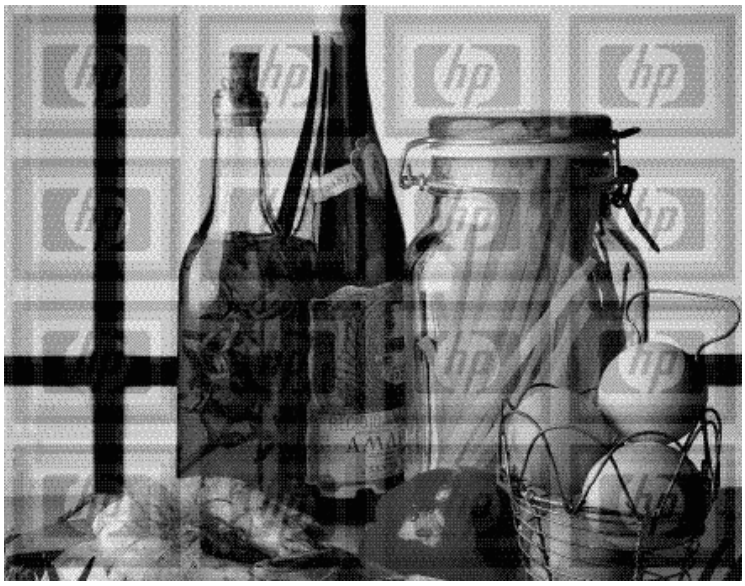


Figure 1.17. Visible Watermarked Image [71]

1.9.1 General Framework for Watermarking

Watermarking is the process that embeds data called a watermark or digital signature or tag or label into a multimedia object such that watermark can be detected or extracted later to make an assertion about the object [70]. The object may be an image or audio or video. A simple example of a digital watermark would be a visible ”seal” placed over an image to identify the copyright, one such example is shown in Fig. 1.17. However, the watermark might contain additional information including the identity of the purchaser of a particular copy of the material.

In general, any watermarking scheme (algorithm) consists of three parts [72].

- The watermark.
- The encoder (insertion algorithm).
- The decoder and comparator (verification or extraction or detection algorithm).

Each owner can use an unique watermark for all objects or an owner can use different watermarks in different objects. The marking algorithm incorporates the watermarks into the object. The verification algorithm authenticates the object determining both the owner and the integrity of the object. A watermark must be detectable or extractable to be useful. Depending on the way the

watermark is inserted and also on the nature of the watermarking algorithm, the method used can involve very distinct approaches. In some watermarking schemes, a watermark can be extracted in its exact form, a procedure we call watermark extraction. In other cases, we can detect only whether a specific given watermarking signal is present in an image, a procedure we call watermark detection. It should be noted that watermark extraction can prove ownership whereas watermark detection can only verify ownership.

Fig. 1.18(a) illustrates the *encoding* process. Let us denote an image by I, a signature by $S = s_1, s_2, ...$ and the watermarked image by $\hat{I}$. E is an encoder function, it takes an image I and a signature S, and it generates a new image which is called watermarked image $\hat{I}$, mathematically,

$$E\left(I, S\right) = \hat{I} \tag{1.7}$$

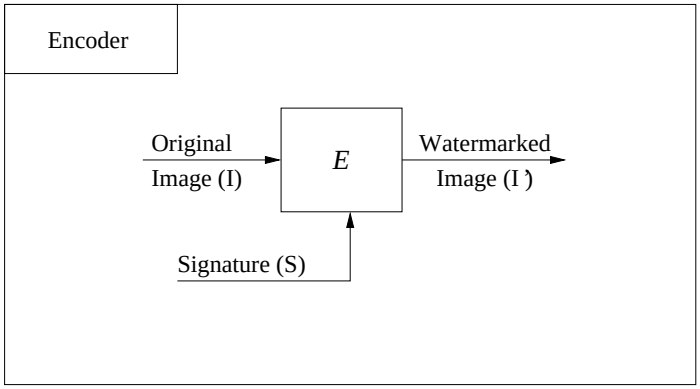
It should be noted that the signature S may be dependent on image I. In such cases, the encoding process described by Eqn. 1.7 still holds.

A *decoder* function D takes an image J (J can be a watermarked or un-watermarked image, and possibly corrupted) whose ownership is to be determined and recovers a signature S' from the image. In this process an additional image I can also be included which is often the original and un-watermarked version of J. This is due to the fact that some encoding schemes may make use of the original images in the watermarking process to provide extra robustness against intentional and unintentional corruption of pixels. The decoding process can be expressed mathematically as,

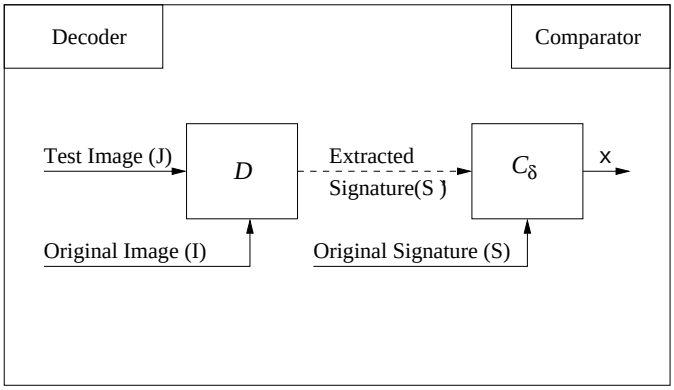
$$D\left(J, I\right) = S' \tag{1.8}$$

The extracted signature S' will then be compared with the owner signature sequence by a *comparator* function C_δ and a binary output decision generated. It is 1 if there is match and 0 otherwise, which can be represented as follows.

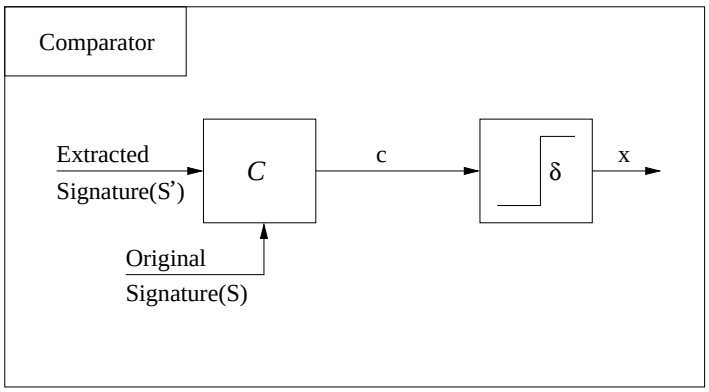
$$C_\delta\left(S', S\right) = \begin{cases} 1, & \mathrm{c} \leq \delta \\ 0, & \text{otherwise} \end{cases} \tag{1.9}$$



(a) Watermarking Encoder



(b) Watermarking Decoder



(c) Watermarking Comparator

Figure 1.18. General Framework of Digital Watermarking

Where C is the correlator, $x = C_\delta(S', S)$. c is the correlation of two signatures and δ is certain threshold. Without loss of generality, watermarking scheme can be treated as a three-tupple (E, D, C_δ). Figs. 1.18(b) and 1.18(c) demonstrate the decoder and the comparator.

1.9.2 Types of Watermarking

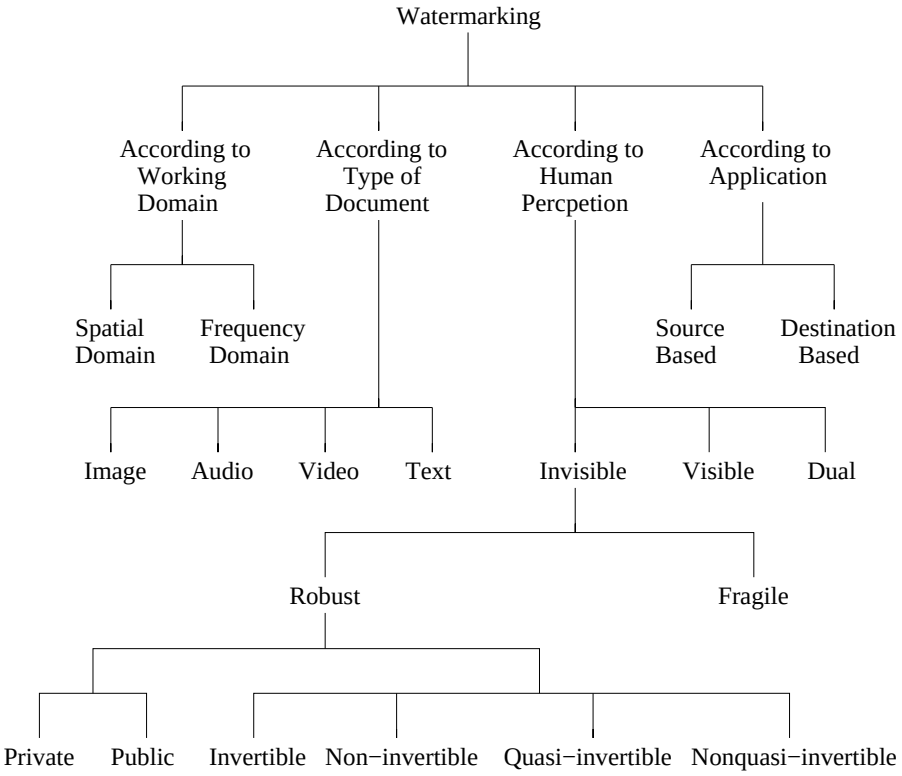
Watermarks and watermarking techniques can be divided into various categories. The watermarks can be applied in *spatial domain* or *frequency domain*. It has been pointed out that the frequency domain methods are more robust than the spatial domain techniques. Different types of watermarks are shown in the Fig. 1.19(a). Watermarking techniques can be divided into four categories according to the type of document to be watermarked as follows.

- Image Watermarking
- Video Watermarking
- Audio Watermarking
- Text Watermarking

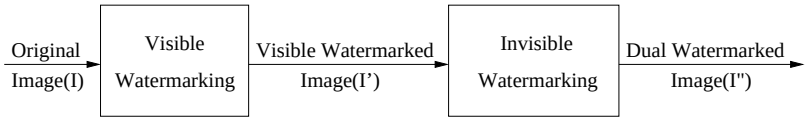
According to the human perception, the digital watermarks can be divide into four different types as follows.

- Visible watermark
- Invisible-Robust watermark
- Invisible-Fragile watermark
- Dual watermark

Visible watermark is a secondary translucent overlaid into the primary image [72, 73, 74, 75, 76, 77]. The watermark appears visible to a casual viewer on a careful inspection. The *invisible-robust* watermark is embed in such a way that alternations made to the pixel value is perceptually not noticed and it can be recovered only with appropriate decoding mechanism [70, 78, 79, 80, 81].



(a) Types of Watermarking



(b) Dual Watermarking

Figure 1.19. Different Types of Watermarks and Watermarking Techniques

The *invisible-fragile* watermark is embedded in such a way that any manipulation or modification of the image would alter or destroy the watermark [82, 83, 84]. Dual watermark is a combination of a visible and an invisible watermark [83]. In this type of watermark an invisible watermark is used as a back up for the visible watermark as clear from the following diagram (Fig. 1.19(b)).

An invisible robust *private* watermarking scheme requires the original or reference image for watermark detection; whereas the *public* watermarks do not. The class of invisible robust watermarking schemes that can be attacked by creating a "counterfeit original" (to be discussed in later sections) is called *invertible* watermarking scheme. Using mathematical notations from Section 1.9.1, an invisible robust watermarking scheme (E, D, C_δ) is called *invertible* if, for any watermarked image $\hat{I}$, there exits a function E^{-1} such that (1) $E^{-1}(\hat{I}) = (I', S')$, (2) $E(I', S') = (\hat{I})$ and (3) $C_\delta(D(\hat{I}), S') = 1$, where E^{-1} is a computationally feasible function, S' belongs to the set of allowable watermarks, and the images I and I' are perceptually similar. Otherwise, the watermarking scheme is *non-invertible*.

A watermarking scheme (E, D, C_δ) is called *quasi-invertible* if, for any watermarked image $\hat{I}$, there exits a function E^{-1} such that (1) $E^{-1}(\hat{I}) = (I', S')$, (2) $C_\delta(D(\hat{I}), S') = 1$, where E^{-1} is a computationally feasible function, S' belongs to the set of allowable watermarks, and the images I and I' are perceptually similar. Otherwise, the watermarking scheme is *nonquasi-invertible*.

From application point of view, digital watermark could be either source based or destination based. *Source-based* watermark are desirable for ownership identification or authentication where a unique watermark identifying the owner is introduced to all the copies of a particular image being distributed. A source-based watermark could be used for authentication and to determine whether a received image or other electronic data has been tampered with. The watermark could also be *destination-based* where each distributed copy gets a unique watermark identifying the particular buyer. The destination -based watermark could be used to trace the buyer in the case of illegal reselling.

The research in digital watermarking is well matured. The software implementation of the proposed algorithms are significantly large, whereas the hardware implementation of the algorithms is lacking. The hardware implementation has advantages over the software implementation in terms

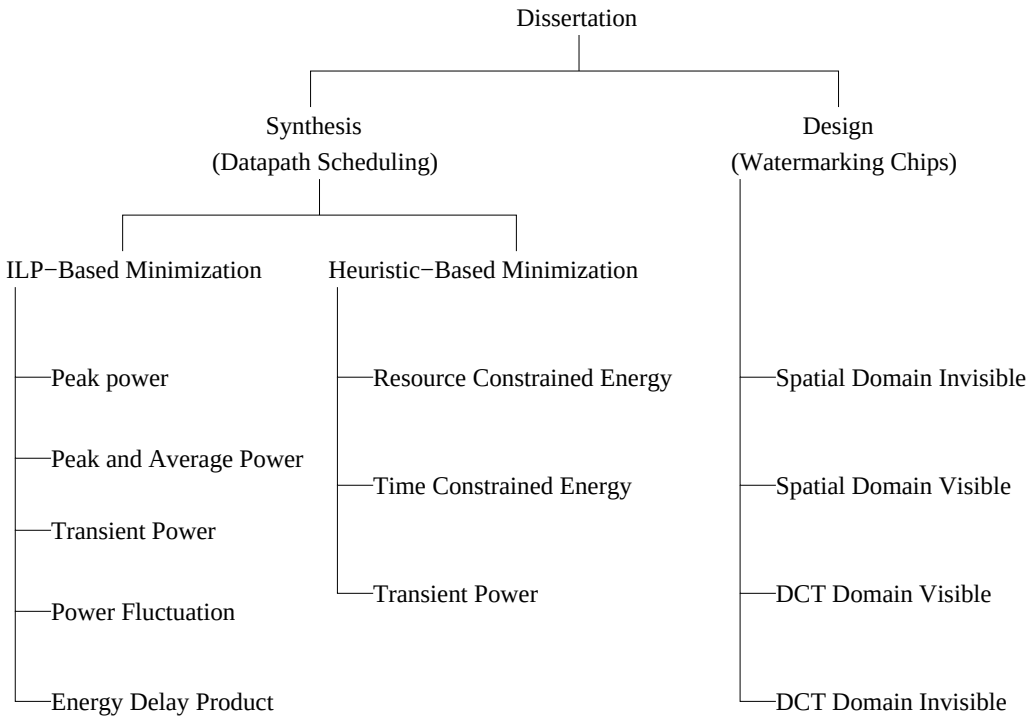


Figure 1.20. Contributions of this Dissertation

of low power, high performance and reliability. In this dissertation, we develop hardware system that can insert invisible-robust, invisible-fragile, visible spatial domain as well as DCT domain watermark in the image. The hardware module can be easily incorporated in JPEG encoder to develop a secure JPEG encoder. It may be noted that the corresponding watermark extraction module has to be inbuilt in a secure JPEG decoder. The secure JPEG codec can be a part of a scanner or a digital camera so that the digitized images are watermarked right at the origin.

1.10 Contributions of this Dissertation

The contributions of this dissertation are in two broad categories, such as scheduling algorithms for low power behavioral synthesis and the design of application specific integrated circuits for digital watermarking. Fig. 1.20 outlines the contributions of this dissertation in detail.

During low power synthesis at behavioral level, several low power subtasks, such as, scheduling, allocation and binding are performed. In this dissertation, scheduling schemes are proposed to reduce peak power, average power, peak power diffential, power fluctuation and energy at be-

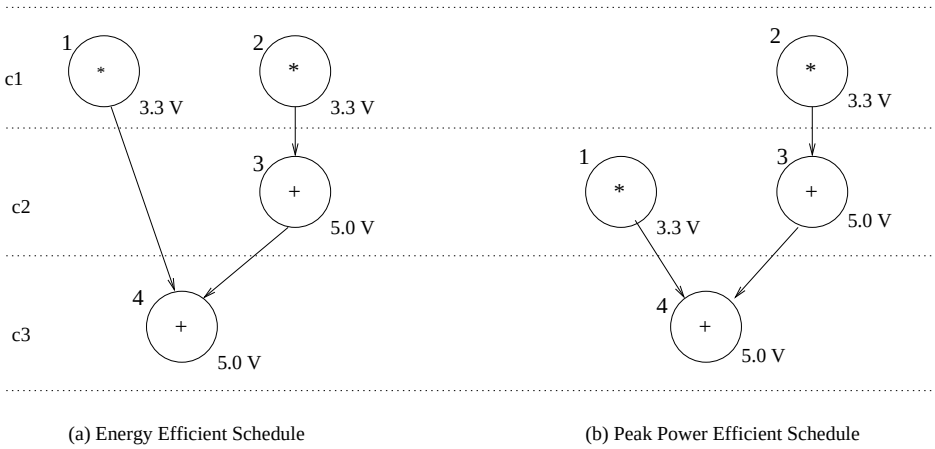


Figure 1.21. Energy Vs Peak Power Efficient Schedule

havioral level using integer linear programming(ILP) models and also using heuristics based algorithms. First, different power models are developed to capture the power characteristics of a datapath circuit. Then, datapath scheduling schemes are developed using multiple supply voltages and dynamic frequency clocking (MVDFC), multiple supply voltages and multicycling(MVMC). Both these schemes are compared with single voltage and single frequency(SVSF) scheme.

To have a clear understanding of the scheduling for energy and peak power minimization, let us refer to data flow graph(DFG) in Fig. 1.21. The figure shows two different possible schedules of the same DFG using multiple supply voltage scheme. Since, in both cases there are two multipliers operating at $3.3V$ and two adders operating at $5.0V$, the energy and average power consumption of both scheduled DFGs is the same. However, the peak power consumption in Fig. 1.21(b) is less than that in Fig. 1.21(a). The approach in this thesis is to generate peak power efficient schedules similar to the one in Fig. 1.21(b).

A class of VLSI architecture are proposed for digital image watermarking implementing a set of watermarking algorithms. Several CMOS VLSI circuits are designed and implemented as prototype circuit design, which can be icorporated in a JPEG encoder or a digital still camera. The VLSI implementation of spatial domain watermarking architectures using 0.35μ CMOS technology is given. To our knowledge, this is the first watermarking chip implementing invisible-robust, invisible-fragile and visible watermarks together. Also, to our knowledge, this is the first water-

marking chip having spatial visible watermarking capability. In this dissertation, we also propose the architecture for DCT domain invisible and visible watermarking algorithms. The prototype implementation of DCT domain invisible and visible watermarking architecture using 0.25μ CMOS technology is given in [85].

1.11 Dissertation Outline

The remainder of the dissertation is organized as follows: Chapter 2 describes the related work in the areas of low power high-level synthesis, variable clocking based systems and the hardware based watermarking schemes. The fundamental concepts of multiple suppy voltages, dynamic frequency clocking and multicycling is introduced in Chapter 1.8. This also describes how energy / power reduction is obtained by use of dynamic frequency clocking and multiple supply voltages in a VLSI circuit. In Chapter 3, heuristic based resource and time constrained algorithms are developed for energy efficient datapath scheduling. Chapter 4 discusses the datapath scheduling scheme for synthesis of energy efficient high performance datapath achieved through energy delay product (EDP) minimization. In Chapter 5, the simultaneous reduction of both peak and average power is discussed. This will also include a section on peak power minimization. A heuristic based framework is given in Chapter 6 for simultaneous minimization of various power parameters. Chapter 7 elaborates transient power minimization through datapath scheduling using ILP-Based models. In this case the cycle difference power is modeled as absolute deviation from mean cycle power (an estimate of average power). The power fluctuation of a datapath circuit is characterised as cycle-to-cycle power gradient in Chapter 8. To achieve the reduction in power fluctuation of a datapath circuit, ILP-based scheduling schemes are developed that minimizes mean power gradient (MPG). VLSI designs for digital watermarking of images are proposed in Chapter 9. This includes three designs, one for invisible spatial domain watermarking, one for visible spatial domain watermarking followed by a DCT domain visible and invisible watermarking chip. Conclusions and future directions of research are discussed in Chapter 10.

CHAPTER 2

RELATED WORK

The energy consumption of a CMOS circuit is dependent on the supply voltage and the effective switching capacitance. Several datapath scheduling algorithms have been proposed in the literature optimizing either one or both of the above parameters for energy reduction. Moreover, variable frequency or multiple frequency operations are also considered as options for power reduction. In this chapter, the various related works are classified as, methods based on voltage reduction, and those based on switching activity reduction. A few research works are based on using multiple, dynamic or variable frequency for synthesis of low power or high performance systems can be found in the literature. This chapter briefly outline these works and further discuss, hardware designs for digital watermarking.

In this chapter, a brief overview of existing literature on energy and power reduction in VLSI circuits is presented. Section 2.1 presents existing works in the low power datapath scheduling methods for energy or average power reduction using lower supply voltages. The high-level synthesis works that achieve energy or average power minimization by reducing the load capacitance or switching activity in a circuit are presented in Section 2.2. Section 2.3 presents a brief overview of literature on datapath scheduling methods for peak power and transient power reduction in a circuit. The scheduling schemes for variable voltage processor core based systems are presented in Section 2.4. In the past frequency scaling or variable latency concepts have been used for the development of either low power or high-performance systems. Section 2.5 reviews such research works proposed in the literature. The design works based on multiple supply voltages are also included in Section 2.5. The hardware based watermarking systems are discussed in Section 2.6.

2.1 Datapath Scheduling for Energy or Average Power Reduction using Voltage Reduction

It is known that voltage reduction is one of the effective methods of power reduction since the power or energy consumption is quadratically dependent on the supply voltage. In this section, we review the works poposed from the literature using multiple supply voltages during datapath scheduling for minimization of energy or average power.

Johnson and Roy [86, 87] present a method called *Minimum Energy Schedule with Voltage Selection* (MESVS) based on Integer Linear Programming(ILP) to optimize the schedule, supply voltage levels, and allocation of resources. The MESVS algorithm takes a directed acyclic data flow graph, the allowable set of supply voltages, a limit on the number of supply voltages that can be selected, a minimum difference between the voltages that can be selected, average switching activity values for each datapath operation, nominal propagation delay and average energy dissipation values for each datapath resource as inputs. The objective function for MESVS is an estimate of datapath energy dissipation expressed as a function of supply voltages. The outputs of the MESVS algorithm are the following : (i) a datapath schedule (represented by scheduled data flow graph), (ii) an energy estimate, (iii) selection of optimal set of supply voltages, (iv) assignment of supply voltage to each operation and (v) allocation of resources to each supply voltage. Since the different resources need to operate at different voltages level conversion is needed. There are four possible schemes, such as omitting the level converter, using a chain of inverters, using an active or passive pullup and using dual cascade voltage switch (DCVS) circuit. The authors claim that energy savings in the range of $46\% - 58\%$ is obtained compared to $5V$ operation. The other observation was that the use of two supply voltages can reduce power dissipation substantially, while three supply voltages resulted in less than 5% reduction compared to two supply voltages.

Johnson and Roy [65] present an algorithm called *Multiple Operating Voltage Energy Reduction*(MOVER) to minimize datapath energy dissipation. Energy savings ranging from $0 - 50\%$ are obtained with the area penalty in the range $0 - 170\%$. The MOVER generates one, two, and three supply voltage designs for consideration by the circuit designer. The user has control over latency constraints, resource constraints, the number of control steps, clock period, and the number of power supplies. The MOVER iteratively searches for the range of minimum voltage levels. The

MOVER uses an ILP to evaluate the feasibility of candidate supply voltage selections, to partition operations among different power supplies and to produce a minimum area schedule under latency constraints once voltages have been selected. The MOVER has the following phases :

- determining maximum and minimum bounds on the time frame in which each operation must execute
- searching for minimum voltage
- partitioning datapath operations into two supply voltage that are either higher or lower supply voltages.
- partitioning the lower voltage group, for the three supply voltage schedule.

The MOVER algorithm [65] is similar to the MESVS algorithm [87] in the following ways :

- both use ILP formulation
- behavior with respect to latency, resource, ad supply voltage constraints
- both use *differential cascode voltage switch*(DCVS).

The difference between the MOVER and MESVS two is that MESVS can only select a discrete set of voltages, whereas MOVER can select a continuous range of voltages. The ILP formulation handles timing and resource constraints and accounts for the cost if level shifters are used. However, MOVER and MESVS have following drawbacks :

- it does not address conditional branches
- does not consider functional pipelining
- energy model used is data-intensive which ignores the effect of input activities on the energy dissipation of a module
- it has exponential worst-case complexity and can not handle large benchmarks.

Chang and Pedram [51, 88] present a dynamic programming technique for multiple supply voltage scheduling. The proposed technique handles both functionally pipelined and non-pipelined datapaths and multicycling operations. The scheduling algorithm assigns a supply voltage level from a fixed set of voltage levels such that the energy consumption is minimum for given constraints. In this algorithm, the level-shifters are used for both step-up and step-down of signals. It may be noted that in most of the algorithms, level-shifters are used for step-up of signals only. An average saving of 40.19% is obtained using three supply voltage levels as compared with single supply voltage level. The algorithm has pseudo-polynomial complexity and produces optimal results for trees and produces suboptimal for general directed acyclic graphs. The scheduling algorithm can handle very large data flow graphs and the results are within 1% error.

In [89], an ILP formulation and a heuristic for variable voltage scheduling is presented by Lin, Hwang and Wu. The authors have considered three different solutions to the problem, such as time constrained, resource constrained, and time-and-resource constrained. The scheduling schemes consider variable supply voltage and multicycling. The heuristic method produces results comparable with those of the ILP method in a fraction of run-time. The time complexity of the heuristic algorithm is $O\left(n^3 logn\right)$. The proposed heuristic is an modification over list-based algorithm with a priority function that considers three factor, such as the power gain of an operation, the mobility of an operation, and the computation density. The authors show that using different cost and delay combinations, power consumption in a single design can differ by as much as a factor of 6 when using mixed $(3.3V$ and $5.0V)$ supply voltages.

Sarrafzadeh and Raje [90] proposed two scheduling algorithms; one is a dynamic programming algorithm and other is an heuristic algorithm based on geometric algorithm. The algorithms assume both time and resource constraints as inputs. The resource constraints is the number and type of each functional units and their operating supply voltage. The algorithms assume only two supply voltages, such as $3.3V$ and $5.0V$. The aim of the algorithms is to maximize the usage of the functional units at the lower supply voltages while satisfying the time constraints. Let n be the number of nodes, k be the time constraint, R is given resource constraint, β is latency of a functional unit that run at a supply voltage of V_β. The running time of the dynamic programming

Table 2.1. Datapath Scheduling Schemes using Multiple Supply Voltages

Proposed Scheme	Optimization Method Used	Constraints Assumed	Operating Voltage Levels	Time Complexity
Johnson and Roy [86, 87]	ILP	Time	$(5.0V \rightarrow 2.0V)$	Expoential
Johnson and Roy [65]	ILP	Time	$(5.0V, 3.3V, 2.4V)$	Expoential
Chang and Pedram [51, 88]	Dynamic Programming	Time	$(5.0V, 3.3V, 2.4V)$	Pseudo-Polynomial
Lin, Hwang and Wu [89]	ILP and Heuristic	Time and Resource	$(5.0V, 3.3V)$	Expoential $O\left(n^3 logn\right)$
Sarrafzadeh and Raje [90]	Dynamic Prog Geometric	Time and Resource	$(5.0V, 3.3V)$	$O\left(n^2 k\beta \lvert R\rvert^2\right)$ $O\left(nC lognC\right)$
Kumar and Bayoumi [91, 92, 93]	Stochastic Evolution	Resource	$(5.0V, 3.3V, 2.4V)$	$O\left(n^2\right)$
Elgamel and Bayoumi [94]	Genetic Algorithms	Time and Area	$(5.0V, 3.3V, 2.4V)$	NA
Shiue and Chakrabarti [95, 96]	List-Based	Time and Resource	$(5.0V, 3.3V)$ or $(5.0V, 3.3V, 2.4V)$	Polynomial
Manzak and Chakrabarti [97]	Lagrangian Multiplier	Time and Resource	$(5.0V, 3.3V,$ $2.4V, 1.5V)$	$O\left(n^2\right)$ and $O\left(n^2 logL\right)$
Manzak and Chakrabarti [98]	List-Based	Time and Resource	$(5.0V, 3.3V,$ $2.4V, 1.5V)$	$O\left(r^2 L^2\right)$

scheduling algorithm is $O\left(n^2 k\beta|R|^2\right)$. If C is the number of control steps, then the time complexity of the geometric algorithm is $O\left(nC lognC\right)$ and can handle more than two supply voltages. The authors reported power reductions in the range of $13.28 - 31.54\%$ for various high-level synthesis benchmarks under various resource and time constraints.

Kumar and Bayoumi [91, 92, 93] proposed scheduling schemes using multiple supply voltages and multicycling. The algorithms essentially has two phases, initial-scheduling and re-scheduling. During initial scheduling parallelism is exploited and the re-scheduling uses an iterative approach, which is based on stochastic evolution. Level-converters are used when a functional unit operating at lower voltage drives a functional unit operating at higher voltage. The time-complexity of the scheduling algorithm is $O\left(n^2\right)$. The authors report power savings upto 80% for three supply

voltage levels of $(5.0V, 3.3V$ and $2.4V)$. The power overhead due to the level-converters is in the range $0-4\%$ and the area overhead is in the range $0-6\%$.

Elgamel and Bayoumi [94] use genetic algorithms to solve multiple supply voltage scheduling problem with multicycling operations. The proposed scheme assumes unscheduled data or control flow graph, datapath component library, area and time constraints as inputs and minimize average power. The algorithms simultaneously solves scheduling, allocation and binding. Power reduction as high as 84% is reported. The results do not consider the power overhead due to the level converters.

Shiue and Chakrabarti [95, 96] discuss a resource constrained and a latency constrained list-based scheduling algorithms using multiple supply voltages. The scheduling scheme consider the effect of switching activity. The algorithms use heuristics to reduce power consumptions in the level-converters. The list based algorithms assign control steps to nodes based on their priorities. The priority of a node is a function of various parameters, such as depth, mobility, switched capacitance, interconnection complexity and need for a level shifter. The level shifters are used between a low-voltage resource and a high-voltage resource for stepping-up the signal. The proposed algorithms are of polynomial time-complexity. The proposed schemes achieve significant power reduction when the operation voltages are $(5.0V$ and $3.3V)$ or $(5V, 3.3V,$ and $2.4V)$.

The Lagrangian multiplier method has been used by Manzak and Chakrabarti [97] to develop resource and time constrained scheduling algorithms. The algorithms which use Lagrangian multiplier method in an iterative fashion, are based on efficient distribution of slack among the nodes in the DFG. If n denotes the number of nodes and L denotes the latency, the time complexity of the two versions of the proposed algorithms are $O\left(n^2\right)$ and $O\left(n^2 logL\right)$. The $O\left(n^2 logL\right)$ algorithm results better savings in energy compared to the $O\left(n^2\right)$ algorithm. Average power or energy reduction of 39% has been obtained when the latency constraint is 1.5 times the critical delay and is improved to 58.5% when the latency constraints relxed to 2 times the critical path delay. The time constraint, resource constraint consisting of the number of resource of each type operating at specific voltage, delay and energy values are given as inputs to the algorithm. The resources are

allowed to operate at one of supply voltages from $(5.0V, 3.3V, 2.4V,$ and $1.5V)$. The level shifters are used whenever step-up of signal is necessary.

Manzak and Chakrabarti [98] proposed list-based latency and resource constrained scheduling algorithms. The scheduling uses priority function based on the number of available resources, the difference between the actual number of cycles left and estimated number of cycles required to schedule remaining nodes. The algorithms consider the switching activity of nodes. The resources are allowed to operate at one of supply voltages from $(5.0V, 3.3V, 2.4V,$ and $1.5V)$. The average power or energy reduction is 59.1% when the latency constraint is 1.5 times the critical delay and the average power or energy reduction is 66.8% when the latency constraint is 2.0 times the critical delay. The time-complexity of the algorithm is $O\left(r^2L^2\right)$, where r is the number of resources, and L is the latency.

A comparative view of the above discussed algorithms which use voltage reduction for average power or energy reduction is given in Table 2.1.

2.2 Switching Activity Reduction During High-Level Synthesis

In this section, we discuss the works on datapath scheduling which use capacitance reduction to reduce average power or energy. An overview of the discussed methods is given in Table 2.2, where the percentage power reduction is the average data.

Kumar, Katkoori, Rader and Vemuri [99, 100] present a profile driven approach to high-level synthesis called as *Profile Driven Synthesis System*(PDSS). The inputs to the PDSS are a subset of VHDL and constraints in terms of clock period and area. The PDSS generates a constraint-satisfying design with the least amount of estimated switching activity. In this system, the input specification is profiled to collect data for various operations and carriers using a user-specified input set of vectors. The switching activity for each module set is estimated by using this profiled data and the raw switching activity data of all modules in the library. The module set with minimum estimate of power consumption is chosen for further synthesized. The goal of profiling is to gather the following data :

- For each node (operation), the number of times the node is executed for a given profiling stimuli is determined and input vectors used as profile stimuli. This number is called the event activity of the operation node.

- For each edge, the number of times the edge is traversed during execution is determined. This number is called the transaction activity of the edge.

- For each edge, the number of times the value on the edge has changed is determined. This number is called the event activity of the edge.

The authors claim that the results obtained are within an accuracy of 10% of the actual switching activity measured at the switch level implementation of the design.

Raghunathan and Jha [101] present a comprehensive low-power datapath synthesis system that performs the various high-level synthesis tasks with the aim of reducing power consumption in the synthesized datapath. The authors call the system as SCALP. The system considers both supply voltage and switching capacitance to reduce the power consumption. The authors claim that SCALP estimates switching capacitance accurately, handles diverse module libraries and utilizes complex scheduling constructs such as multicycling, chaining, and structural pipelining. The input to the SCALP is a control data flow graph (CDFG), input sampling period, and a library of components to be used for datapath implementation. The SCALP minimizes power consumption both by voltage scaling and switching capacitance reduction. This is done by first pruning the set of candidate supply voltages to a small set of supply voltages. For each supply voltage in the pruned set, a datapath is synthesized that has minimal capacitance. The best solution among these datapaths in terms of power consumption is then chosen.

Raghunathan and Jha [102] are the first researchers to purpose the allocation method for low power. The method is based on iterative improvement of some initial solution. The authors assume random input in a structurally pipelined design. The method can also handle non-random input sequences. The method is implemented in the framework of *Genesis* behavioral synthesis system[103]. In this system, register and module allocations are performed simultaneously, while minimizing the amount of interconnect needed. A lifetime analysis is performed for the scheduled

CDFG. Two variables are said to be compatible and can share hardware resources if they are not alive at the same time. Similarly, two operations are compatible if they are not performed at the same time. Allocation is based on a weighted graph called compatibility graph (CG). Initially, each variable and operation corresponds to a node in the CG, with undirected edges connecting compatible pairs. Weights are assigned to edges in the CG to indicate the preference on the two variables or operations for sharing the same resource. A single step of allocation selects the edge in the CG with the highest composite weight, and merges the two nodes it joins, maps the corresponding variable (or operation) to the same module (register). If two or more edges have the same composite weight, the tie is broken based on the corresponding transition activity weights (or some cases arbitrarily). Power reduction is achieved by the help of two factors, capacitance and transition activity. Capacitance is reduced by minimizing the number of functional modules, registers and multiplexers. The allocation scheme selects a sequence of operations (variables) for a module or register such that the transition activity is reduced.

Chiou, Muhammand and Roy [104] propose scheduling and allocation method that reduce power consumption of data intensive applications by minimizing switching activity. The main idea of the synthesis technique is to reduce the signal strength difference among the inputs of shared resources. The signal strength is derived from word-level statistics. The authors have proposed a formula that relates switching power with resource sharing as follows.

$$\text{Switching increment} = \frac{\text{Difference in switching activity with and without sharing}}{\text{Switching activity without sharing}} \quad (2.1)$$

It is observed that sharing resources between two operations with high signal similarity will lower switching activity and hence reduce switching power. This observation serves as the major principle behind the proposed scheduling and allocation techniques. The proposed scheduling algorithm is heuristic based and uses greedy approach in making module selections. Average power reduction upto 49% is obtained using the proposed techniques compared to the conventional ones.

A comprehensive high-level synthesis system is proposed by Khouri, Lakshminarayana and Jha [105] to synthesize both control-flow intensive and data-intensive circuits. The system handles

conventional synthesis tasks, such as scheduling, module selection, and resource sharing. Moreover, power-conscious structuring of multiplexer networks, which are predominant in control-flow intensive circuits, is the key additional feature in the system. Experimental results demonstrate power reduction of 62% for control-flow intensive benchmarks as compared to V_{dd}-scaled area-optimized designs. The power reduction for the data-dominated benchmarks is 58% as compared to V_{dd}-scaled (delay-optimized) designs. The power reductions come with an area penalty of approximately 40%.

Henning and Chakrabarti [106, 107] propose an intutive switching activity model to capture data characteristics in terms of statistical parameters. Then, heuristics are proposed for scheduling and allocation exploration. The novelity of the model is a relation between switching activity of datapath interconnect to the fixed-point, two's complement data. The model is based on four practical parameters, which are basically the bits of the two values involved in the transition, such as sign bits, the number of intersecting sign bits, number of truncation bits in the two values and all other bits of a value that are not sign or truncation bits. Since, the model is dependent on only four parameters the scheduling and allocation is efficient. The heurstic is applied to synthesize a speech codec design. It is reported that average power reduction is about 15% during encoding.

An ILP-based resource binding scheme is proposed Shiue and Chakrabarti [108] that minimizes the amount of switching at the inputs of functional units. The idea of resource binding is to find n disjoint paths from a multistage graph with m stages, where m is the number of cycles in the schedule and n is the number of nodes per stage. The first step of binding is to find a multistage graph called the binding graph. The total number of nodes of such graph is $n \times m$, and two nodes for source and sink. If two nodes are located in two different stages can share a resouce, then the two are connected with a edge. Each edge is labeled with a cost corresponding to the switching activity. The LP objective is to find n disjoint paths such that the total cost of these paths is minimum. Power savings in the range of $8.2 - 34.4\%$ are obtained using the proposed binding scheme for various resource constraints as compared to random binding scheme.

Musoll and Cortadella [38] present algorithms for scheduling and resource-binding to reduce power consumption during behavioral synthesis. The algorithms reduce power consumptions by

Table 2.2. High-Level Synthesis Schemes using Switching Activity Reduction

Proposed Work	Synthesis Tasks Performed	Methods Used	Time Complexity	% Power Reduction
Kumar, Katkoori, Rader and Vemuri [99, 100]	Scheduling, Register Optimization, etc.	Simulation of DFG	NA	NA
Raghunathan and Jha [101]	Tranformation, Scheduling and Allocation	Iterative Improvement	Polynomial	4.6
Raghunathan and Jha [102]	Allocation	Simulation	NA	14.6
Chiou, Muhammand and Roy [104]	Scheduling and Allocation	Heuristic Based	Polynomial	30.13
Khouri, Lakshminarayana and Jha [105]	Scheduling and Resource Sharing	Heuristic	Polynomial	22
Henning and Chakrabarti [106, 107]	Scheduling and Allocation	Intutive Heuristic	Polynomial	15
Shiue and Chakrabarti [108]	Resource Binding	Integer Linear Programming	Exponential	24.08
Musoll and Cortadella [38]	Scheduling and Resource Binding	List-Based Algorithm	$O\left(n^2m\right)$	6.67
Lundberg, Muhammad, Roy and Wilson [109, 110]	NA	Hierarchical	NA	14.93
Shin and Lin [111]	Resource Allocation	Heuristic	Polynomial	7.84
Monteiro, Devadas, Ashar and Mauskar [113]	Scheduling	HYPER [112]	NA	22.43
Cherabuddi, Bayoumi [114]	Partitioning and Binding	Stochastic Evolution	Polynomial	23.89
Lee, Lee, Park and Hwang [115]	Scheduling	Heuristic	Polynomial	16.5
Gupta and Katkoori [116]	Scheduling	Force-Directed Heuristic	$O\left(n^4t\right)$	16.4
Murugavel and Ranganathan [117]	Scheduling Binding	Game Theory	Exponential	13.9

reducing the transitions of their input operands. The power consumption of a functional unit is divided into *useful* and *useless* power. Useful power is consumed when an operation is executed and useless power is the consumption due to an input transition while the functional unit is idle. The algorithms proposed reduces both useful and useless power consumption. The scheduling algorithm is list-based in which the operation priority is set in such a way that operations sharing the same operand are scheduled in control steps as close as possible. For n number of operations and m number of functional units, the running time of the proposed low power list scheduling (LPLS) is $O\left(n^2m\right)$. The algorithm for resource-binding is based on clique partition that reduces power consumption by taking the average Hamming distance (AHD) among the variables. For two operands p and q, if $H(p,q)$ is the Hamming distance and x_i is the value of operand x in cycle i, the average Hamming distance is defined as follows.

$$AHD(x) \quad = \quad \lim_{n\to\infty}\left(\frac{\sum_{i=1}^{n} H(x_i,x_{i-1})}{n}\right) \tag{2.2}$$

The average Hamming distance is used as a measure of energy in nJ/operation. Power reductions in the range of $5-8\%$ have been reported.

Lundberg, Muhammad, Roy and Wilson [109, 110] proposed switching activity models and use them to synthesize low power digital signal processing systems. The models can be easily integrated in any CAD tool. The accuracy of estimates obtained using the proposed models is reported to be within 4%. Switching activity reductions upto 20% is obtained using the proposed approach. The models consider switching occuring at the output of functional units, but do not consider the capacitance difference due to the interconnect lengths. The bits of a signal are divided into three regions, such as low switching region, high switching region and the region in between. The low switching region consists of the most significant bits (MSBs), the high switching region is the least significant bits (LSBs) and the inbetween region is considered to be a linear transition connecting the other two regions. Using these models, the output switching of basic building blocks, such as one-bit delay, half-adder, full-adder have estimated. It is assumed that the number

of internal transitions of a half-adder and a full adder is twice and thrice, respectively more than that of an one-bit delay.

Shin and Lin [111] propose an efficient resource allocation algorithm that minimizes switching activity to reduce the dynamic power consumption of the DSP datapath. Let X be a certain binary input sequence. Suppose, L is the length of X and S is the number of "1"s in the input sequence X. The average switching activity of X is calculated as follows.

$$\alpha_{switching} = 2\left(\frac{S}{L}\right)\left(\frac{L-S}{L}\right) \tag{2.3}$$

For example, for a input sequence 0110110000, $L = 10$ and $S = 4$. The input to the allocation algorithm is a scheduled data flow graph. The algorithm executes all control steps, and compare functional unit with low power consuming register and interconnects of DSP circuits. The algorithm is of polynomial time complexity. Power reduction upto 8.5% reported using the algorithm.

Shut-down techniques are used by Monteiro, Devadas, Ashar and Mauskar [113] to eliminate switching activity and hence power dissipation. The conditions under which the output of a module is not used for a particular cycle is identified and the input latches for that module is disabled when the conditions are met. The proposed scheduling algorithm maximizes the shut-down period of functional units. The scheduling algorithm is time and resource constrained. The techniques, such as multiplexor reordering, pipelining are proposed to improve power management under these stringent contraints. The power reduction as high as 41.67% has been reported.

Cherabuddi and Bayoumi [114] propose partitioning and binding algorithms that minimize the switching activity of functional units and global buses for single-chip applications. Cherabuddi, Bayoumi and Krishnamurthy [118] extend the same work for multi-chip applications. The authors have used a stochastic evolution based technique for partitioning. Power reduction up to 60% has been reported. The switching activity is computed by iteratively changing the input data pattern and a switching activity matrix is constructed. The partition algorithms partition the data flow graph such that each one of them can be implemented in different chips of multi-chip modules (MCMs). The stochastic evolution approach is used in the partition algorithm for faster conver-

gence. Scheduling and binding steps are performed for each move on the partitioning. An incompatible graph is constructed from the original graph for resource allocation purpose. To find optimal solutions for low-power binding, a multistage graph is formulated and dynamic programming approach is used. The total switching activity of a schedule is calculated as the summation of switching activity of the chips on the module and the switching activities on the interchip buses.

Lee, Lee, Park and Hwang [115] propose a scheduling algorithm that reduces the switching activity of the functional units under area or time constraints and thus reducing the power consumption. The switching activity is minimized by scheduling operations such that the Hamming distance between the variables appearing in the input and output port is minimum. The functional unit allocation is performed by partitioning the operations in the given behavioral description and the switching activity is kept at minimum. After allocation is performed, the scheduling algorithm attempts to schedule the operations using the minimum number of functional modules. The algorithm is of polynomial time complexity. The results indicate that switching reduction of 16.5% in average can be obtained.

Gupta and Katkoori [116] propose a scheduling algorithm based on the original force-directed scheduling algorithm proposed in [24]. For a given data flow graph and input data environment the DFG is profiled with the representative data streams. The probability of selecting a combinations among the operations which would share a resource is evluated. Assuming that the force equation is $F = kx$, the switching capacitance inside a module is modeled as spring constant k and the probability of selecting such an combination is modeled as displacement x. For t number of possible time steps and n number of operations, the time complexity of the proposed algorithm is $O\left(n^4 t\right)$. It may be noted that the original force-directed scheduling algorithm has running time of $O\left(n^2\right)$. The authors have reported a power reduction of 16.4% over the conventional force-directed algorithm.

Murugavel and Ranganathan [117] describe a game theory based algorithm for average power minimization during behavioral synthesis using low power binding. The techniques of functional unit sharing, path balancing, and register assignment are incorporated within the binding algorithm for power reduction. For the binding algorithm, each functional unit in the datapath is modeled as

Table 2.3. Relative Performance of Various Schemes Proposed for Peak Power Minimization

Proposed Work	Synthesis Tasks Performed	Methods Used	Time Complexity	% Power Reduction
Martin and Knight [41, 44]	Scheduling Assignment	Genetic Algorithms	NA	40.3-60.0
Shiue and et. al. [119, 120, 121, 108]	Scheduling	ILP Force Directed	Exponential $O\left(cn^3\right)$	$50.0 - 75.0$
Raghunathan, and et. al. [47]	Scheduling	Data Monitor Operations	NA	17.42-32.46

a player bidding for executing an operation with the estimated power consumption as the bid. The operations are assigned to the functional units such that the number of inputs to the functional units that change is minimized thus reducing switching activity. The proposed algorithm yields power reduction improvement of 13.9% without any increase in area or delay overhead.

2.3 Datapath Scheduling for Peak Power Reduction

Few research works have appeared addressing peak power minimization at behavioral level. In this section, we briefly discuss those works and give a overview of their relative performance in Table 2.3.

Martin and Knight [44, 41] have proposed a scheme which combines the SPICE simulations with a behavioral synthesis tool to estimate and optimize digital ASIC's peak power consumption. SPICE is used to measure the power consumption accurately. The behavioral synthesis tool is used for simultaneous assignment and scheduling such that the use of power in each clock cycle is minimum. Genetic algorithms are used in the behavioral synthesis tool for optimization. The author claim that genetic algorithms have advantages over the other conventional optimization tools since they never get stuck in local minima and do not need fine tuning. The proposed synthesis tool can minimize the following parameters.

- average power with area, delay, and peak power constraints
- peak power with area, delay, and average power constraints

- delay with area and peak- or average power constraints
- area with delay, average- and/or peak-power constraints
- any combination of area and power as weighted formula

The optimizer searches for the best combination of architecture and schedule while satisfying all given constraints. They reported peak power reduction in the range of $40 - 60\%$, which comes at the cost of $0.3 - 2.7\%$ penalty in average power. The work also considers mixed supply voltage scenario $(3.3V, 5.0V)$. It is reported that the time penalty is large if the circuit is operated at low voltage, but significant power reduction is achieved.

Shiue [119, 120], Shiue and Chakrabarti [108], and Shiue, Denison and Horak [121] propose different datapath scheduling schemes to minimize peak power at behavioral level. In [108, 121, 120] integer linear programming formulations are proposed, whereas [119] also includes a modified force directed scheduling algorithm. The running time of the proposed modified force directed scheduling algorithm is $O\left(cn^3\right)$, if c is the number of control steps and n is the number of nodes. The scheduling schemes in [119] minimize peak power while satisfying time constraint. The scheduling algorithms in [108, 121, 120] minimize both peak power and peak area while satisfying latency constraints. The simultaneous minimization is performed by the help of multicost objective using the user defined weighting factors. The formulation consider multicycling and pipelining and single supply voltage design. Peak power reductions in the range of $50 - 75\%$ have been reported after scheduling and pipelining. The reduction in peak area is also in the range of $50 - 75\%$.

In [47] a high level synthesis approach is presented by Raghunathan, Ravi, Raghunathan, and Lakshminarayana for transient power management. The power optimization includes the peak power and peak power differential. The authors advocate the need for judicious choice of transient power metric to avoid area and performance overheads. The authors propose the use of data monitor operations for simultaneous reduction of peak power and peak power differential. The proposed scheduling algorithm takes constraints on power characteristics in addition to conventional resource

and time constraints. In this scheme, peak power reduction in the range of $17 - 32\%$ has been obtained. The reduction in the peak power differential is in the range of $25 - 58\%$.

2.4 Scheduling for Variable Voltage Processor

The variable voltage processor has special instructions for controlling power. The supply voltage and clock frequency can be changed at any time by the instructions in the application programs or operating systems. Examples of such processors are Transmeta crusoe, Itsy, Intel StrongARM, etc. The clock frequency is adjusted according to the supply voltage to guarantee correct operation (figure 2.1). The four approaches to manage variable voltage processor are as follows [122] : (1) hardware based (no information), (2) interval-based (load information only), (3) integrated schedulers (all operating system statistics), and (4) application-specific (complete knowledge). In this section, we discuss the scheduling algorithms proposed for variable voltage core-based systems under the assumption that the operating system has a voltage scheduler (as in case 3). We also discuss instruction scheduling for variable voltage processor which assigns voltage and frequency at complier level. The variable scheduling scheme may be either static (off-line) or dynamic (online), but the instruction scheduling schemes are off-line. The variable voltage or instruction scheduling schemes be either preemptive or nonpreemptive. It may be noted that variable voltage processors also referred as variable frequency processors. An overall view of the scheduling algorithms is given in Table 2.4.

Ishihara and Yasuura [123] propose a static voltage scheduling algorithm using integer linear programming formulations. The processor core can have single supply voltage at each instant of time, which can be changed dynamically. The average switching capacitance SC_j per cycle of $task_j$ is calculated as follows.

$$SC_j = \frac{\sum_{i=1}^{EC_j} \sum_{k=1}^{M} CL_k SW_{ijk}}{EC_j} \tag{2.4}$$

where, EC_j is the number of execution cycles for $task_j$, M is the number of gates in the processor, CL_k is the load capacitance of a gate g_k, and SW_{ijk} is the switching count of g_k while the i^{th} cycle

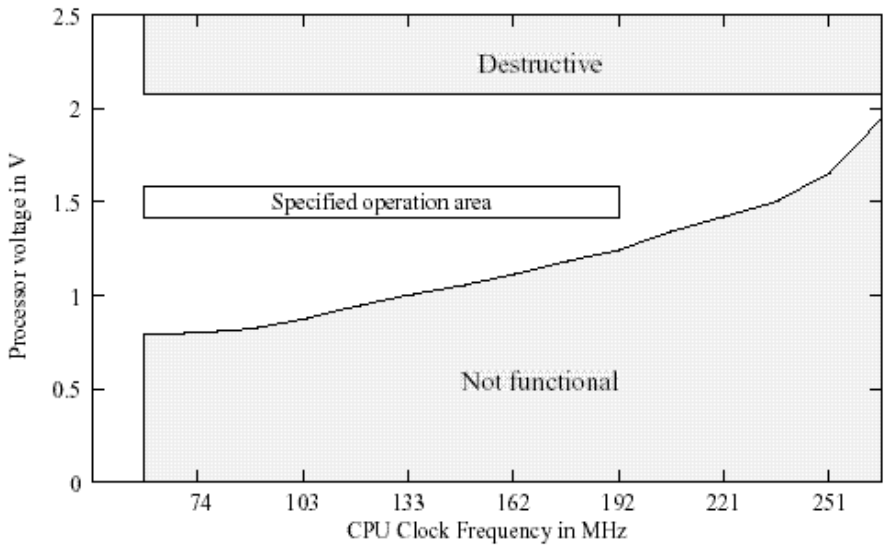


Figure 2.1. Variable Voltage Processor Operation : Voltage Vs Frequency [122]

of $task_j$ is executed. On the basis of the assumption that the processor can use only a small number of discretely variable voltages, the authors have proposed many theorems, some of them are given below.

- For a processor that can use consecutive voltage, only a single voltage can minimize energy consumption satisfying the time constraints.
- The voltage scheduling with at most two voltages minimizes energy consumption usnder any time constraints if a processor can use only a small number of discrete voltages.

The authors have reported energy reduction upto 70%. Various processors with minimum operating voltage $0.9V$ and maximum operating voltage $3.3V$ are used in the experiments. Okuma, Ishihara, and Yasuura [124, 125] propose both static and dynamic voltage scheduling in the above framework.

Hong, Potkonjak, and Srivastava [126] propose preemptive variable voltage scheduling for real-time tasks comprising of both on-line and off-line workloads. The scheduling scheme ensures that the deadlines are met. The variable voltage is generated using DC-DC switching regulator. The authors pointout that the time overhead for clock frequency stabilization is negligible. A periodic

(off-line) task is characterized as $T_i\,(C_i, D_i, P_i)$, where C_i is the worst-case computation time at the highest voltage, D_i is the hard deadline, and P_i is the period. Similarly, a sporadic (on-line) task is characterized as $S_i\,(A_i, C_i, D_i)$, where A_i is the arrival time, C_i is the computation time at highest voltage, D_i is the hard deadline. The on-line scheduling algorithms is heuristic based and has $O\,(m)$ time-complexity for m number of tasks. Two algorithms are proposed that can handle both on-line and off-line tasks. The running time of the optimal algorithm is $O(N + m)$, where N is the total number of requests in each hyperperiod of the n periodic tasks and m is the number of on-line tasks that have been accepted, but uncompleted. The suboptimal heuristic algorithm has time-complexity $O(m)$. The heuristic-based schedulers use a priority task queue in which the tasks are ordered on the Earliest-Deadline-First (EDF). Power reduction upto 20% reported by the authors. In [127], Hong, Kirovski, Qu, Potkonjak, and Srivastava propose a nonpreemptive scheduling heuristic of the same problem.

Mansour, Mansour, Hajj, and Shanbhag [128] propose time constrained and resource constrained instruction scheduling algorithms considering latencies of instructions for a variable voltage processor. The RISC architecture assumed has an integer unit and a floating point unit. The integer unit has a pipelined integer adder, multiplier, and divider. Similarly, the floating point unit has a pipelined floating point adder, multiplier, and divider. The operating voltages assumed are $5.0V, 3.3V, 2.0V,$ and $2.0V$. The architecture also assumed to have load and store instruction for accessing memory. The proposed algorithm is list-based heuristic. The algorithm uses a power gain metric at each node n_i defined as,

$$G_i \;=\; \frac{\Delta P_i}{D_{i,max}} \;=\; \frac{P_i\big(V_{high}\big) - P_i(V_{low})}{D_{i,max}} \tag{2.5}$$

where, $P_i\,(V)$ is the power consumed by n_i when scheduled at voltage V and $D_{i,max}$ is the maximum delay occured by rescheduling n_i. The node with highest G_i is selected for rescheduling. The algorithm maintains a *prologue* of instructions preceeding n_i and an *epilogue* of instructions following n_i in a data flow graph constructed for an instruction set. The time-complexity of the algorithm is $O\left(n^4\right)$. Power savings up to 56% has been reported using this technique.

Table 2.4. Scheduling Algorithms for Variable Voltage Processor

Proposed Work	Working Level	Static or Dynamic	Method Used	Running Time	% Power Savings
Ishihara and Yasuura [123]	OS	Static	ILP	Exponential	70
Okuma, Ishihara, and Yasuura [124, 125]	OS	Static Dynamic	ILP Heuristic	Exponential NA	56 58
Hong, Potkonjak, and Srivastava [126]	OS	Dynamic	Heuristic	$O(N+m)$	20
Hong, Kirovski, and et. al. [127]	System	Static	Heuristic	$O(n^3)$	25
Mansour, Mansour, and et. al. [128]	Circuit and Behavioral	Static	List-based Heuristic	$O\left(n^4\right)$	56
Azevedo, Issenin, and Cornea [129, 130]	Compiler	Static	Heuristic	NA	82
Swaminathan and Chakrabarty [131]	OS Dynamic	Dynamic	ILP Heuristic	Exponential NA	15 NA
Swaminathan and Chakrabarty [132]	OS	Dynamic	Prunning	Polynomial	NA
Hsu, Kremer, and Hsiao [133, 134]	Compiler	Static	Heuristic	NA	70
Pering, Burd and Brodersen [58]	OS	Static	Heuristic	$O(n)$	80
Lee and [135] Krishna [135]	OS	Static Dynamic	Heuristic Heuristic	$O\left(n^2\left(\frac{T_{max}}{T_{min}}\right)\right)$ NA	54.5 65.6
Pouwelse, Langendoen, and Sips [52]	OS	Dynamic	Heuristic	$O\left(n^3\right)$	50
Yao, Demers, and Shenker [136]	OS and Circuit	Static Dynamic	Heuristic NA	$O\left(nlog^2n\right)$ NA	NA NA
Luo and Jha [137]	OS		Heuristic	NA	50
Luo and Jha [138]	OS	Static Dynamic		Polynomial	NA

In [129, 130], Azevedo, Issenin and Cornea propose a dynamic voltage scaling technique that works at the compiler level instead of the operating system level. Checkpoints are introduced at compilation time which indicate places in the code where the processor speed and voltage should be recalculated. Two heuristic based algorithms are proposed. One heuristic results energy reduction of 82% compared to the program execution without DVS. The proposed heuristic algorithms are power and time constrained and is divided into two major phases, such as ahead of time profiling phase and run-time power scheduling phase. The four different clock frequency and voltage combinations supported are $600MHz - 2.2V$, $500MHz - 1.8V$, $400MHz - 1.5V$, and $300MHz - 1.1V$.

On-line scheduling algorithms for periodic tasks are proposed in [131] by Swaminathan and Chakrabarty. The authors describe an intger linear programming (ILP) and a heuristic algorithm. The heuristic algorithm is based on Earliest-Deadline-First (EDF) approach. The CPU assumed has two speeds and the real time tasks are nonpreemptive. For example, for two supply voltages $1.65V$ and $3.3V$ the operating frequencies are $100MHz$ and $200MHz$ respectively. The supply voltage to the CPU is controlled by operating system and the operating system may dynamically switch the voltage during run-time. The ILP based approach results a power reduction of approximately $10 - 15\%$ as compared to the EDF method. In [132], the same authors have proposed a polynomial time-complexity prunning based algorithms called energy-optimal device scheduler (EDS) in the same framework. The prunning is performed based on time and energy. Temporal prunning is done when a partial schedules results in missing deadlines.

Hsu, Kremer, and Hsiao [133, 134] propose a compilation process that faciliates dynamic frequency and voltage scaling for energy reduction with marginal execution time overhead. It is a known fact that the modern architectures exploit temporal and spatial locality. For the programs (computations) with less temporal / spatial locality, the processors often stall, waiting for the memory to provide data. This leads to the principle behind this work, which slows down the CPU that would stall or idle using new compiler strategy. The total program execution time T is divided into

three portions as given below.

$$T = \text{CPUBusy} + \text{MemoryBusy} + \text{BothBusy} \tag{2.6}$$

If the CPU speed is reduced by a factor δ, then new execution time becomes,

$$T_{\text{new}} = \delta * \text{CPUBusy} + maximum\,(\text{MemoryBusy} + \text{BothBusy}, \delta * \text{BothBusy}) \tag{2.7}$$

In order to have the new execution time very close to the original one so that the time penalty is minimal, the follwing four condition must be satisfied : (i) $(\delta - 1) * \text{CPUBusy} \leq 1\%$, (ii) $1 \leq \delta \leq 1 + \frac{\text{MemoryBusy}}{\text{BothBusy}}$, (iii) memory latency is divisible by δ, and (iv) δ has an integral value. The following compilation strategy has been proposed by the authors : (1) Program regions are identified as scheduling candidates, (2) Expected performance is modeled that involves computation of CPUBusy, MemoryBusy, BothBusy, and δ, and (3) Voltage / frequency scheduling instructions are generated for each scheduling candidate. The authors have reported energy reduction of $33\% - 70\%$ under the assumption of transmeta Crusoe processor.

Pering, Burd, and Brodersen [58] introduce a voltage scheduler as a part of operating system. The scheduler determines appropriate operating voltage by analyzing application constraints and requirements. The simulated lpARM processor is based on ARM8 core and designed to operate between $1.1V$ and $3.3V$, with operating frequency between $10MHz$ and $100MHz$. An Earliest-Deadline-First (EDF) policy is used for temporal scheduling, which is optimal for fixed-speed systems. The voltage scheduler needs support for four types of hardwares, such as speed-control register, processor cycle counter, wall-clock time and system sleep control. The proposed scheduling algorithm assumes that all tasks are sporadic and calculate the minimum speed necessary to complete all tasks assumming that they are all currently runable. This speed is calculated as,

$$\text{speed} = maximum\left(\frac{\sum_{j<i} \text{work}_j}{\text{deadline}_i - \text{current time}}\right)_{\forall(i \leq n)} \tag{2.8}$$

when the threads are sorted in EDF order. The algorithm has running time of $O(n)$. Energy reduction up to 80% has been reported.

Both static and dynamic variable voltages scheduling algorithms are proposed by Lee and Krishna [135]. The processor is assumed to run either at high or low voltage and correspondingly at high and low frequency. The first algorithm assigns each task to either high-voltage-fast-clock (H-mode) or low-voltage-slow-clock (L-mode) operation modes while meeting all deadline requirements. On the other hand, the dynamic scheduler switches operational modes based on the accumulated processing workload. In case a task completes before its deadline then the dynamic algorithm reclaims the unused processing time and use less of the high-voltage-fast-clock mode. When the processor switches between the two modes, there is a switching interval for the voltage regulator and the PLL clock generator to complete the mode change and the processor does not function during that time interval. Let us assume that there are n tasks, task$_1$, task$_2$, ... task$_n$, which are numbered in decreasing priority order. Let C_i be the worst-case execution time of task$_i$ when the processor is running in L-mode, D_i be the deadline before which task$_i$ must be completed, and T_i be the minimum time interval between two consecutive instances of task$_i$. It may be noted that $C_i \leq D_i \leq T_i$. If α is the relative processing speed of H-mode with respect to the L-mode ($\alpha \geq 1$), then the scheduling problem is to partition the task into two disjoint subsets such that $\frac{1}{\alpha}\sum_{i\in H}\frac{C_i}{T_i} + \sum_{i\in L}\frac{C_i}{T_i} \leq n\left(2^{1/n}-1\right)$ and $\sum_{i\in H}\frac{C_i}{T_i}$ is minimized. The time-complexity of the scheduler is $O\left(n^2\left(\frac{T_{max}}{T_{min}}\right)\right)$, where T_{max} is the maximum and T_{min} is the minimum of T_i respectively. For static scheduling, average power savings in the range of $43.1 - 54.5\%$ and for dynamic scheduling, average power reduction in the range of $57 - 65.6\%$ are obtained.

Pouwelse, Langendoen, and Sips [52] propose a heuristic called energy priority scheduling (EPS) that arranges the tasks as per the deadline (ascending order priority). In this scheme, the low-priority tasks are scheduled first since they can be preempted to make room for the high-priority tasks. The energy priority scheduler is on-line heuristic that follows an incremental approach and dynamically adjusts the clock schedule when new tasks arrive and old tasks complete or are preempted. The worst-case running time of the proposed heuristic is $O\left(n^3\right)$. The algorithm is implemented as a part of complete system consisting of hardware, OS, clock scheduler and ap-

plications. The hardware is designed using a StrongARM SA1100 processor that supports clock speeds in the range $59 - 251MHz$. Energy reduction up to 50% has been reported.

In [136], Yao, Demers and Shenker invstigate various methods for reducing energy consumption, both at circuit and at operating system level. The authors also propose an off-line scheduling algorithms that executes the job between its arrival and deadline such that for a set of jobs, the energy consumption is minimum. An on-line algorithm has also been proposed. Assuming that J is the set of jobs, for any job $j \in J$, if a_j is the arrival time, b_j is the deadline and R_j is the number of CPU cycles required, then a feasible schedule for J must satisfy the following.

$$\int_{a_j}^{b_j} s(t)\delta\left(job(t), j\right) dt \quad = \quad R_j \tag{2.9}$$

Where, $s(t)$ is the processor speed at time t, $job(t)$ is the job executed at time t and $\delta(x, y)$ is 1 if $x = y$ or else 0. The proposed average rate (AVR) heuristic sets the processor speed at $s(t) = \sum_j d_j(t)$ and use the earliest-deadline policy to choose among the available jobs, where $d_j = \frac{R_j}{b_j - a_j}$ is the average rate requirement or the density. The running time of the optimal algorithm is $O\left(nlog^2 n\right)$.

Luo and Jha [137] propose a power-profile scheduling algorithm for real-time heterogeneous distributed embedded system scheduling algorithm. The algorithm satisfies the precedence relationship, the hard real-time constraints and while minimizing the power consumption by variable voltage scheduling. The scheduler performs variable voltage scaling by addressing variations in power consumption of different tasks and characteristics of different voltage-scalable processing elements (PEs). If n is the number f tasks, k is the number of inter-PE communication edges and M is the number of iterative steps, then the time-complexity of the proposed algorithm is $O\left((n + k)log(n + k) + (n + k)M\right)$. Power reduction upto 50% has been reported by the authors. The same authors have proposed both static and dynamic variable voltage scheduling algorithms for real-time heterogeneous distributed embedded systems in [138]. The time-complexity of the proposed algorithm is polynomial. Power reduction upto 37% has been reported. Similar work is also address in [139] by Luo, Peh and Jha.

2.5 Design and Synthesis for Low-Power or High-Performance Variable Voltage / Frequency / Latency and Multiple Voltage Based Systems

In this section, we discuss the research works proposed in the current literature that deal with multiple supply voltages, variable voltages (frequency) or dyanamic clocking frequency based systems designed for low power or high performance applications. An overview of the proposed works is given in Table 2.5. In the table, for low-power works percentage reduction in power is given and for the high-performance works percentage improvement in performance is tabulated.

Usami, Igarashi and et. al. [66, 68, 69] propose multiple supply voltage based techniques for low power media processor design. The method involves a combination of clustered voltage scaling and row-by-row optimization of power supply. The number of level converters used in the design is minimized because of the clustered voltage scaling. At the same time, the clustered voltage scaling technique maximizes the number of low V_{dd} operating gates, while maintaining the time constraint. A new power bus wiring scheme called "RRPS" (row-by-row optimized power supply) is proposed that provides different supply voltages to each cell row. A in-house layout tool called ChipMaster is developed that places the multiple supply voltage circuits using RRPS scheme and creates the corresponding clocking scheme. The ChipMaster back annotates the estimated interconnect capacitance based on the placement result to the PowerSlimmer (the multiple supply voltage scaling tool). Using the back annotates information, the PowerSlimmer reoptimizes the multiple-supply-voltage netlist. The ChipMaster takes the reoptimized netlist and performs the layout again. The two types of cell libraries used are VDDH and VDDL. VDDH is the conventional high operating voltage cell library and the VDDL is the low operating voltage cell library. The ChipMaster places the VDDH and the VDDL cells close together on the critical path and controls the wire length so that the interconnect delay is minimized. The post-placement netlist optimizer (PNO) performs the gate resizing or replaces cell model which had different gate width and has the same function such that the critical path delay is minimum. The clock tree is designed based on the RRPS scheme. The supply voltage level of all flip-flops are reduced to low-voltage level and also the introduced buffer cells operate at lower voltage. The level converters are placed in the VDDH row to supply the VDDH. The proposed method is used to design a media processor with

$3.3V$ and $1.9V$ supply voltage and $75MHz$ main clock frequency. The power reduction obtained is 47% with an area overhead of 15%. Automated low-power techniques have been proposed in [68, 69] for the same design methodology. The power reduction in the clock tree is 73% as reported in [68]. A design technique combining a variable supply voltage scheme and above clustered voltage scaling is proposed in [67]. Power reduction of 55% is obtained when the design methodology is applied to a video codec design.

Ranganathan, Vijaykrishnan, and Bhavanishankar [59, 60, 140, 141] introduce the concept of dynamic frequency clocking (DFC) and use it in designing high-performance image processing architectures. They propose a SIMD (single instruction multiple data) architecture for real-time image processing applications using dynamic frequency clocking. The VLSI chip developed using the proposed architecture was implemented using Cadence tool. The chip operates in the frequency range of $50-400MHz$. The DFC scheme is more suitable for data flow intensive application (such as DSP and image processing). The DFC scheme is a combination of three concepts : reconfigurable architecture, frequency synthesizer and clock dividing strategy. In the reconfigurable architecture, frequencies are switched as the circuit changes while in DFC scheme, frequency switching occurs based on the units being used. In the clock divider strategy, each unit receives a separate clock operating at a different frequency, whereas in DFC strategy, the same clock switches dynamically. Different functional units can have different maximum operating frequencies, for example, maximum frequency of multiplier has $50MHz$, RAM has $100MHz$, logical unit has $200MHz$, adder has $400MHz$, etc. A dynamic clocking unit (DCU) interprets and decodes each instruction and drives the processing unit at a suitable frequency. For a master clock at $400MHz$, the output frequency, such as $200MHz$, $100MHz$, and $50MHz$ is generated using clock-divider strategy. The speed up, obtained using dynamic frequency, is in the range of $1.79 - 3.0$ as compared to the single frequency operation. The authors advocate the use of dynamic frequency clocking alongwith pipelining for further improvement of perfomance.

Krishna, Ranganathan, and Vijaykrishnan [142, 143] propose a resource and time constrained energy efficient datapath scheduling for synthesis of circuits using dynamic frequency clocking and multiple supply voltages (DFMVS). The proposed scheduling scheme DFMVS has two main

Table 2.5. Design and Synthesis Works on Variable Frequency or Multiple Frequency

Proposed Work	Design or Synthesis	Power or Performance	Operation Mode	Voltage or Frequency	Result
Usami, Igarashi, and et. al. [66, 68]	Design Synthesis	Low-Power	Multiple Voltage	$(3.3, 1.9)V$	47% (max)
Usami, Igarashi, and et. al. [67]	Design	Low-Power	Variable Voltage	NA	55% (max)
Ranganathan, and et. al. [59, 60]	Design	High Performance	Dynamic Frequency	$50 - 400MHz$	1.79-3.0 (times)
Krishna, and et. al. [142, 143]	Synthesis (Scheduling)	Low-Power	Dynamic Frequency	$(5.0, 3.3, 2.4)V$	$2 - 54\%$
Papachristou, and et. al. [144]	Synthesis (Allocation)	Low-Power	Multiple Frequency	NA	50% (max)
Burd, Brodersen, and et. al. [145, 146]	Design	Low-Power	Variable Voltage	$1.2 - 3.8V$	11% (avg)
Kim and Chae [63]	Design	Low-Power	Frequency Scaling	NA	NA
Pouwelse, and, et. al. [122]	Design	Low-power	Variable Frequency	$0.8 - 2.0V$ $59 - 251MHz$	NA
Acquaviva, Benini, and Riccò [147]	Design	Low-power	Variable Frequency	NA	40% (max)
Benini, and et. al. [148, 149]	Design Synthesis	High Performance	Variable Latency	NA	27%
Raghunathan, and et. al. [150]	Synthesis	High Performance	Variable Latency	NA	$1.6\times$
Nowka and [151, 152]	Design	Low-power	Frequency Scaling	$1.0 - 1.8V$	NA
Lu, Benini, and Michelli [153]	Design	Low-power	Frequency Scaling	$103 - 206MHz$	46% (max)

modules, such as dynamic_freq_sched and modify_sched. The first module generates the initial schedule in which the control steps are clocked at different frequencies. The second schedule is a schedule modifier that regroups the operations of the intial schedule such that mutiple supply voltages can be used to reduce the energy consumption. The algorithm is list-based heuristic which takes unscheduled data flow graph, number of resources with their operating frequencies, and the time constraint of the whole schedule as input. Experiments are conducted for three operating voltages ($5.0V, 3.3V, 2.4V$). Results show that using three supply voltages, an average energy saving of 53.5% has been obtained when compared to using a uni-frequency clocking scheme with single supply voltage.

Papachristou, Nourani and Spining [144] propose a resource allocation technology for low-power design using multiple frequency. The contribution of the paper is two fold. First, using nonoverlapping multiple clocking to design a partitioned datapath, so that each partition is assigned a distinct clock. For n number of partitions and master clock frequency of f, the operating frequency of each partition is $\left(\frac{f}{n}\right)$. The inactive partitions are "turned-off" during their off duty cycle to reduce power dissipation. The other contribution is a multiple clock allocation algorithm for power reduction. Two allocation techniques are proposed. In first scheme, called split-allocation, DFG is partitioned based on clock assignments and then each partition is synthesised separately. The second allocation algorithm performs allocation in an integrated way taking into account the clock assignment of DFG nodes. The advantage of this algorithm is better sharing of the resources. Similarly, the advantage of split-allocation technique is its adaptibility with any existing allocator. Experimental results show power reduction with an increase in area penalty.

Burd, Brodersen, and et. al. [154, 145, 146, 155, 50] propose variable voltage (frequency) based system for low-power and high-perfomance applications. The system consists of an ARM8 core, $16kB$ cache and DC-DC regulator. The operating voltage of the systems is in the range of $1.2 - 3.8V$ in [145] and $1.1 - 3.3V$ in [154]. The three components for implementing dynamic voltage scaling in general purpose processor are as follows : a microprocessor that can operate at a wide voltage range, a operating system that can vary processor speed and a regulation loop that can generate the voltage required at a particular speed. A new component which needs to be added in

the operating system is the voltage scheduler. The voltage scheduler controls the processor speed by writing the desired clock frequency to a system control register. This register value is used in the voltage-frequency regulation loop. A ring oscillator, whose output frequency is a function of voltage, serves as the heart of voltage regulator. The authors have reported energy reduction of 11% for MPEG benchmark and reduction of $4.5\times$ in energy for AUDIO benchmark. In [50], authors introduce various modes computation of processors, such as fixed throughput mode, maximum throughput mode and burst throughput mode. The three key principles of energy efficient circuit design proposed are as follows:

- High performance is energy efficient,
- Clock reduction is not energy efficient, and
- Faster operation can limit efficiency.

Kim and Chae [63] propose a VLSI architecture of MPEG2 decoder using frequency scaling. The system clock is adjusted to lowest possible frequency depending on the current workload. The data-dependent applications require less hardware and consume less power than the data-independent applications due to the use of frequency scaling. The system consists of four major components, such as clock controller, programmable clock generator, circuit status detector and synchronizer. The clock controller gets the current status from the system, compares it with the required status, and changes the clock frequency accordingly. The programmable clock generator takes the input from the clock controller and generates appropriate frequency. The circuit status detector guarantees the operating margin of the circuit from the variable clock frequency. The synchronizer is used to synchronize the signals between flip-flops using different clocks.

Pouwelse, Langendoen, and Sips [122] propose a variable frequency and voltage based microprocessor system for energy reduction. The authors report that the energy consumption per instruction at low speed is $\frac{1}{5}$th of the energy required at full speed. The major components of the developed system (called LART) include Intel StrongARM 1100 $190MHz$ processor, $32MB$ volatile memory, $4MB$ non-volatile memory, and voltage regulator. The Linux 2.4.0 operating system kernel module is modified to change the clock frequency. The kernel module also adjust the

memory parameters that control the read / write cycles on the external bus. It should be noted that the external memory is not available during the frequency change. The minimum clock frequency at which the processor can operate is $59MHz$ at $0.79V$. The authors have studied the performance of overall system, memory and applications.

Acquaviva, Benini, and Riccò [147] describe a software-controlled approach for adaptively minimizing energy in embedded systems for real-time multimedia application. The software controller dynamically adjusts processor clock speed (supply voltage) to the frame rate requirements of the incoming multimedia stream. The targeted CPU is Intel StrongARM1100 processor in which twelve frequency levels are available by programming a PLL. Multimedia stream processing algorithms take data streams as input. The input stream which consists of frames is processed in the CPU. Let, C_{eff} is the average switching capacitance, V_{dd} is the supply voltage, f is the CPU frequency, and T_{frame} is the time for processing a frame. The energy consumed for processing a frame is then given by,

$$E_{frame} = V_{dd}^2 C_{eff} f T_{frame} \tag{2.10}$$

Depending on the output bandwidth for a given time T_{max}, the following constraint must be satisfied for just-in-time computation.

$$\frac{1}{T_{frame}} \geq \frac{1}{T_{max}} \tag{2.11}$$

Since the frequency can not be adjusted continuously, there will be some idle time. The authors have reported energy saving up to 40% per frame.

Benini, Macii, Poncino, and Michelli [148] introduce variable-latency units (called *telescopic units*) to improve overall performance. The variable-latency units complete execution in a varaiable number of clock cycles, depending on the input data given to them. There are two overheads involved in such design. First a completion signal is needed and second the controller should be able to synchronize among the components. This is similar to *architectural retiming* proposed in [156] and *speculative completion* proposed in [157]. It should be noted that the speculation completion is an asynchronous datapath design unit. Suppose, f_h is the additional signal of the telescopic unit, T is the clock cycle time without variable-latency operation, T^* is the clock cycle

time with telescopic units, and $Prob(f_h)$ is the probability that f_h is one. The following condition must be satisified for throughput improvement.

$$Prob(f_h) \quad < \quad \frac{2(T-T^*)}{T} \tag{2.12}$$

Heuristic algorithms, such as BDD-based heuristics and sum-of-product (SOP) based heuristics are proposed for synthesis of telescopic units. Various experiments conducted showed that throughput improvement is obtained at the cost of area penalty. Benini, Micheli, Macii, Odasso, and Poncino [149] propose another automatic synthesis technique formulated as *time supersetting* problem for synthesizing telescopic units. Raghunathan, Ravi, and Lakshinarayana [150] proposed high-level synthesis methodology for synthesis of variable latency units proposed above in [148, 149]. The authors propose novel techniques to reduce the area penalty. The proposed algorithms use iterative approach and synthesize the circuit under resource constraints. Performance improvement of $1.6\times$ was obtained with maximum area penalty of 17.9%. It has also been reported that the performance improvement is accompanied with power savings of 35.7%.

Nowka and et. al. [151, 152] discuss a system-on-a-chip processor using dynamic voltage and frequency scaling. The voltage or frequency is adaptible to change in performance demand and power consumption. The targeted processor is fixed voltage IBM PowerPC 405 core. The operating voltage of the chip is in range $1.0 - 1.8V$. An on-chip regulator alongwith the PLL helps in continuously operating the chip even when the supply voltage is modified. When the demands for resources are low, the active power consumption is reduced using dynamic voltage scaling, frequency scaling, unit and register level functional clock gating. Both the voltage and the frequency of the processor are varied using software control and both active and standby power is minimized. The processor can enter a low-leakage sleep state and a state-preserving deep-sleep state to minimize standby power consumption.

Lu, Benini, and Michelli [153] discuss the energy reduction of interactive systems for mixed workloads of multimedia applications using dynamic frequency (voltage scaling). The proposed technique is software-based works for processors that have only finite frequencies. The main idea

is to insert buffers such that constant output can be maintained even though the input rate may be changing. The multimedia programs are divided into into stages and data buffers are inserted between them. The data buffers support constant output rates, allow frequency scaling and shorten the response times of sporadic jobs. Data are processed and stored in the buffers when the processor runs at a higher frequency. Later, the processor runs at a lower frequency to reduce power and data are taken from the buffers to maintain the same output rate. Before the buffers become empty, the processor begins to run at a higher frequency again. The authors construct frequency-assignment graphs. Each vertex represents the current state of the buffers and the frequencies of the processor. An efficient graph-walk algorithm that assigns frequencies to reduce energy has been proposed. The time-complexity of the algorithms are polynomial, one is $O\left(|V|^2\right)$ and other $O\left(|V|^3\right)$. The method reduces the power consumption of an MPEG program by 46%.

2.6 Hardware Based Digital Watermarking Systems

There are several image watermarking algorithms available in current literature, which are implemented using software. The watermarking schemes work in spatial domain, DCT domain and wavelet domain. However, hardware based watermarking systems are quite few. In this section, we discuss the hardware based watermarking systems. A comparative view of the proposed watermarking chips is given in Table 2.6.

Strycker, Termont, Vandewege, Haitsma, Kalker, Maes and Depovere [158] propose a real-time watermarking scheme for television broadcast monitoring. They address the implementation of a real-time watermark embedder and detector on a Trimedia TM-1000 VLIW processor developed by Philips semiconductors. The watermark is in spatial domain. In the insertion procedure, pseudo-random numbers are added to the incoming video stream. The depth of watermark insertion depends on the luminance value of each frame. The watermark detection is based on the calculation of correlation values. Mathai, Kundur and Sheikholeslami [159] present hardware implementation of the same video watermarking algorithm. The chip is implemented using 0.18μ technology. The authors did not provide any lay out details for the proposed hardware and did not mention its power consumption and operating frequency.

Table 2.6. Watermarking Chips Proposed in Current Literature

Proposed Work	Type of Watermark	Target Object	Working Domain	Techno-logy	Chip Area	Chip Power Consumption
Mathai and et. al. [159]	Invisible Robust	Video	Wavelet	0.18μ	NA	NA
Tsai and Lu [160]	Invisible Robust	Image	DCT	0.35μ	3.064×3.064 mm^2	$62.78mW$ $3.3V, 50MHz$
Garimella and et. al. [161]	Invisible Fragile	Image	Spatial	0.13μ	3453×3453 μm^2	$37.6\mu W$ $1.2V$

A DCT domain invisible watermarking chip is presented by Tsai and Lu [160]. The watermark systems embeds a pseudo-random sequence of real numbers in a selected set of DCT coefficients. They also proposed a JPEG architecture incorporating the watermarking module in it. The watermark is extracted without resorting to the original image. The authors claim that the watermark is resistant to the JPEG attacks upto 10% compression ratio. The watermark chip is implemented using TSMC $0.35\mu m$ technology and occupies a die size of $3.064 \times 3.064mm^2$ for 46374 gates. The chip consumes $62.78mW$ power when operated at $50MHz$ with $3.3V$ supply voltage.

Garimella, Satyanarayan, Kumar, Murugesh and Niranjan [161] propose an watermarking VLSI arcitecture for invisible-fragile watermarking in spatial domain. In this scheme, the differential error is encrypted and interleaved along the first sample. The watermark can be extracted by accumulating the consecutive LSBs of pixels and then decrypting. The extracted watermark is then compared with the original watermark for image authentication. The ASIC is implemented using 0.13μ technology. The area of the chip is $3453 \times 3453\mu m^2$ and consumes $37.6\mu W$ power when operated at $1.2V$. The critical path delay of the circuit is $5.89ns$.

2.7 This Dissertation

The synthesis techniques discussed in Sections 2.1 and 2.2 are based on a single clock frequency and consider multiple supply voltages, voltage scaling, capacitance reduction, and switching activity reduction to minimize total energy or average power. However, not both at the same time. Further, these works have not considered dynamic frequency clocking or transient power

reduction. The works in Section 2.3 address only peak power issues and do not include energy minimization or transient power. It it evident from Section 2.4 and Section 2.5 that voltage scaling or frequency is an effective method for power reduction and performance improvement. In this disseration, we propose scheduling techniques to minimize total energy (or average power). We also propose scheduling techinques for peak power and transient power reduction. Behavioral synthesis frameworks are proposed for reduction simultaneous reduction of energy, average power, peak power and transient power. A new parameter called *Cycle Power Function* (CPF) is defined which is an equally weighted sum of normalized mean cycle power and normalized mean cycle differential power. Minimizing this parameter using multiple supply voltages (MV), dynamic frequency clocking (DFC) and multicycling results in the reduction of both energy and transient power. Both ILP and heuristics based approaches have been investigated. In Section 2.6, we have discussed the few watermarking hardware systems available. In this dissertation we introduce few VLSI implementations of existing watermarking algorithms. We intend to use multiple supply voltage and variable frequnecy in the watermarking chip design.

CHAPTER 3

ENERGY MINIMIZATION

Dynamic frequency scaling has been explored at the CPU and system levels for power optimization. In this chapter, we discuss datapath scheduling algorithms that use multiple supply voltages and dynamic clocking in a co-ordinated manner in order to reduce energy and energy delay product [54, 55]. The strategy is to schedule high energy units, such as the multipliers at lower frequencies so that they can be operated at lower voltages to reduce energy consumption and the low energy units, such as adders at higher frequencies, to compensate for speed. The proposed heuristic based time and resource constrained algorithms have been applied to various high level synthesis benchmark circuits under different time and resource constraints. This chapter is organised as follows. Section 3.1 discusses the target architecture model and frequency selection scheme. Section 3.2 and 3.3 present the time constrained scheduling (TC-DFC) and the resource constrained scheduling (RC-DFC) algorithms followed by results and conclusions.

3.1 Target Architecture and Datapath Specifications

The target architecture model assumed in the design of the scheduling schemes is shown in Fig. 3.1. All functional units have one register each and one multiplexor. Each functional unit feeds into a single register. The register and the multiplexor operate at the same voltage level as that of the functional units. Level converters are used when a low-voltage functional unit is driving a high-voltage functional unit [65, 95]. A controller decides which functional units are active in each control step and those that are not active are disabled using the multiplexors. The controller has a storage unit to store the parameters cfi_c obtained from the scheduling. The cycle frequency f_c ($=\frac{f_{base}}{cfi_c}$) is generated dynamically and a functional unit operating at one of the supply voltages is activated.

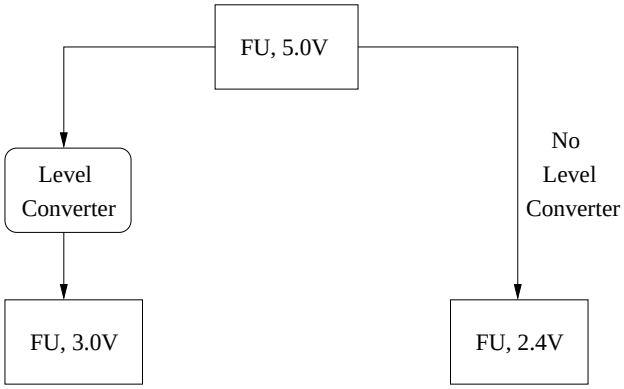


Figure 3.1. Level Converters Needed for Stepping up Signal

The datapath is specified as a sequencing data flow graph (DFG) [21]. Each vertex of the DFG represents an operation and each edge represents a dataflow (or dependency). The DFG does not support the hierarchical entities. The conditional statements are handles using comparison operation. Since, the dynamic frequency clocking scheme is useful only in the case of signal processing applications, we assume that the above does not exist in the directed acyclic DFG representation of datapaths. Each vertex has attributes that specifies the operation type such as addition, subtraction, multiplication or null opeations (NOPs).

The delay of a control step is dependent on the delays of the functional unit and the multiplexer and register pair. Let, d_{reg} be the delay of the register, d_{mux} be the delay of the multiplexor, d_{fu} be the delay of the functional unit and d_{level} be the delay of the level converter. The worst case operational delay of a functional unit can be written as :

$$d_{FU} = d_{reg} + d_{mux} + d_{fu} + d_{level} \tag{3.1}$$

The register delays include the set-up and propagation delays. The delay of control step d_c is the delay of the slowest functional unit in the control step c. Using the above delay model, the worst case delays of the library components are estimated. For a given base frequency (f_{base}), maximum frequencies of each FU is scaled down to operating frequencies given by ($\frac{f_{base}}{cfi_c}$), where, $cfi_c = 1, 2, ..., anyinteger$. The value of cfi_c is bound by the product of the total number of resource types and number of voltage levels. For three frequency levels, the possible frequencies

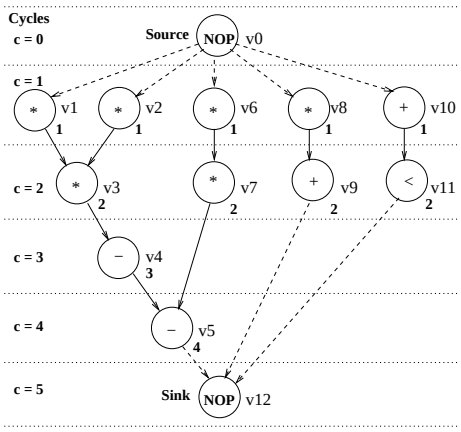


Figure 3.2. HAL Differential Equation Solver (with ASAP labels)

are, $ALU_{High}(cfi_c = 1)$, $ALU_{Med}(cfi_c = 2)$, $ALU_{Low}(cfi_c = 4)$, $MULT_{High}(cfi_c = 2)$, $MULT_{Med}(cfi_c = 4)$ and $MULT_{Low}$ $(cfi_c = 8)$. For example, if the base frequency fed to the DCU is $36MHz$, then the frequencies generated are, $28MHz$, $9MHz$ and $4.5MHz$. The clock frequency for a given control step is the minimum of the operating frequencies of all FUs active in that step.

3.2 Time Constrained Scheduling

The datapath is represented in the form of a data flow graph (DFG) constructed as a sequencing graph. Fig. 3.2 shows such a graph for the HAL benchmark. The inputs to the algorithm are an unscheduled data flow graph (UDFG), the scaled down operating frequencies, and the execution time constraint T_c for the whole schedule. To get more energy savings and at the same time maintain performance, the multipliers are to be operated at as low frequencies as possible and the adders at as high frequencies as possible. This objective can be achieved if adders / subtractors are not operated alongwith multipliers in the same duty cycle. In cases, when they are to be operated during the same cycle to meet the time constraint, energy savings will come from the multipliers only. Initially, TC-DFC generates a schedule such that the low frequency operators are scheduled at earlier steps and the high frequency operators are scheduled at later steps. Later on, the TC-DFC modifies the schedule by moving operations from one step to another with the objective of meeting the time constraint. It then finds appropriate clock cycle width and assigns appropriate voltage.

Step 1	: Find an ASAP schedule for the sequencing UDFG.
Step 2	: Create a priority list of vertices using the ASAP schedule in Step 1.
Step 3	: Assign control steps to the operations such that the higher priority vertex scheduled at earlier time stamp, precedence is satisfied, and the multiplications and ALU operations are not scheduled in the same cycle.
Step 4	: Find the cycles having only ALU operations and, those with only multiplications, and those with both ALU operations and multiplications (mixed) for the currently obtained schedule.
Step 5	: Create a priority list of clock cycles such that cycles with only ALU operations get higher priority than the cycles with only multiplications or those with mixed operations (cycles with only multiplications get higher priority than the cycles with mixed operations).
Step 6	: Initialise cycle frequency to the minimum operating frequency.
Step 7	: If time constraint is not satisfied, the highest priority cycle is assigned the next higher frequency and repeat the step for the next higher priority cycle if necessary.
Step 8	: If any cycle has multiplier operating at highest frequency, then eliminate the cycle having minimum number of ALU operations, adjust the schedule and go to Step 4.
Step 9	: Do voltage assignment and determine energy details.
Step 10	: Find the cycle frequency index for each cycle.

Figure 3.3. TC-DFC Scheduling Algorithm Flow

3.2.1 Algorithm Flow

Fig. 3.3 shows the flow of the proposed TC-DFC scheduling algorithm. In step 1, an ASAP schedule for the data flow graph (DFG) is determined. In step 2, the scheduler creates a priority list of the vertices such that all multiplications (i.e low frequency operators) are grouped with higher priority than the ALU operations (i.e. high frequency operators, such as additions, subtractions, comparisons, etc.). Among the multiplication operations higher priority is given to the operations with smaller ASAP time stamp, same is done for the group of ALU operations. In step 3, the vertices are time stamped such that no multiplication and ALU operations scheduled to function concurrently. In addition, it is made sure that operation precedence is satisfied and higher priority vertex scheduled at earlier time stamp. In step 4, for the current schedule, the cycles are categorised as, cycles having only ALU operations, only multiplication and both ALU operations and multiplication (mixed operations). In step 5, priority list of clock cycles created such that cycles with only ALU operations get higher priority than cycles with only multiplications or mixed operations. The cycles with only multiplications get higher priority than the cycles with mixed operations.

Further, among the cycles with only ALU (or multiplication) operations higher priority is given to the cycle having lesser number of ALU (or multiplication) operations. Similarly, among the cycles with mixed operations higher priority is given to cycles having lesser number of multiplications. In step 6, initial cycle frequency is taken as minimum operating frequency with the help of Table 3.3. In step 7, in order to fulfil time constraint, the highest priority cycle frequency is increased using Table 3.3. If needed the process is repeated for the next higher priority cycle. In step 8, if it is found that a cycle with multiplication is highest voltage then the cycle having minimum number of ALU operations is eliminated and the schedule is adjusted. In step 9, voltage assignment is done and energy estimates for entire DFG is found out. In step 10, the cycle frequency index for each cycle is found out. The pseudo-code for the algorithm is given in Fig. 3.4.

Table 3.1. List of Functions used in the TC-DFC Algorithm

Functions	Description	Complexity
ASAPScheduler	: Determines the ASAP time of the vertices.	$\Theta(\|V\| + \|E\|)$
CreateVertexPriorityList	: Creates a priority list of vertices such that the vertex with lower operating frequency gets the higher priority.	$\Theta(\|V\|)$
TOP	: Finds the first vertex from priority list array.	$\Theta(1)$
CheckFrequencyConstraint	: Checks the frequency constriant in a cycle.	$\Theta(1)$
Maximum	: Finds the maximum value from an array.	$\Theta(c)$
CreateCyclePriorityList	: Constructs the cycle priority list in an array.	$\Theta(c)$
FindMinimumFrequency	: Finds the minimum available frequency.	$\Theta(L_f)$
CalculateDelay	: Calculates the critical path delay.	$\Theta(c)$
FindNextHigherFrequency	: Finds the next higher available frequency.	$O(L_f)$
FindCycleWithMinimumALU	: Finds the control step with minimum number of ALU operations.	$\Theta(cR_T)$
Adjust Predecessor	: Adjusts time stamp of predecessor	$O(\|V\|)$
Adjust Successor	: Adjusts time stamp of successor	$O(\|V\|)$
Update *CyclePriorityList*	: Updates the array.	$O(c)$
Voltage Assignment	: Assigns voltage to each vertex.	$\Theta(\|V\|)$
Find Cycle Frequency Index	: Finds cycles frequency indices of all cycles.	$\Theta(c)$

Table 3.2. List of Variables and Data Structures used in the TC-DFC Algorithm Description

Data Structures	Descriptions
ASAPSchedule	: An array used to store ASAP time stamp of each vertex.
TC-DFCSchedStep	: An array used to store TC-DFC time stamp of each vertex.
ScheduledVertexList	: An array used to store vertices already scheduled.
VertexPriorityList	: An array used to store vertices in a priority order.
CyclePriorityList	: An array used to store control steps in a priority order.
TC-DFCNoOfSteps	: Total number of control steps of TC-DFC schedule.
CycleFrequencyList	: An array used to store frequency of each cycle.
cycle, ControlStepIndicator	: Temporary variables.

3.2.2 Pseudocode Description

The list of functions needed in implementation of the algorithm is given in Table 3.1. Similarly, the data structures or the identifiers used in the algorithm description is summarized in Table 3.2. The pseudocode of the algorithm is given in Fig. 3.4.

Table 3.3. TC-DFC Freqeuncy Selection : from left $\rightarrow$ right

	$MULT_{Low}$	$MULT_{Med}$	ALU_{Med}	ALU_{High}
Frequency	$4.5MHz$	$9MHz$	$18MHz$	$36MHz$
cfi_c	8	4	2	1

Table 3.4. Vertex Priority List

v0	v1	v2	v6	v8	v3	v7	v10	v9	v11	v4	v5	v12
0	1	2	3	4	5	6	7	8	9	10	11	12

In line 01, the ASAP schedule for the UDFG is found out. The procedure *CreateVertexPriorityList* creates the *VertexPriorityList* such that the vertex with the lower operating frequency gets the higher priority to be scheduled at earlier a control step than the lower priority vertices. Table 3.4 shows such an list obtained for the DFG given in Fig. 3.2. *TC-DFCSchedSteps*$_{v_i}$ (line 02) is a data structure that contains the clock cycle step for any vertex v_i. It is initialized to zero for the source vertex. *ScheduledVertexList* (line 02) is a data structure to maintain the list of vertices already scheduled which is initialised to the source vertex. The while loop (line 03) takes the highest priority vertex each time (line 04) and schedules it in an appropriate cycle checking

```
TC-DFCAlgorithm(UDFG, T_c, Operating Frequency)
{
(01) ASAPScheduler(UDFG); CreateVertexPriorityList(ASAPSchedule); cycle = 1;
(02) TC-DFCSchedSteps_{v_0} = 0; ScheduledVertexList = v_0; // source vertex scheduled
(03) while(VertexPriorityList ≠ NULL)
     {
(04)   v_i = TOP(VertexPriorityList);
(05)   if(v_i ∉ ScheduledVertexList and AllPredecessor_{v_i} ∈ ScheduledVertexList)
       {
(06)     if(CheckFrequencyConstraint(cycle))
                  then cycle = Maximum (TC-DFCSchedSteps) + 1;
(07)     else schdule in current cycle;
(08)     TC-DFCSchedSteps_{v_i} = cycle; VertexPriorityList = VertexPriorityList − v_i;
(09)     ScheduledVertexList = ScheduledVertexList ∪ v_i;
       } // end if (05)
     } // end while (03)
(10) TC-DFCNoOfSteps = Maximum(TC-DFCSchedSteps);
(11) CreateCyclePriorityList(CurrentSchedule, TC-DFCNoOfSteps);
(12) CycleFrequencyList = FindMinimumFrequency(Table 3.3);
(13) T_s = CalculateDelay(CycleFrequencyList); ControlStepIndicator = 1;
(14) while (ControlStepIndicator)
     {
(15)   while (T_s > T_c)
       {
(16)     c_i = TOP(CyclePriorityList);
                  CycleFrequencyList_{c_i} = FindNextHigherFrequency(Table 3.3);
(17)     T_s = CalculateDelay(CycleFrequencyList);
       } // end while (15)
(18)   if (no multiplier is operating at highest frequency) then ControlStepIndicator = 0;
(19)   else
       {
(20)     c_i = FindCycleWithMinimumALU(for all cycle c_i);
(21)     for each v_i ∈ c_i do reduce time stamp of v_i
                  and adjust Predecessor_{v_i} and Successor_{v_i}
(22)     CycleFrequencyList = FindMinimumFrequency(Table 3.3);
(23)     T_s = CalculateDelay(CycleFrequencyList); Update CyclePriorityList;
(24)   } // end else (19)
     } // end while (14)
(25) Do voltage assignment ; Find cycle frequency index ;
} // End Algorithm TC-DFC
```

Figure 3.4. Pseudo-code for TC-DFC Scheduling Algorithm

for the frequency constraint violation provided all of its predecessors are already scheduled. The function *CheckFrequencyConstraint* (line 06) helps in checking the frequency constraint. This assures that two vertices operating at different frequencies are not scheduled during the same cycle. *TC-DFCNoOfSteps* (line 10) is the number of control steps for the schedule already generated.

Procedure *CreateCyclePriorityList* (line 11) creates the *CyclePriorityList* in which the higher priority cycles will be assigned higher frequencies. Table 3.5 shows such a list obtained for the schedule generated in using lines 01-09. The data structure *CycleFrequencyList* (line 12) is used to store the operating frequency of each cycle. Initially, each cycle is assigned the minimum frequency from Table 3.3, and the critical delay of the schedule is found (line 12). While the time constraint is not satisfied, with the help of *CyclePriorityList* appropriate clock cycles is assigned to the next higher frequency and checked if time constraint is satisfied (line 14-24). The algorithm terminates if no cycle has multiplier scheduled operating at highest frequency (line 18). Otherwise, the cycle having minimum number of ALU is eliminated (line 20) and *CyclePriorityList* is updated, and lines 14-24 are repeated. Table 3.6 shows an updated *CyclePriorityList*. Finally, proper voltage value are assigned to the vertices. The algorithm also calculates the energy value of the schedule. Algorithm finds the cycle frequency index using *CycleFrequencyList*. The final scheduled datapath is shown in Figs. 3.5(a), 3.5(b) and 3.5(c) for different time constraints.

Table 3.5. Cycle Priority List : $T_c \approx 2 * T_{cp} or 1.75 * T_{cp}$

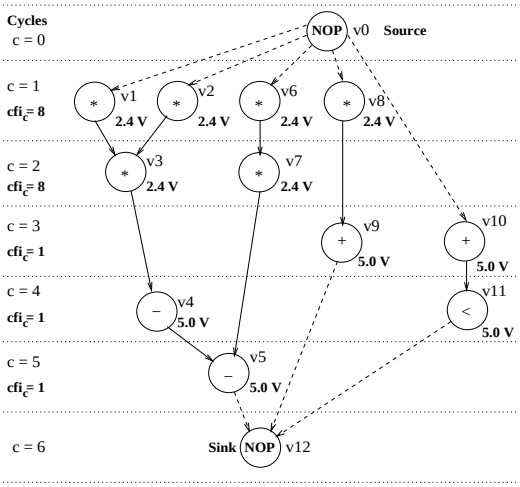
Cycles	c5	c4	c3	c2	c1	c6	c0
Priorities	0	1	2	3	4	5	6

Table 3.6. Cycle Priority List : $T_c \approx 1.5 * T_{cp}$

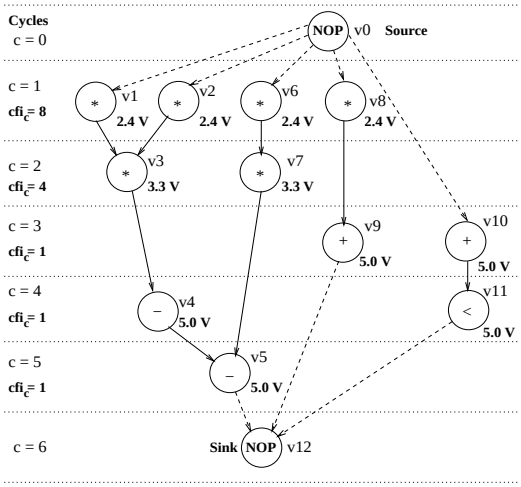
Cycles	c4	c3	c2	c1	c5	c0
Priorities	0	1	2	3	4	5

3.2.3 Time Complexity

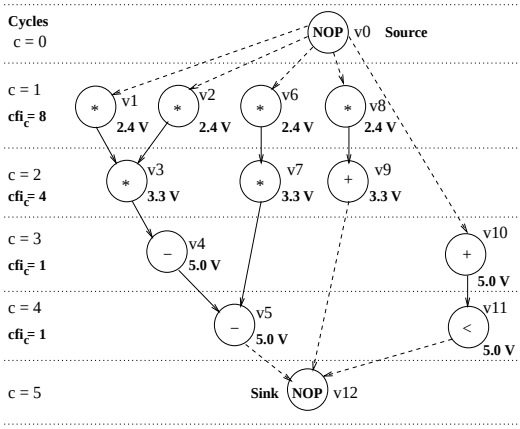
Let there be $|V|$ number of vertices and $|E|$ number of edges in the DFG. Suppose the number of control steps found out from the ASAP scheduling is c. Let L_f denote the number of frequency



(a) Time Constrained : $T_c \approx 2.0 * T_{cp}$



(b) Time Constrained : $T_c \approx 1.75 * T_{cp}$



(c) Time Constrained : $T_c \approx 1.5 * T_{cp}$

Figure 3.5. Schedules Obtained for HAL Benchmark for Different Time Constraints using TC-DFC

levels and R_T denote the number of resource types. Based on the time complexity of the different functions given in Table 3.1, we provide the following analysis for the worste-case running time of the TC-DFC algorithm. Time taken by the instruction from line 01-02 is $\Theta(|V| + |E|) + \Theta(|V|)$. The running time of the code-segment line 03-09 is $\Theta(c|V|)$. Similarly, $\Theta(c) + \Theta(L_f)$ is the running time of the code segment line 10-13. Assuming the *while* loops are executed for constant number of time (independent of the input size $|V|$ or $|E|$), the time complexity of the code segment line 14-25 is $\Theta(cR_T) + \Theta(|V|) + \Theta(L_f) + \Theta(c)$. Without loss of generality, we can assume that the R_T, L_f and c are upper bounded by the number of vertices $|V|$. Using this assumption the overall running time of the algorithm is expressed as : $\Theta(|V| + |E|) + \Theta(|V||V|)$. For strongly data-dependency, we have $|E| \approx |V|^2$ and for weak data-dependency $|E| << |V|^2$. In either case, *the simplified time-complexity of the TC-DFC scheduling algorithm is* $|V|^2$*, meaning the time-complexity is polynomial to the number of vertices (operations) in the data flow graph.*

3.3 Resource Constrained Scheduling

The objective of RC-DFC is to minimize the energy-delay-product while assigning a schedule for the DFG. For a resource i operating in clock step c, let, (i) $\alpha_{i,c}$ be the switching, (ii) $C_{i,c}$ be the load capacitance and (iii) $V_{i,c}$ be the operating voltage. If a level converter is needed, it is considered as a resource needed in the particular clock cycle in which it needs to step up the signal. If N is the total number of clock cycles for the DFG, NR_c is the number of resources active in cycle c, and f_c is the cycle frequency, then, the total energy consumption of the DFG is given by Eqn. 3.2.

$$E_D = \sum_{c=1}^{N} \sum_{i=1}^{NR_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 \tag{3.2}$$

The energy-delay-product (EDP) is characterised by Eqn. 3.3.

$$EDP_D = E_D * T_D = \left(\sum_{c=1}^{N} \sum_{i=1}^{NR_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 \right) * \sum_{c=1}^{N} \frac{1}{f_c} \tag{3.3}$$

The objective of RC-DFC is to minimize the EDP given as equation 3.3. RC-DFC attempts to operate the multipliers at as low frequency as possible, the resulting decrease in per-

Table 3.7. Frequency Selection (From Left to Right in Each Step)

FUs in a cycle		Frequency priority order
MULT	-	$MULT_{Low}, MULT_{Med}, MULT_{High}$
MULT	ALU	$MULT_{Low}, ALU_{Low}, MULT_{High}$
-	ALU	$ALU_{High}, ALU_{Med}, ALU_{Low}$

Table 3.8. Resource Look-up Table (order, From Left to Right)

Clock	MULT			ALU		
Cycle	2.4 V	3.3 V	5.0 V	5.0 V	3.3 V	2.4 V
c	1	2	1	1	1	0

formance is compensated by operating the ALUs at as high frequency as possible. Depending on which functional units are active in a given cycle, the algorithm determines the frequency using a lookup table (LUT), called "frequency selection LUT", such as the one shown in Table 3.7 scanning it left to right. In a schedule, if only multipliers are needed in a particular cycle the frequency selection is in the order $MULT_{Low}, MULT_{Med}, MULT_{High}$. If both multipliers and the ALUs are all operating in a given clock cycle, the frequency selection is in the order $MULT_{Low}, ALU_{Low}, MULT_{High}$. If only ALUs are operating in a control step, then the frequency selection is in the order $ALU_{High}, ALU_{Med}, ALU_{Low}$. Another lookup table called "resource assignment LUT" constructed considering the resource constraints is used to match the selected frequency with a corresponding voltage level. The resources are assigned scanning the LUT, from left to right. The scheduling algorithm uses heuristics to minimize the number of times level conversions needed. An example resource assignment LUT, is shown in Table 3.8 with resource constraints: one MULT at $2.4V$, two MULT at $3.3V$, one MULT at $5.0V$, one ALU at $3.3V$ and one ALU at $5.0V$. The dimension of this LUT depends on the total number of clock cycles of the schedule and the number of resource types. It should be noted that the arrangement of the MULTs is in the order from low to high voltage, whereas for the ALUs it is from high to low. The LUT is updated during each assignment to make sure that the resource-constraints are not violated.

Step 1	: Derive ASAP and ALAP schedules for the unscheduled DFG.
Step 2	: Determine the number of resources at different operating voltages.
Step 3	: Using above number of resources modify the schedules obtained in Step 1.
Step 4	: Calculate the total number of control steps which is the larger those of ASAP and ALAP schedules from Step 3.
Step 5	: Construct the ”resource assignment LUT” and ”frequency selection LUT”.
Step 6	: Find the vertices having non-zero mobility and vertices with zero mobility and assume ASAP schedule in Step 3 as the current schedule.
Step 7	: Do voltage and frequency assignment using the current schedule and the LUTs.
Step 8	: Taking a vertex with non-zero mobility time stamp it using LUTs such that energy delay product of the execution of whole DFG is minimum.
Step 9	: Adjust current schedule, predecessor and successor time stamps, LUTs, and repeat Steps 7 and 8 to time stamp remaining non-zero mobility vertices.
Step 10	: Determine the clock frequency index for each cycle.

Figure 3.6. RC-DFC Scheduling Algorithm Flow

3.3.1 Algorithm Flow

Fig. 3.6 shows the flow of the proposed algorithm. The data flow graph is modeled as a sequencing graph [21]. The inputs to the algorithm are an unscheduled data flow graph (UDFG), the resource constraints which include the number of resources, their corresponding operating voltages and the scaled down operating frequencies. In step 1, the scheduler determines the ASAP and the ALAP schedules for the UDFG. In step 2, the total number of resources is found out as the sum of each resource at different voltage levels. In step 3, the ASAP and ALAP schedules of step 1 are modified using the number of resources found in step 2. In step 4, the total number of control steps for both ASAP and ALAP schedule are found out and the number of control steps for the final steps is assumed to be the maximum of the two. In step 5, the ”resource assignment LUT” and ”frequency selection LUT” are constructed. In step 6, the vertices having non-zero mobility and the vertices with zero mobility are found out and the current schedule is initialized as the ASAP schedule obtained in step 3. In step 7, voltage and frequency assignments are made for the current schedule using the LUTs. In step 8, the scheduler finds a proper step for each vertex having non-zero mobility such that the number of level converters needed for the execuction of the whole DFG is minimum. As long as the voltage and frequency assignments follow the LUTs order, energy consumption is kept to a minimum. In step 9, current schedule, LUTs are adjusted to satisfy the

Table 3.9. List of Functions used in the RC-DFC Algorithm

Functions	Description	Complexity
ASAPScheduler	: Determines ASAP time of the vertices.	$\Theta(\|V\| + \|E\|)$
ALAPScheduler	: Determines ALAP time of the vertices.	$\Theta(\|V\| + \|E\|)$
ModifySchedule	: Modifies the unconstrained schedules to incorporate resorce constraints.	$\Theta(\|V\| + \|E\|)$
ConstructResAssignmentTable	: Constructs resource assignment LUT.	$\Theta(cL_vR_T)$
Maximum	: To find maximum of to control steps.	$\Theta(1)$
FindResTypeForEachVertex	: Identifies the FU needed for each vertex.	$\Theta(\|V\|)$
ConstructFreqSelectionLUT	: Constructs frequency selection LUT.	$\Theta(L_f)$
FindMobileVertexList	: Finds the mobility of each vertex.	$\Theta(\|V\|)$
AllocateVoltAndFreq	: Allocates the voltage and frequency levels using LUTs and current schedule.	$\Theta(c\|V\|L_vR_T)$
CalculateEDP	: Calculates the EDP of the whole DFG.	$\Theta(\|V\|)$
AdjustSchedule	: Adjusts the predessor and successor time stamps such that the precedence is satisfied.	$O(\|V\|)$
Update Res Assignment LUT	: Updates resource assignment LUT.	$\Theta(1)$
FindEnergyAndDelay	: Determines energy and delay.	$\Theta(\|V\|)$
FindCycleFreqIndex	: Finds cycles frequency indices.	$\Theta(c)$

precedence. In step 10, cycle frequency indices are found for all cycles which would be stored in the controller and would be fed to the DCU for dynamic frequency generation. The algorithm terminates once all non-zero mobility vertices are scheduled.

3.3.2 Pseudocode of the Resource Constrained Algorithm

The list of functions needed in implementation of the algorithm is given in Table 3.9. Similarly, the data structures or the identifiers used in the algorithm description is summarized in Table 3.10. The pseudocode of the algorithm is given in Fig. 3.7.

The inputs to the algorithm are the unscheduled data flow graph (UDFG) and resource constraints which includes number and type of each functional units, the operating voltage levels and the operating frequencies. The procedures in line 01, *ASAPScheduler* and *ALAPScheduler* find the unconstrained ASAP and ALAP schedules for the UDFG respectively. In line 02, the total number of multiplier and ALU FUs with different voltage levels is determined. For example, if the resource constraint is 2 ALUs at $2.4V$, 1 ALU at $3.3V$, 1 multiplier at $2.4V$, and 3 multipliers at $5.0V$, then

```
RC-DFCAlgorithm(UDFG, FUs, Voltage Levels, Operating Frequencies)
{
(01)ASAPScheduler(UDFG); ALAPScheduler(UDFG);
(02)MULT = ∑ Multipliers of different voltage levels;
                        ALU = ∑ ALUs of different voltage levels;
(03)ModifySchedule(ASAPSchedule, MULT, ALU);
                        ModifySchedule(ALAPSchedule, MULT, ALU);
(04)NoOfControlSteps = Maximum(ASAPControlSteps, ALAPControlSteps);
(05)ConstructResAssignmentLUT(NoOfControlSteps, FUs);
(06)FindResTypeForEachVertex(UDFG); ConstructFreqSelectionLUT(Operating Frequency);
(07)FindMobileVertexList(ASAPSchedule, ALAPSchedule);
                        CurrentSchedule = ASAPSchedule;
(08)while(NonZeroMobilityVertexList is NOT empty)
   {
(09)    max = −∞; AllocateVoltAndFreq(CurrentSchedule, LUTs);
(10)    CurrentEDP = CalculateEDP (VoltageArray,FrequencyArray);
(11)    for each v_i ∈ NonZeroMobilityVertexList
        {
(12)       start = CurrentSchedule[v_i]; end = ALAPSchedule[v_i];
(13)       for cycle = start → end in steps of 1
           {
(14)          TempSchedule = AdjustSchedule(CurrentSchedule, v_i, cycle);
(15)          AllocateVoltAndFreq(TempSchedule, LUTs);
(16)          TempEDP = CalculateEDP(VoltageArray,FrequencyArray);
(16)          ExtraEDP = CurrentEDP − TempEDP;
(17)          if(ExtraEDP > max)
              {
(18)             max = ExtraEDP; CurrentVertex = v_i;
(19)             CurrentCycle = cycle;
              } // end if (17)
           } // end for (13)
        } // end for (11)
(20)    CurrentSchedule = AdjustSchedule(CurrentSchedule, CurrentVertex, Currentcycle);
(21)    Update the "resource assignment LUT";
(22)    ZeroMobilityVertexList = ZeroMobilityVertexList ∪ CurrentVertex;
(23)    NonZeroMobilityVertexList = NonZeroMobilityVertexList − CurrentVertex;
   }//end while(08)
(24)AllocateVoltAndFreq(CurrentSchedule, LUTs);
(25)EnergyAndDelayDetails(VoltageArray, FrequencyArray);
                        FindCycleFreqIndex(FrequencyArray);
} // End Algorithm RC-DFC
```

Figure 3.7. Pseudo-code for RC-DFC Scheduler

Table 3.10. List of Variables and Data Structures used in the RC-DFC Algorithm Description

Data Structures	Descriptions
ASAPSchedule	: An array used to store ASAP time stamp of each vertex.
ALAPSchedule	: An array used to store ALAP time stamp of each vertex.
CurrentSchedule	: An array used to store current schedule time stamp.
TempSchedule	: An array used to store temporary schedule time stamp.
MULT	: Number of multipliers at all voltage levels.
ALU	: Number of ALUs at all voltage levels.
ASAPControlSteps	: Total number of control steps of ASAP schedule.
ALAPControlSteps	: Total number of control steps of ALAP schedule.
NoOfControlSteps	: Number of control steps of the schedule.
ResAssignmentLUT	: Resource assignment look-up table.
FreqSelectionLUT	: Frequency selection look-up table.
max, start, end, cycle	: Temporary variables.
CurrentEDP, TempEDP, ExtraEDP	: Temporary variables.
CurrentVertex, CurrentCycle	: Temporary variables.
VoltageArray	: An array used to store operating voltage for each vertex.
FrequencyArray	: An array used to store operating fequency for each cycle
ZeroMobilityVertexList	: An array storing the vertices with zero mobility.
NonZeroMobilityVertexList	: An array storing the vertices with non-zero mobility.

the number of ALUs is 3 and the number of multipliers is 4. Using the number of multipliers and ALUs found above as initial resource constraint (with relaxed voltage constraint), the *ModifySchedule* procedure (line 03) modifies the ASAP and ALAP schedules so that the resource constraints are not violated. In this process, the mobility of the vertices are restricted to great extent and the search space for the following steps reduces. Next, the total number of cycles for the schedule is assumed as the maximum of the number of cycles for the ASAP and ALAP schedules (line 04). The resource assignment LUT is constructed (similar to Table 3.8) in line 05 whose size depends on *(NoOfControlSteps * NoOfResourceTypes)*. The procedure *FindResTypeForEachVertex* (line 06) identifies the functional unit(s) required at each vertex of the DFG. In line 06, frequency selection LUT similar to Table 3.7 is constructed. The *FindMobileVertexList* procedure (line 07) takes as input the modified ASAP and the modified ALAP schedules (line 04) to determine two lists: the list, *ZeroMobilityVertexList*, containing the vertices with zero mobility (same ASAP and ALAP

time stamps) and another, *NonZeroMobilityVertexList*, containing the non-zero mobility vertices (different ASAP and ALAP time stamps).

In line 07, the *CurrentSchedule* is initialized as the modified ASAP schedule (obtained in line 03). The procedure *AllocateVoltAndFreq* (lines 09 and 24) allocates the voltage levels and frequency levels to the FU's using the LUTs and the current schedule. This procedure returns two lists: one containing the assigned voltage of each vertex *(VoltageArray)* and the other *(FrequencyArray)* containing the selected frequency. *FrequencyArray* is in turn used to derive the cfi_c for the control steps. The procedure *CalculateEDP* (line 10) the energy delay product of the whole DFG using a schedule with voltage assignment stored in *VoltageArray* and frequency contained in *FrequencyArray*. The procedure *AdjustSchedule* (lines 14 and 20) schedules each vertex to a specific cycle while adjusting its predecessor and successor time stamps. The for loop (lines 11 to 19) considers all the vertices from the *NonZeroMobilityVertexList* and finds a suitable vertex and its time stamp such that the energy delay product of the whole DFG with current schedule is minimum. In line 21, resource assignment LUT is updated. The while loop (lines 08 to 23) terminates when all the vertices with non-zero mobility have been assigned the proper time stamp. The procedure *FindEnergyAndDelay* (line 25) determines the energy consumption and execution time for the schedule. Line 25, *FindCycleFreqIndex* finds cycles frequency indices of all cycles which is going to help in dynamic frequency generation. Figure 3.8 is obtained after executing the RC-DFC algorithm for the resource constraint (one MULT at $2.4V$, one MULT $3.3V$, one ALU at $3.3V$ and one ALU at $5.0V$).

3.3.3 Time Complexity

Let there be $|V|$ number of vertices and $|E|$ number edges in the DFG, out of which $|V_m|$ number of vertices have mobility and the maximum mobility of any mobile vertex is t_m. Let L_v denote the number of voltage levels and L_f denote the number of frequency levels. Suppose the number of control steps found out from the ASAP scheduling is c. Assuming that $L_V$and L_f are upper bounded by $|V|$, the running time of the code segment from line 01-07 is $\Theta(|V| + |E|)$ $+\Theta(cL_V R_T)$. The time-complexity of the instruction in line 11-19 is $\Theta(c|V|L_V R_T|V_m|t_m)$.

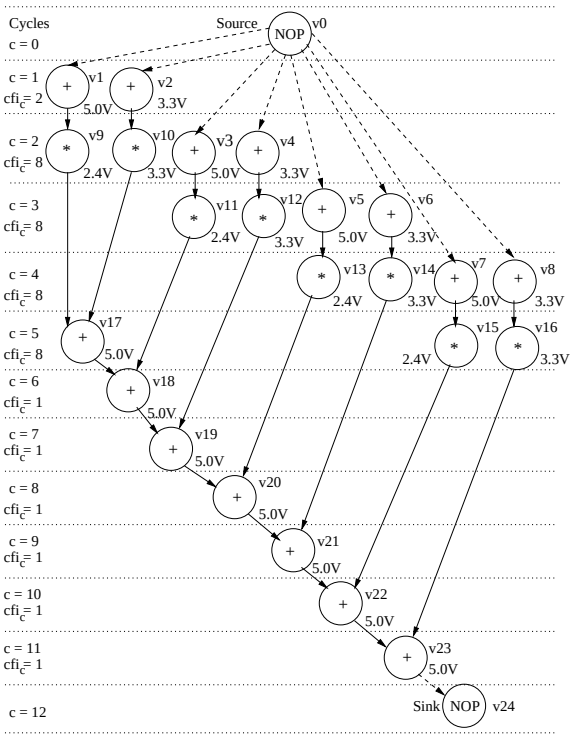


Figure 3.8. Final Schedule of FIR Filter DFG (using RC-DFC)

The code-segment line 09 to 19 has running time $\Theta\left(c\left|V\right|L_{V}R_{T}\left|V_{m}\right|t_{m}\right)+\Theta\left(\left|V\right|\right)+\Theta\left(c\left|V\right|L_{V}R_{T}\right)$ $=\Theta\left(c\left|V\right|L_{V}R_{T}\left|V_{m}\right|t_{m}\right)$. The running time of the code segment line08-19 is $\Theta\left(c\left|V\right|L_{V}R_{T}|V_{m}|^{2}t_{m}\right)$. The time complexity of line 20-25 is $\Theta\left(\left|V\right|\right)+\Theta\left(c\left|V\right|L_{V}R_{T}\right)+\Theta\left(c\right)=\Theta\left(c\left|V\right|L_{V}R_{T}\right)$. So, the running time of the overall algorithm is $\Theta\left(\left|V\right|+\left|E\right|\right)+\Theta\left(cL_{V}R_{T}\right)+\Theta\left(c\left|V\right|L_{V}R_{T}|V_{m}|^{2}t_{m}\right)$ $+\Theta\left(c\left|V\right|L_{V}R_{T}\right)=\Theta\left(\left|V\right|+\left|E\right|\right)+\Theta\left(c\left|V\right|L_{V}R_{T}|V_{m}|^{2}t_{m}\right)$. Assuming that $|E|$ is upper bounded by $|V|^{2}$ and $|V_{m}|$ is upper boounded by $|V|$, the above expression can be simplified to $O\left(c|V|^{3}L_{V}R_{T}t_{m}\right)$.

3.4 Experimental Results

Both RC-DFC and TC-DFC schedulers were implemented in C and tested with selected benchmark circuits. The benchmarks used are :

- Auto-Regressive (ARF) filter [162]
- Band-Pass filter (BPF) [27]

- Elliptic-Wave filter (EWF) [163]
- DCT [164]
- FIR filter [91]
- HAL differential equation solver [21].

The FUs used are ALUs and multipliers. The energy values are computed using the datapath components given in [54, 55]. The following notations are used to express the results :

- E_S and E_D are the total energy consumption (in pJ) for single supply voltage and multiple supply voltage operations respectively.
- EDP_S and EDP_D are the energy-delay-products (in $10^{-18}J - s$) for single supply voltage and single frequency and for multiple supply voltage and dynamic clocking operations respectively.
- T_S and T_D are the corresponding delays (in ns) for the two modes of operations.
- N_S denotes the number of clock steps of the schedule for single supply voltage and and single frequency operations.
- N_D is the equivalent clock steps of T_D found out taking the delay of slowest functional unit as the base clock width in case of multiple voltage operation.
- The percentage energy savings is calculated as, $\Delta E = \frac{(E_S - E_D)}{E_S} * 100$. In similar manner, we calculated percentage reduction in EDP which is denoted as ΔEDP.

For RC-DFC scheduler, the experimental set-up is as follows. The algorithm was tested using the different sets of resource constraints listed in Table 3.11. The experimental results for various benchmark circuits are reported in Table 3.12. The energy estimation includes the energy consumption of the overhead units. It is assumed that each resource has equal switching activity. The results are reported for two supply voltage and for switching $= 0.5$. It is obsorved that the energy consmption is increased for higher switching and decreased for lower switching activity,

Table 3.11. Resource Constraints used in our Experiements

Resource Constraints				Assigned Serial No. (RC)
Multipliers		ALUs		
3.3 V	5.0 V	3.3 V	5.0 V	
2	1	1	1	1
3	0	1	1	2
2	0	0	2	3
1	1	0	2	4

but, under the assumption that switching is same for each resource, the percentage energy savings is not affected. We also conducted experiments with three supply voltage levels and it is found that the percentage energy savings could only increase by 5%. Fig. 3.9(a) shows the percentage savings (average ΔE) averaged over all resource constraints. From the chart it is evident that the scheduling yields approximately equal savings for all kinds of benchmark circuits. The EDP reduction (average ΔEDP) averaged over all resource constraints are shown in Fig. 3.9(c). From the above, we may conclude that the scheduling algorithm yields appreciable energy savings and EDP reduction. In order to find the right combination of the types and the number of resources that will yield the best results in terms of energy reduction and high performance, we plotted energy consumption (%) versus time ratio ($\frac{T_D}{T_S}$), which is nothing but the the configuration correspoding to maximum ΔEDP. Based on this analysis, the processor configurations that yield the lowest execution time for each benchmark is listed in Table 3.13.

The TC-DFC scheduler was tested for three different time constraints: 1.5, 1.75 and 2.0 times critical path delay (T_{cp}). The voltage constraint is relaxed unlike the RC-DFC. The results for various benchmark circuits are reported in Table 3.14. Fig. 3.9(b) shows the chart indicating the energy savings for different benchmarks averaged over all time constraints. Our observation is that circuits which require equal number of ALUs related operations (addition, subtraction or comparison) and multiplier operations save more energy. The energy savings increased as the time constraints relaxed from $1.5T_{cp}$ to $2.0T_{cp}$.

The energy savings from the proposed RC-DFC scheduling algorithm is listed alongwith other resource constrained multiple voltage scheduling algorithms in Table 3.15. The minimum and

Table 3.12. Energy Details for Different Benchmarks (for $\alpha = 0.5$) using RC-DFC Scheduler

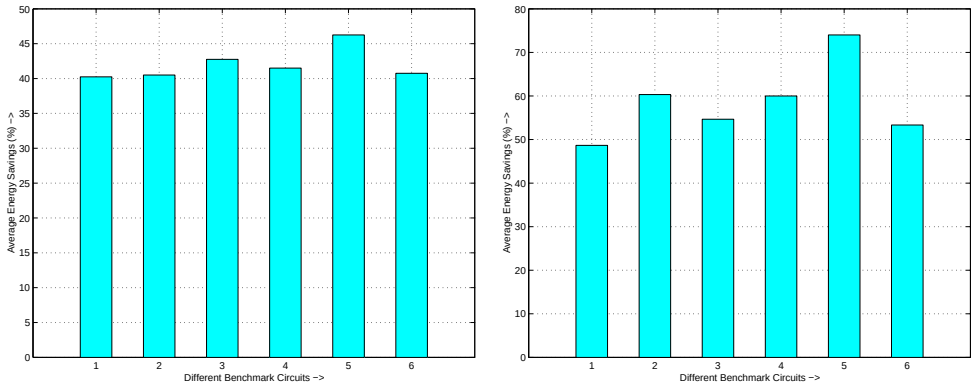
	RC	Energy Estimates (pJ)			Energy-Delay-Product ($10^{-18}Js$)			Time Estimates (ns or cycles)			
		E_S	E_D	ΔE	EDP_S	EDP_D	ΔEDP	N_S	T_S	T_D	N_D
(1)	1	36168	21768	40	20093	19954	1	10	556	917	9
A	2	36168	18205	50	20093	16688	17	10	556	917	9
R	3	36168	19065	47	20093	18006	10	10	556	944	9
F	4	36168	27617	24	26121	31452	NA	13	722	1139	10
	Average Data			40.3			7.0				
(2)	1	27654	16491	40	13827	14659	NA	9	500	889	8
B	2	27654	14175	49	13827	12600	9	9	500	889	8
P	3	27654	14827	46	13827	12356	11	9	500	833	8
F	4	27654	20172	27	26118	23253	11	17	944	1153	10
	Average Data			40.5			7.8				
(3)	1	19404	10802	44	17248	12902	25	16	889	1194	11
E	2	19404	10802	44	17248	12902	25	16	889	1194	11
W	3	19404	10853	44	17248	11154	35	16	889	1028	10
F	4	19404	11922	39	29106	17055	41	27	1500	1431	12
	Average Data			42.8			31.5				
(4)	1	30675	17846	42	25547	26274	NA	15	833	1472	14
D	2	30675	17846	42	25547	26274	NA	15	833	1472	14
C	3	30675	18008	41	25548	25511	0	15	833	1416	13
T	4	30675	18008	41	49392	37267	25	29	1611	2069	17
	Average Data			41.5			6.3				
(5)	1	18678	9979	47	11414	6653	42	11	611	667	7
F	2	18678	9979	47	11414	6653	42	11	611	667	7
I	3	18678	10126	45	11414	6470	43	11	611	639	6
R	4	18678	10127	46	15565	12096	22	15	833	1194	10
	Average Data			46.3			37.3				
(6)	1	13596	8927	34	3021	2728	10	4	222	306	3
H	2	13596	6433	53	3021	1966	35	4	222	306	3
A	3	13596	6648	51	3021	2401	21	4	222	361	4
L	4	13596	10211	25	3777	4396	NA	5	278	431	4
	Average Data			40.8			16.5				
Overall Average Data				42.0			17.7				

Table 3.13. Configurations for Minimum EDP using RC-DFC

Bench-mark Circuits	Processor Configurations			
	Multipliers		ALUs	
	3.3 V	5.0 V	3.3 V	5.0 V
AR	3	0	1	1
BPF	2	0	0	1
EWF	2	0	0	1
DCT	1	1	0	1
FIR	2	0	0	2
HAL	3	0	1	1

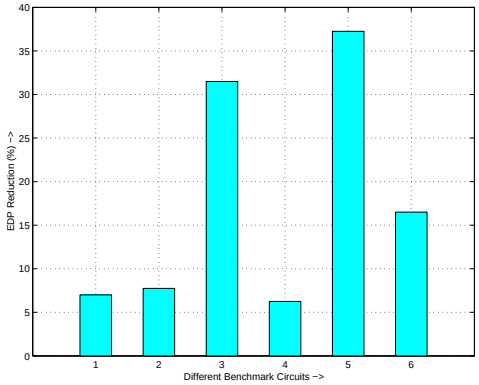
Table 3.14. Energy Savings using TC-DFC Scheduler

Bench. Circuits	Time Cons.	Energy consumption and savings		
		$E_S(pJ)$	$E_D(pJ)$	$\Delta E(\%)$
(1) ARF	$1.5T_{cp}$	36186	21491	41
	$1.75T_{cp}$	36186	18139	47
	$2.0T_{cp}$	36186	15274	58
	Average Data			48.67
(2) BPF	$1.5T_{cp}$	27672	15187	45
	$1.75T_{cp}$	27672	9350	66
	$2.0T_{cp}$	27672	8249	70
	Average Data			60.33
(3) EWF	$1.5T_{cp}$	19422	12335	36
	$1.75T_{cp}$	19422	8814	55
	$2.0T_{cp}$	19422	5341	73
	Average Data			54.67
(4) DCT	$1.5T_{cp}$	30675	14611	52
	$1.75T_{cp}$	30675	14489	53
	$2.0T_{cp}$	30675	7714	75
	Average Data			60.0
(5) FIR	$1.5T_{cp}$	18696	4910	74
	$1.75T_{cp}$	18696	4877	74
	$2.0T_{cp}$	18696	4820	74
	Average Data			74.0
(6) HAL	$1.50T_{cp}$	13614	7808	43
	$1.75T_{cp}$	13614	6821	50
	$2.0T_{cp}$	13614	4449	67
	Average Data			53.33
Overall Average Data				58.50



(a) Energy Reduction for RC-DFC

(b) Energy Reduction for TC-DFC



(c) EDP Reduction for RC-DFC

Figure 3.9. Average Energy and EDP Reduction for Benchmarks

maximum range of energy savings are shown in the table. As clear from column (15) of Table 3.12, RC-DFC gives better energy savings for lesser time penalties. The energy savings obtained using different existing multiple voltage based time-constraints scheduling algorithm is shown in Table 3.16. In all cases, the time constraints are $1.5 * T_{cp}$ to $2.0 * T_{cp}$.

3.5 Conclusions

Our aim is to use frequency scaling concepts for energy-efficient high-performance special propose processor (ASIC) design. The energy reduction is achieved by voltage reduction and the performance is maintained by using DFC alongwith multiple voltages. We developed resource-

Table 3.15. Savings for Various Resource Constrained Schedulings

Ben. mark Ckt	% Energy savings and time penalties (T) in cycles							
	RC-DFC		Shiue[95]		Sarrafzadeh[90]		Johnson[65]	
	ΔE	N_D	ΔE	T	ΔE	T	ΔE	T
ARF	24-58	9-10	11-14	11-16	16-20	17-24	16-59	10-18
BPF	27-56	8-10	-	-	-	-	-	-
EWF	38-61	10-13	14-14	17-20	13-32	21-25	11-50	12-24
DCT	41-63	13-18	-	-	-	-	-	-
FIR	20-67	6-10	-	-	16-29	10-15	28-73	5-10
HAL	29-62	2-3	19-28	5-6	-	-	-	-

Table 3.16. Savings for Various Time Constrained Schedulings

Bench-marks	% Energy savings			
	TC-DFC	Chang[51]	Shiue[95]	Manzak[97]
AR	41-58	40-63	38-76	25-61
BPF	45-70	-	-	-
EWF	36-73	44-69	13-76	10-55
FDCT	52-75	43-69	-	-
FIR	74-74	-	-	-
HAL	43-67	41-61	22-77	19-62

constrained and time-constrained datapath scheduling algorithms based on dynamic frequency clocking. The use of dynamic frequency clocking could generate enough slack to apply reduced voltages which in turn saves energy. It is observed that when using two supply voltage levels an average energy reduction of 41% and for three supply voltage levels, an average reduction of 46% is obtained for the benchmarks using the RC-DFC algorithm. Similarly, for TC-DFC, an average energy reduction of 46% (for $1.5 \times T_{cp}$) and 68% (for $2.0 \times T_{cp}$) are obtained. The processor configurations for various benchmark circuits that would result minimum energy-delay-product were determined through experiments. The integration of such a scheduler into a low power datapath synthesis tool will significantly benefit low power processor design especially for data intensive applications.

CHAPTER 4

ENERGY DELAY PRODUCT MINIMIZATION

In this chapter, we describe an integer linear programming (ILP) based datapath scheduling algorithm which incorporates multiple supply voltages and dynamic frequency clocking (MVDFC) for energy reduction [64]. The scheduling technique assumes the number and type of different functional units as resource constraints and minimizes the energy delay product (EDP). The energy savings is from the use of multiple supply voltages while the performance improvement from dynamic frequency clocking. Further, we consider the simultaneous use of multiple supply voltages and multicyling (MVMC) to achieve reduction in energy and energy delay product. Both the MVDFC and MVMC based schemes have been applied to various high level synthesis benchmark circuits under different resource constraints. The experimental results show appreciable reductions in both energy and energy delay product.

This chapter is organized as follows. We first outline the related works proposed in the literature. Then we provide the ILP-formulations to minimize the energy delay product. The next section discusses the ILP-based scheduler, followed by experimental results.

4.1 Energy Delay Product of a Datapath Circuit

A CMOS circuit can be operated in different modes, namely, single supply voltage and single frequency, multiple supply voltages and single frequency, and multiple supply voltages and dynamic frequency. Traditionally, CMOS circuits are operated in the single supply voltage and single frequency mode, in which, during each cycle the clock width is dictated by the slowest operator delay and each functional unit is operated at equal voltage level. In multiple supply voltages and single frequency mode, different functional units are operated at different voltage levels to reduce energy consumption [65, 51, 89]. In this case, energy consumption of the level converters is to be

taken into account. More recently, multiple supply voltages and dynamic frequency clocking mode of operation is being explored as a possible strategy for low power high performance operation. In dynamic frequency clocking, the clock frequency is varied on-the-fly based on the functional unit active in that cycle. In this scheme, all the units are clocked by single clock line which switches at run time. This scheme, in particular, is suitable for data intensive or compute intensive, DSP applications. The architecture for dynamic clocking based systems consists of a datapath, a controller and a dynamic clocking unit (DCU). The datapath consists of funtional units with registers and multiplexors. The controller decides which functional units are active in each control step and those not active are disabled using a multiplexor. The DCU generates the required clock frequency usually using clock divider strategy [59, 62] which are submultiples of base frequency. The base frequency is the maximum frequency (or multiple of maximum) of any functional unit at maximum supply voltage. The controller has storage units to store a parameter called, "clock frequency index" ([55]) for each control step derived during the datapath scheduling. This clock frequency index parameter serves as the clock dividing factor for the DCU. The cycle frequency is generated dynamically and the functional units with the appropriate supply voltages are activated. The main overheads in this scheme are, level converters, the dynamic clocking unit, and some additional storage in the control unit. When a value of cfi_c is loaded into the DCU, the DCU provides a divided output clock frequency, $\frac{f_{base}}{cfi_c}$.

Let us assume that the datapath is represented as a sequencing data flow graph. We use the notations given in Table 4.1 for developing the following energy and energy delay product for a datapath. The energy consumption in any cycle c is the energy consumption of all the resources active in c, which is given as,

$$E_c = \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 \tag{4.1}$$

The level converters are considered as resources operating in the control step in which it needs to step up the signal. The total energy consumption of the whole DFG (or datapath) is the sum of the

Table 4.1. Notations used in Description

O	: total number of operations in the DFG excluding the source and sink nodes (NO-OPs)
o_i	: any operation such that $1 \leq i \leq O$
N	: total number of control steps in the DFG
c	: any control step or clock cycle in DFG
R_c	: number of resources active in step c
f_c	: cycle frequency for control step c
$\alpha_{i,c}$	: switching at resource i used by operation o_i operating in step c
$C_{i,c}$	: load capacitance of resource i used by operation o_i operating in control step c
$V_{i,c}$	: operating voltage of resource i used by operation o_i operating in control step c
E_c	: energy consumption of all functional units active in cycle c
EDP_c	: energy delay product of all functional units active in cycle c
T	: critical path delay of the DFG
E	: total energy consumption of the DFG
EDP	: total energy delay product of the DFG
S	: subscript used for single supply voltage and single frequency operation
D	: subscript used for miltiple supply voltage and dynamic frequency operation
M	: subscript used for miltiple supply voltage and multicycling operation
f_{clk}	: operating clock frequency for single frequncy or multicycling opeartions

energy consumption for all cycles as given in Eqn. 4.2 below.

$$\begin{aligned} E_D &= \sum_{c=1}^{N} E_c \\ &= \sum_{c=1}^{N} \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 \end{aligned} \tag{4.2}$$

The dynamic clocking unit (DCU) is responsible for generating dynamic clock is considered as a resource operating in all the control steps. The energy consumptions of the DCU is to be added alongwith Eqn. 4.2, but need not be considered for minimization.

The critical path delay of the DFG is given by the summation of the inverse of the clock frequencies.

$$T_D = \sum_{c=1}^{N} f_c \tag{4.3}$$

The total energy delay product can be calculated as the product of the total energy consumption and the critical path delay as shown in the following equation.

$$\begin{aligned} EDP_D &= E_D * T_D \\ &= \left(\sum_{c=1}^{N} \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2\right) * \sum_{c=1}^{N} f_c \end{aligned} \tag{4.4}$$

This should be the objective function for the scheduling algorithm for minimization.

We are aiming at minimizing both the voltage and frequency. Since the objective function involves the product of the two variables, and is a non-linear function, we can not use integer linear programming (ILP) for its minimization. Hence, in stead of finding the energy consumption for each cycle c as in Eqn. 4.1, we derive the energy delay product for each cycle.

$$\begin{aligned} EDP_c &= \frac{E_c}{f_c} \\ &= \frac{\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2}{f_c} \end{aligned} \tag{4.5}$$

The total energy delay product of the DFG is the sum of above EDP_c for all control steps which is given as follows.

$$\begin{aligned} EDP_D &= \sum_{c=1}^{N} EDP_c \\ &= \sum_{c=1}^{N} \frac{\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2}{f_c} \\ &= \sum_{c=1}^{N} \sum_{i=1}^{R_c} \frac{\alpha_{i,c} C_{i,c} V_{i,c}^2}{f_c} \end{aligned} \tag{4.6}$$

For single voltage and single frequency mode of operation, $V_{i,c}$ and f_c are the same for any clock cycle (c) and any operation (i). However, for multiple supply voltage and multicycling operation, f_c is the same for all control steps and let us denote it as f_{clk}. Following the same steps as above the total energy delay product of the DFG for multiple supply voltage and multicycling operation is given by the following equation.

$$\begin{aligned} EDP_M &= \sum_{c=1}^{N} EDP_c \\ &= \sum_{c=1}^{N} \frac{\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2}{f_{clk}} \\ &= \sum_{c=1}^{N} \sum_{i=1}^{R_c} \frac{\alpha_{i,c} C_{i,c} V_{i,c}^2}{f_{clk}} \end{aligned} \tag{4.7}$$

4.2 ILP Formulations

In this section, we discuss the ILP formulations to minimize the peak and average power consumption of a datapath circuit. We first discuss the formulations for multiple supply voltages and dynamic clocking based system followed by multiple supply voltages and multicycling based system. In order to formulate an ILP based model for the objective function and the scheduling scheme for the DFG, the notations given in Table 4.2 are required.

Table 4.2. Notations used in ILP Formulations

$F_{k,v}$	: functional unit of type k operating at voltage level v
$M_{k,v}$	: maximum number of functional units of type k operating at voltage level v
S_i	: as soon as possible (ASAP) time stamp for the operation o_i
E_i	: as late as possible (ALAP) time stamp for the operation o_i
$EDP(i,v,f)$	: energy delay product of functional unit used by operation o_i operating at voltage level v and frequency f
$x_{i,c,v,f}$	: decision variable which takes the value of 1 if operation o_i is scheduled in control step c using the functional unit $F_{k,v}$ and c has frequency f_c
$y_{i,v,l,m}$	: decision variable which takes the value of 1 if o_i is using the functional unit $F_{k,v}$ and scheduled in control steps $l \rightarrow m$
$L_{i,v}$	: latency for operation o_i using resource operating at voltage v (in terms of number of clock cycles)

4.2.1 ILP Formulations : Dynamic Frequency Clocking

First, we derive the ILP formulation for the objective function given in Eqn. 4.6 for multiple supply voltages and dynamic clocking frequency.

Objective Function : The objective function minimizes the total energy delay product of the entire DFG. Using the decision variable $x_{i,c,v,f}$, we write the objective function as follows.

$$\begin{aligned} Minimize &: EDP_D \\ Minimize &: \sum_c \sum_i \sum_c \sum_v \sum_f x_{i,c,v,f} * EDP(i,v,f) \end{aligned} \tag{4.8}$$

Uniqueness Constraints : These constraints ensure that each operation o_i is scheduled to an unique control step within the mobility range (S_i, E_i) with a particular supply voltage and operating frequency. We represent them as, $\forall i$, $1 \leq i \leq N$,

$$\sum_c \sum_v \sum_f x_{i,c,v,f} \quad = \quad 1 \tag{4.9}$$

Precedence Constraints : These constraints guarantee that for an operation o_i, all its predecessors are scheduled in earlier control steps and its successors are scheduled in later control steps. These are modelled as, $\forall i, j, o_i \in Pred_{o_j}$,

$$\sum_v \sum_f \sum_{d=S_i}^{E_i} d * x_{i,d,v,f} \quad - \quad \sum_v \sum_f \sum_{e=S_j}^{E_j} e * x_{j,e,v,f} \quad \leq \quad -1 \tag{4.10}$$

Resource Constraints : The resource constraints make sure that no control step contains more than $FU_{k,v}$ operations of type k operating at voltage v. These can be enforced as, $\forall c$, $1 \leq c \leq N$ and $\forall v$,

$$\sum_i \sum_f x_{i,c,v,f} \quad \leq \quad M_{k,v} \tag{4.11}$$

Frequency Constraints : This set ensures that if a functional unit is operating at a higher voltage level then it can be schduled in a lower frequency control step, whereas, a functional unit operating at a lower voltage level then it can not be scheduled during a higher frequency control step. We write these constraints as, $\forall i$, $1 \leq i \leq N$, $\forall c$, $1 \leq c \leq N$, if $f < v$, then $x_{i,c,v,f} = 0$.

4.2.2 ILP Formulations : Multicycling

Now, we give the ILP formulation for the objective function given in Eqn. 4.7 for multiple supply voltages and multicycling operation mode.

Objective Function : The objective is to minimize the energy delay product of the whole DFG

over all control steps using multiple supply voltages and multicycling.

$$\begin{aligned} Minimize &: EDP_M \\ Minimize &: \sum_l \sum_i \sum_v y_{i,v,l,(l+L_{i,v}-1)} * EDP(i,v,f_{clk}) \end{aligned} \tag{4.12}$$

Uniqueness Constraints : These constraints ensure that each operation o_i is scheduled in the appropriate control step within the mobility range (S_i, E_i) begin assigned the specific supply voltage. An operation may be operated with more than one clock cycle sometimes depending on the supply voltage. These constraints are represented as, $\forall i$, $1 \leq i \leq O$,

$$\sum_v \sum_{l=S_i}^{S_i+E_i+1-L_{i,v}} y_{i,v,l,(l+L_{i,v}-1)} = 1 \tag{4.13}$$

When an operation is scheduled at the highest voltage, then it is scheduled in one unique control step, whereas, when they are to be operated at lower voltages they need more than one clock cycle for completion. Thus, for lower voltages the mobility is restricted.

Precedence Constraints : These constraints guarantee that for an operation o_i, all its predecessors are scheduled in earlier control steps and its successors are scheduled in later control steps. These constraints should also take care of the multicycling operations. These are modeled as, $\forall i, j, o_i \in Pred_{o_j}$,

$$\sum_v \sum_{l=S_i}^{E_i} (l + L_{i,v} - 1) * y_{i,v,l,(l+L_{i,v}-1)} - \sum_v \sum_{l=S_j}^{E_j} l * y_{j,v,l,(l+L_{j,v}-1)} \leq -1 \tag{4.14}$$

Resource Constraints : These constraints ensure that each control step contains no more than $F_{k,v}$ operations of type k operating at voltage v. This can be enforced as, $\forall v$ and $\forall l$, $1 \leq l \leq N$,

$$\sum_i \sum_l y_{i,v,l,(l+L_{i,v}-1)} \leq M_{k,v} \tag{4.15}$$

4.3 Datapath Scheduling Algorithm

In this section, we discuss the solution for the ILP formulations obtained in the previous section. The same target architecture and the same characterised datapath components used in [55] are assumed. The ILP based scheduler attempts to minimize the EDP is outlined in Fig. 4.1. The first step is to determine the ASAP and ALAP time stamp of each operation. The ASAP time stamp is the start time and ALAP time stamp is the finish time of each operation. These two times provide the mobility of a operation and the operation must be scheduled in this mobile range. Then the scheduler finds the ILP formulations based the models described in Section 4.2. The scheduler determines the cycle frequencies in step 6, which contribute the smallest frequencies of all operations scheduled in a particular cycle. Finally, we estimate the energy delay product and the energy consumptions of the whole DFG.

Step 1	: Determine the ASAP and ALAP schedules of the UDFG.
Step 2	: Determine the mobility graph of each node.
Step 3	: Construct the ILP formulations for the DFG.
Step 4	: Solve the ILP formulations using LP-Solve.
Step 5	: Find the scheduled DFG.
Step 6	: Determine the cycle frequencies.
Step 7	: Find the energy and EDP estimates of the DFG.

Figure 4.1. ILP Based Scheduling for Low EDP

4.3.1 Scheduling for MVDFC

We illustrate the solution for the ILP formulation in the MVDFC case, with the help of the DFG shown in Fig. 4.2. The ASAP schedule is shown in Fig. 4.2(a) and the ALAP schedule is shown in Fig. 4.2(b). From the ASAP and ALAP schedules, we obtain the mobility graph as in Fig. 4.2(c). Using this mobility graph, we have the ILP formulations shown in Fig. 4.3 for the resource constrain (RC2), three multipliers at $2.4V$, one ALU at $2.4V$, and one ALU operating at $3.3V$. We solved the formulations using LP-solve and based on the results, we obtained the scheduled DFG shown is Fig. 4.3(d). In Fig. 4.3, we used the following additional notations, $Mmult1$: number of

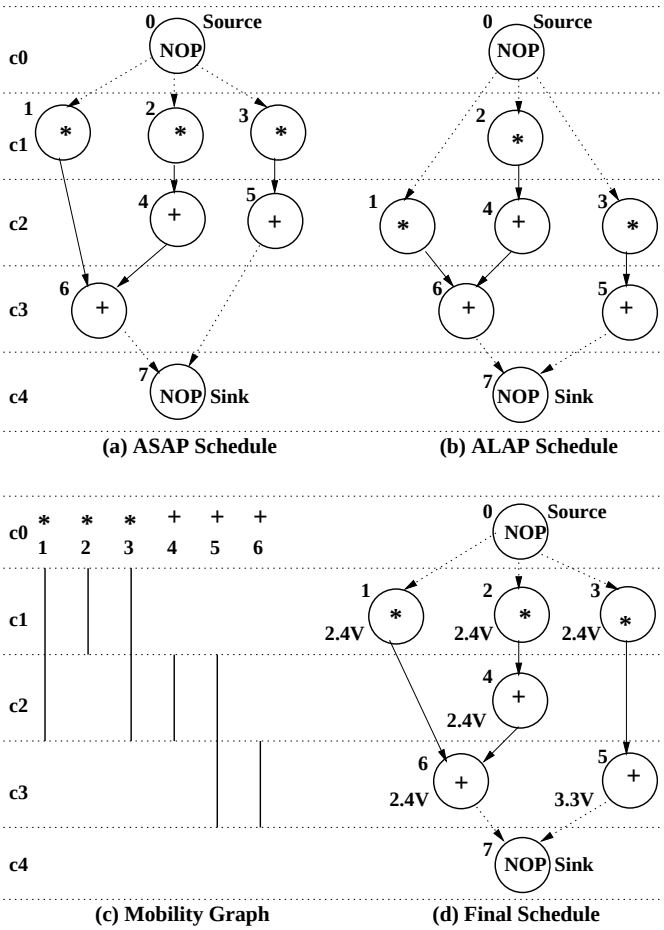


Figure 4.2. Example Data Flow Graph for Multiple Supply Voltages and Dynamic Frequency Clocking

multipliers at voltage level 1, $Mmult2$: number of multipliers at voltage level 2, $Malu1$: number of ALUs at voltage level 1, and $Malu2$: number of ALUs at voltage level 2.

4.3.2 Scheduling for MVMC

We illustrate the solution for the ILP formulation of the MVMC case, using the DFG shown in Fig. 4.4. The ASAP schedule is shown in Fig. 4.4(a) and the ALAP schedule is shown in Fig. 4.4(b). From the ASAP and ALAP schedules, we obtain the mobility graph shown in Fig.4.4(c). It should be noted that this mobility graph is different from that shown in Fig. 4.2(c). In the MVMC case, the mobility graph considers the multicycle operations. We assume two operating voltage levels, and when a multiplier is operated at the lower voltage level, it take two clock cycles for

```
/* ILP Formulation for Energy Delay Product Minimization for MVDFC scheme */


/* Objective Function */
min: 106.6 x1111 + 213.2 x1112 + 56.4 x1121 + 112.8 x1122 + 106.6 x1211 + 213.2 x1212
    + 56.4 x1221 + 112.8 x1222 + 106.6 x2111 + 213.2 x2112 + 56.4 x2121 + 112.8 x2122
    + 106.6 x3111 + 213.2 x3112 + 56.4 x3121 + 112.8 x3122 + 106.6 x3211 + 213.2 x3212
    + 56.4 x3221 + 112.8 x3222 + 2.8 x4211 + 5.5 x4212 + 1.5 x4221 + 2.9 x4222 + 2.8 x5211
    + 5.5 x5212 + 1.5 x5221 + 2.9 x5222 + 2.8 x5311 + 5.5 x5312 + 1.5 x5321 + 2.9 x5322
    + 2.8 x6311 + 5.5 x6312 + 1.5 x6321 + 2.9 x6322;

/* Uniqueness Constraints */
x1111 + x1112 + x1121 + x1122 + x1211 + x1212 + x1221 + x1222 = 1;
x2111 + x2112 + x2121 + x2122 = 1;
x3111 + x3112 + x3121 + x3122 + x3211 + x3212 + x3221 + x3222= 1;
x4211 + x4212 + x4221 + x4222 = 1;
x5211 + x5212 + x5221 + x5222 + x5311 + x5312 + x5321 + x5322 = 1;
x6311 + x6312 + x6321 + x6322 = 1;

/* Precedence Constraints */
3 x6311 + 3 x6312 + 3 x6321 + 3 x6322 - 2 x1211 - 2 x1212 - 2 x1221 - 2 x1222 - x1111
    - x1112 - x1121 - x1122 ≥ 1;
2 x4211 + 2 x4212 + 2 x4221 + 2 x4222 - x2111 - x2112 - x2121 - x2122 ≥ 1;
3 x6311 + 3 x6312 + 3 x6321 + 3 x6322 - x4211 - x4212 - x4221 - x4222 ≥ 1;
3 x5311 + 3 x5312 + 3 x5321 + 3 x5322 + 2 x5211 + 2 x5212 + 2 x5221 + 2 x5222
    - 2 x3211 - 2 x3212 - 2 x3221 - 2 x3222 - x3111 - x3112 - x3121 - x3122 ≥ 1;

/* Resource Constraints */
x1111 + x2111 + x3111 + x1112 + x2112 + x3112 ≤ 0; /* mult1 */
x1121 + x2121 + x3121 + x1122 + x2122 + x3122 ≤ 3; /* mult2 */
x1211 + x3211 + x1212 + x3212 ≤ 0; /* mult1 */
x1221 + x3221 + x1222 + x3222 ≤ 3; /* mult2 */
x4211 + x5211 + x4212 + x5212 ≤ 1; /* alu1 */
x4221 + x5221 + x4222 + x5222 ≤ 1; /* alu2 */
x5311 + x6311 + x5312 + x6312 ≤ 1; /* alu1 */
x5321 + x6321 + x5322 + x6322 ≤ 1; /* alu2 */

/* Frequency Constraints */
x1121 = 0; x1221 = 0; x2121 = 0; x3121 = 0; x3221 = 0; x4221 = 0; x5221 = 0; x5321 = 0; x6321 = 0;

/* Zero-One Type Cast */
INT x1111, x1112, x1121, x1122, x1211, x1212, x1221, x1222, x2111, x2112, x2121, x2122, x3111,
    x3112, x3121, x3122, x3211, x3212, x3221, x3222, x4211, x4212, x4221, x4222, x5211,
    x5212, x5221, x5222, x5311, x5312, x5321, x5322, x6311, x6312, x6321, x6322;
```

Figure 4.3. ILP Formulation for Example DFG for Multiple Supply Voltages and Dynamic Frequency Clocking

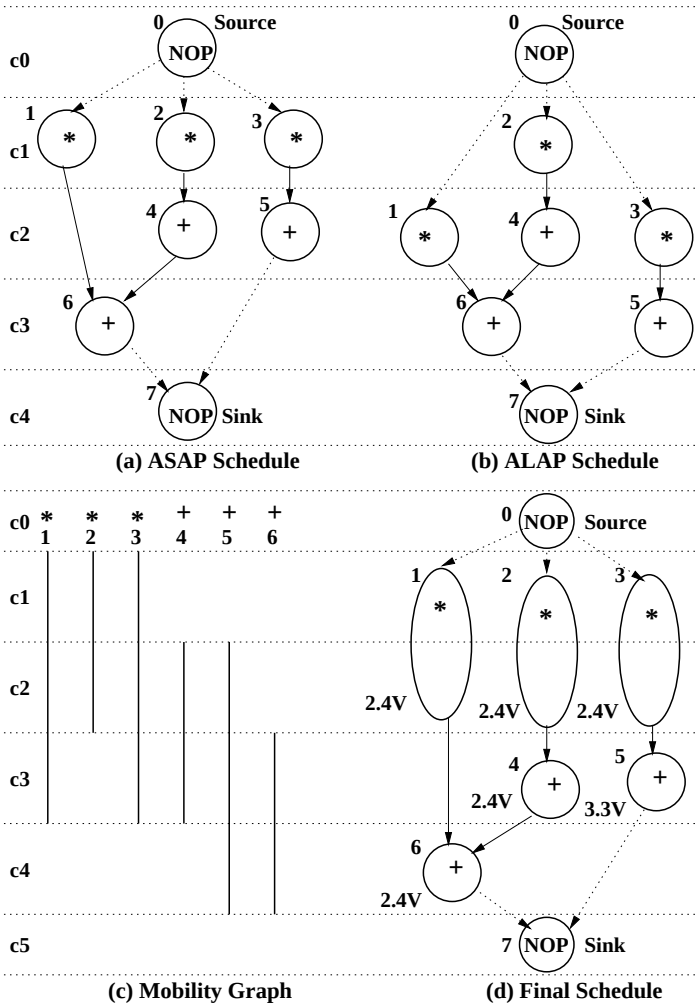


Figure 4.4. Example DFG (for RC2) (MVMC)

completing the operation. For the characterised cells used in our experiment [55], the operating clock frequency, f_{clk} is $9MHz$. Using this mobility graph, we have the ILP formulations shown in Fig. 4.3 for the resource constrain (RC2), three multipliers at $2.4V$, one ALU at $2.4V$, and one ALU operating at $3.3V$. We solved the formulation using LP-solve and based on the results we obtained the scheduled DFG shown is Fig. 4.2(d). In Fig. 4.5, the notations, such as, $Mmult1$, $Mmult2$, $Malu1$ and $Malu2$ are the same as those used in the case of the MVDFC.

```
/* ILP Formulation for Energy Delay Product Minimization for MVMC scheme */


/* Objective Function */
min: 106.6 x1111 + 106.6 x1122 + 106.6 x1133 + 56.4 x1212 + 56.4 x1223 + 106.6 x2111
    + 106.6 x2122 + 56.4 x2212 + 106.6 x3111 + 106.6 x3122 + 106.6 x3133 + 56.4 x3212
    + 56.4 x3223 + 2.8 x4122 + 2.8 x4133 + 1.5 x4222 + 1.5 x4233 + 2.8 x5122 + 2.8 x5133 + 2.8 x5144
    + 1.5 x5222 + 1.5 x5233 + 1.5 x5244 + 2.8 x6133 + 2.8 x6144 + 1.5 x6233 + 1.5 x6244;

/*Uniqueness Constraints*/
x1111 + x1122 + x1133 + x1212 + x1223 = 1;
x2111 + x2122 + x2212 = 1;
x3111 + x3122 + x3133 + x3212 + x3223 = 1;
x4122 + x4133 + x4222 + x4233 = 1;
x5122 + x5133 + x5144 + x5222 + x5233 + x5244 = 1;
x6133 + x6144 + x6233 + x6244 = 1;

/* Resource Constraints */
x1111 + x2111 + x3111 ≤ 0; /* Mmult1 */
x1212 + x2212 + x3212 ≤ 3; /* Mmult2 */
x1122 + x2122 + x3122 ≤ 0; /* Mmult1 */
x1212 + x1223 + x2212 + x3212 + x3223 ≤ 3; /* Mmult2 */
x1133 + x3133 ≤ 0; /* Mmult1 */
x1223 + x3223 ≤ 3; /* Mmult2 */
x4122 + x5122 ≤ 1; /* Malu1 */
x4222 + x5222 ≤ 1; /* Malu2 */
x4133 + x5133 + x6133 ≤ 1; /* Malu1 */
x4233 + x5233 + x6233 ≤ 1; /* Malu2 */
x5144 + x6144 ≤ 1; /* Malu1 */
x5244 + x6244 ≤ 1; /* Malu2 */

/* Precedence Constraints */
4 x6144 + 4 x6244 + 3 x6133 + 3 x6233 - 3 x1133 - 3 x1223 - 2 x1122 - 2 x1212 - x1111 ≥ 1;
4 x6144 + 4 x6244 + 3 x6133 + 3 x6233 - 3 x4133 - 3 x4233 - 2 x4122 - 2 x4222 ≥ 1;
3 x4133 + 3 x4233 + 2 x4122 + 2 x4222 - 2 x2122 - 2 x2212 - x2111 ≥ 1;
4 x5144 + 4 x5244 + 3 x5133 + 3 x5233 + 2 x5122 + 2 x5222 - 3 x3133
    - 3 x3223 - 2 x3122 - 2 x3212 - x3111 ≥ 1;

/* Integer Constraints */
INT x1111, x1122, x1133, x1212, x1223, x2111, x2122, x2212, x3111, x3122, x3133,
    x3212, x3223, x4122, x4133, x4222, x4233, x5122, x5133, x5144, x5222, x5233,
    x5244, x6133, x6144, x6233, x6244;
```

Figure 4.5. ILP Formulation for Example DFG for Multiple Supply Voltages and Multicycling

4.4 Experimental Results

We tested the ILP scheduler with selected benchmark circuits, such as, (1) Example circuit, (2) FIR filter, (3) IIR filter, (4) HAL differential equation solver and (5) Auto regressive filter. The functional units (FUs) assumed are ALUs and MULTs. The datapath cells and their characterization are considered from [55]. The following notations are used to express results :

- E_S, E_M and E_D represent the total energy consumption (in pJ) for single supply voltage, MVDFC and MVMC operations respectively.
- EDP_S, EDP_M and EDP_D are the energy-delay-products (in $10^{-18}J-s$) for single supply voltage and single frequency, for multiple supply voltage and single frequency and for multiple supply voltage and dynamic clocking operations, respectively.
- Rhe percentage energy savings is calculated as, $\Delta E_M = \frac{(E_S - E_M)}{E_S} * 100$ and $\Delta E_D = \frac{(E_S - E_D)}{E_S} * 100$.
- The percentage EDP reduction ΔEDP_M is calculated as, $\Delta EDP_M = \frac{(EDP_S - EDP_M)}{EDP_S} * 100$ and $\Delta EDP_D = \frac{(EDP_S - EDP_D)}{EDP_S} * 100$.

The datapath scheduling algorithms were tested using the different sets of resource constraints listed below.

(RC1)	multipliers (2 at $2.4V$ and 1 at $3.3V$)	and ALUs (1 at $2.4V$ and 1 at $3.3V$)
(RC2)	multipliers (3 at $2.4V$)	and ALUs (1 at $2.4V$ and 1 at $3.3V$)
(RC3)	multipliers (2 at $2.4V$)	and ALUs (2 at $3.3V$)
(RC4)	multipliers (2 at $2.4V$)	and ALUs (1 at $3.3V$)

The experimental results for various benchmark circuits are reported in Table 4.3. Fig. 4.6 shows the results for the various benchmarks averaged over different resource constraints. The energy estimation includes the energy consumption of the overheads. The results reported are based on the assumption of two supply voltages and switching activity of 0.5. The energy savings for the proposed algorithm is listed alongwith other multiple voltage scheduling algorithms in Table 4.4.

Table 4.3. Energy and EDP Estimates for Benchmarks for MVDFC and MVMC Schemes

	RC	Energy Estimates (pJ)					Energy Delay Products ($10^{-18} Js$)				
		E_S	E_M	E_D	ΔE_M	ΔE_D	EDP_S	EDP_M	EDP_D	ΔEDP_M	ΔEDP_D
1	2	3	4	5	6	7	8	9	10	11	12
(1)	1	2955	2013	1572	31.9	46.8	985.0	894.7	873.3	9.2	11.3
E	2	2955	1572	1572	46.8	46.8	985.0	698.7	698.7	29.1	29.1
X	3	2955	1596	1596	46.0	46.0	985.0	886.7	798.0	10.0	19.0
P	4	2955	1596	1596	46.0	46.0	1313.3	886.7	886.7	32.5	32.5
	Average Reduction				42.7	46.4				20.2	23.0
(2)	1	4900	3040	2587	38.0	47.2	2722.2	2026.7	2299.6	25.6	15.5
F	2	4900	2587	2587	47.2	47.2	2722.2	1724.7	2012.1	36.6	26.1
I	3	4900	2635	2635	46.2	46.2	2722.2	2049.4	2049.4	24.7	24.7
R	4	4900	2635	2635	46.2	46.2	2722.2	2049.4	2049.4	24.7	24.7
	Average Reduction				44.4	46.7				27.9	22.8
(3)	1	4900	3958	3052	19.2	37.7	2177.8	2198.8	2373.8	NA	NA
I	2	4900	2587	2549	47.2	47.0	2177.8	1724.7	2021.4	20.8	7.2
I	3	4900	2635	2635	46.2	46.2	2722.2	2342.2	2049.4	14.0	24.7
R	4	4900	2635	2635	46.2	46.2	2722.2	2342.2	2049.4	14.0	24.7
	Average Reduction				39.7	44.3				12.2	18.9
(4)	1	5885	4013	3119	31.8	47.0	2615.6	2675.3	2425.9	NA	7.3
H	2	5885	3119	3107	47.0	47.2	2615.6	2079.3	2071.3	20.5	20.8
A	3	5885	3167	3167	46.2	46.2	2615.6	2463.2	2287.3	5.8	12.5
L	4	5885	3167	3167	46.2	46.2	3269.4	3319.3	2463.2	NA	24.7
	Average Reduction				42.8	46.6				6.6	16.3
(5)	1	5000	2639	2639	47.2	47.2	5555.6	3811.8	4398.3	31.4	20.8
A	2	5000	2639	2639	47.2	47.2	5555.6	3811.8	4398.3	31.4	20.8
R	3	5000	2735	2735	45.3	45.3	5555.6	6839.4	3798.6	NA	31.6
F	4	5000	2735	2735	45.3	45.3	5555.6	6839.4	3798.6	NA	31.6
	Average Reduction				46.3	46.3				15.7	26.2
Overall Average Reduction					43.2	46.1				16.5	21.4

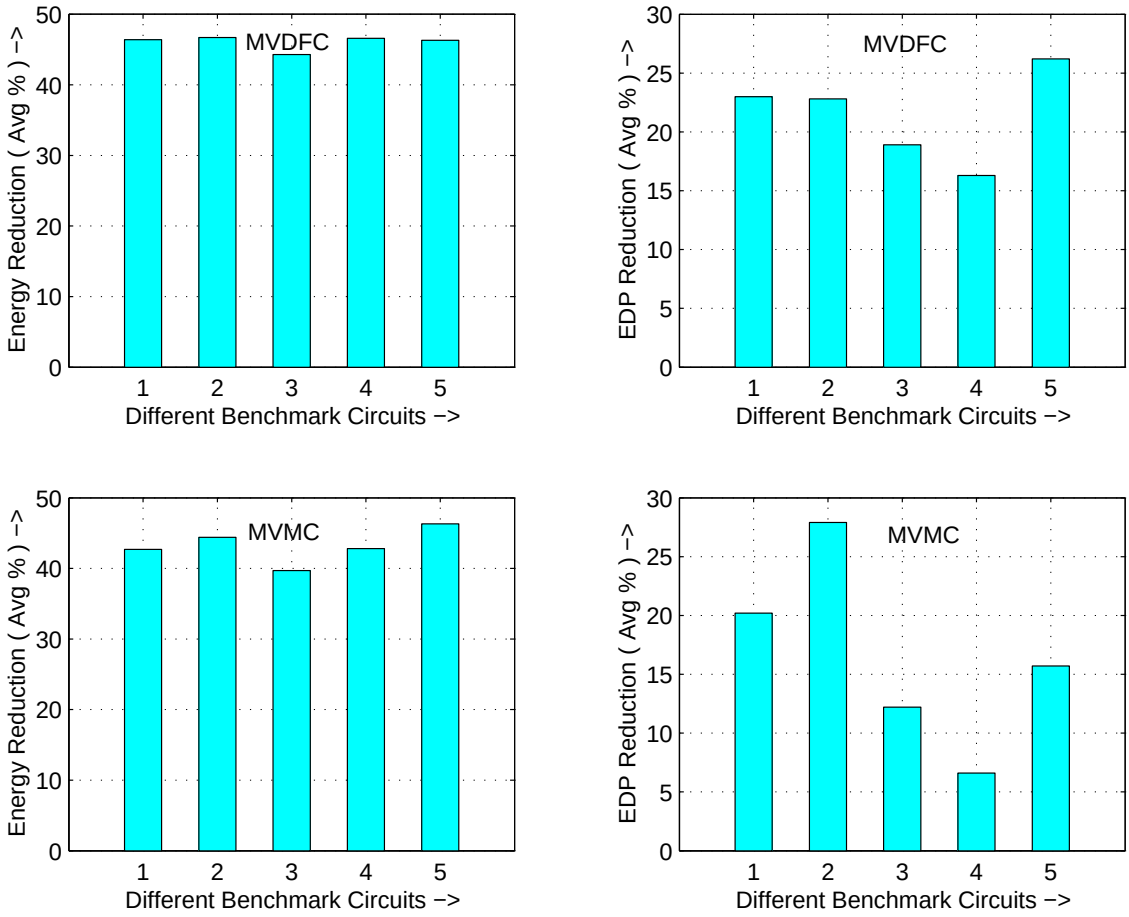


Figure 4.6. Reduction for Different Benchmarks Expressed as Percentage in Average

From the table, we observe that both the energy and the energy delay product are reduced considerably for both MVDFC and MVMC schemes. The MVDFC scheme results in better savings than due to that of the MVMC scheme for most of the cases, except the FIR benchmark. The energy savings of both the MVDFC and MVMC schemes are the same for most cases except for few resource constraints. The savings would have been same for both the schemes on using energy as objective function, as the energy savings is due to the voltage reduction, not due to the dynamic frequency clocking or multicycling. However, use of energy as objective function would have increased the energy delay product, thus reducing the performance.

Table 4.4. Savings for Various Schedulings Schemes

Bench-mark Circuits	% Average energy savings						
	This work		Shiue [95]	Sarrafzadeh [90]	Johnson [65]	Chang [51]	Mohanty [55]
	DFC	MC					
(2)fir	47	44	-	23	53	-	46
(3)iir	44	40	-	-	36	-	-
(4)hal	47	43	24	-	-	36	40
(5)arf	46	46	12	18	39	29	39

4.5 Conclusions

Our aim is to use frequency scaling concepts for energy-efficient high-performance ASIC design. The energy reduction is achieved through the use of voltage reduction and high-performance by using DFC. This chapter introduced a ILP based resource-constrained datapath scheduling algorithm using both multiple supply voltages and dynamic frequency clocking. It is observed that using two supply voltage levels, an average energy reduction of 46% and an average EDP reduction of 21% is obtained using MVDFC. Whereas, for MVMC scheme an average energy reduction of 43% and average EDP reduction of 16% is obtained. If in the critical path there are proportionate number of multipliers and ALUs such that the net performance degradation due to the low frequency operation of multipliers can be overcome by high frequency operation of ALUs then the reduction was significant. With such a scheduler incorporated into a low-power datapath synthesis tool will greatly benefit low power processor design especially for compute intensive applications.

CHAPTER 5

PEAK POWER AND AVERAGE POWER MINIMIZATION

The use of multiple supply voltages for energy and average power reduction is well researched and several works have appeared in the literature. However, in low power design for deep submicron and nanometer regimes, the peak power, peak power differential, average power and total energy are equally critical design constraints. In this work, we propose datapath scheduling algorithms for simultaneous minimization of peak and average power [46]. The minimization schemes based on integer linear programming (ILP) are developed for the design of datapaths that can function in three modes of operation: (1) single supply voltage and single frequency (SVSF), (2) multiple supply voltages and dynamic frequency clocking (MVDFC) and (3) multiple supply voltages and multicycling (MVMC). The use of dynamic frequency clocking is effective for power reduction in design of data intensive signal processing applications. The effectiveness of our proposed technique is measured by estimating the peak power consumption, the average power consumption and the power delay product of the datapath circuits. Various experiments are conducted on selected high-level synthesis benchmark circuits under different resource constraints.

This chapter is organized as follows. The ILP-formulations to minimize the peak and average power consumption are described first. The ILP-based scheduler is then introduced, followed by experimental results. We also investigated the scheduling schemes for only peak power minimzation without considering average power, which is represented in the last section.

5.1 Peak and Average Power Consumption of a Datapath Circuit

In this section, we first mention the different notations and terminology needed for a scheduling model. Let us assume that the datapath is represented in the form of a sequencing data flow graph. The datapath uses various resources or functional units operating at different supply voltages. The

level converters are considered as resource overheads often needed when the voltage level needs to be stepped up in any control step. The dynamic clocking unit (DCU) that generates the different frequency levels is also accounted as a resource that will operate during all the control steps. The notation and terminolgies are given in Table 5.1. It may be noted that for single frequency and single supply voltage mode of operation, $V_{i,c}$ and f_c are the same for any clock cycle (c) and resource (i). Similarly, for multicycling operation f_c is the same for any clock cycle (c).

Table 5.1. Notations used in Description

c	: any control step or clock cycle in DFG
N	: total number of control steps in the DFG
R_c	: number of resources active in step c
f_c	: cycle frequency for control step c
$\alpha_{i,c}$	: switching at resource i operating in step c
$C_{i,c}$	: load capacitance of resource i operating in control step c
$V_{i,c}$	: operating voltage of resource i operating in control step c
P_c	: power consumption for the DFG for any control step c
P_p	: maximum power consumption for the DFG
P_a	: average power consumption for the DFG
T	: critical path delay of the DFG
PDP	: power delay product of the DFG

The power consumption for any control step c is

$$P_c = \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c \tag{5.1}$$

The peak power consumption of the DFG is the maximum power consumption over all the control steps which is expressed as below.

$$P_p = Max\left(P_c\right)_{\forall c=1,2,\ldots,N} \tag{5.2}$$

We rewrite Eqn. 5.2 using Eqn. 5.1 as follows.

$$P_p = Max\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right)_{\forall c=1,2,\ldots.N} \tag{5.3}$$

The average power consumption of the DFG is characterised as the mean of the cycle powers (P_c) for all control steps.

$$P_a = \frac{1}{N}\sum_{i=1}^{N} P_c \tag{5.4}$$

Again using Eqn. 5.1, we rewrite Eqn. 5.4 as follows.

$$P_a = \frac{1}{N}\sum_{i=1}^{N}\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c \tag{5.5}$$

Since the simultaneous reduction of both peak and average power is aimed for, the objective function to be minimized by the scheduling algorithm is the sum of Eqn. 5.3 and 5.5.

The critical path delay of the DFG can be calculated as,

$$T = \sum_{i=1}^{N} \frac{1}{f_c} \tag{5.6}$$

It should be noted that the f_c is the same for single frequency and multicycling operations for all values of c and may be different for dynamic frequency clocking operations. The power delay product of the DFG is defined as the product of the average power consumption and critical path delay as shown below.

$$PDP = P_a * T \tag{5.7}$$

Using Eqns. 5.4 and 5.6, the following expression for the power delay product is obtained.

$$PDP = \frac{1}{N}\sum_{i=1}^{N} P_c * \sum_{i=1}^{N} \frac{1}{f_c} \tag{5.8}$$

Similarly, the following expression for the power delay product is arrived using Eqns. 5.5 and 5.6.

$$PDP = \frac{1}{N}\sum_{i=1}^{N}\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c * \sum_{i=1}^{N} \frac{1}{f_c} \tag{5.9}$$

To study the impact of the scheduling algorithms on the performance of the datapath the power delay product of the scheduled DFGs using the above expression will be estimated.

5.2 ILP Formulations

In this section, we discuss the ILP formulations to minimize the peak and average power consumption of a datapath circuit. We first discuss the formulations for multiple supply voltages and dynamic clocking based system followed by multiple supply voltages and multicycling based system.

5.2.1 ILP Formulations for DFC

In this section, the ILP formulation for simultaneous peak (Eqn. 5.3) and average power (Eqn. 5.5) minimization using multiple supply voltages and dynamic frequency clocking (DFC) is described. In dynamic frequency clocking [62, 63], the clock frequency is varied on-the-fly based on the functional units active in that cycle. In this clocking scheme, all the units are clocked by a single clock line which switches at run-time. The frequency reduction creates an opportunity to operate the different functional units at different voltages, which in turn, helps in further reduction of power. The notations used for ILP formulations are given in Table 5.2.

Table 5.2. Notations used in ILP Formulations

O	: total number of operations in the DFG excluding the source and sink nodes
o_i	: any operation i, $1 \leq i \leq O$
$F_{k,v}$	: functional unit of type k operating at voltage level v
$M_{k,v}$	: maximum number of functional units of type k operating at voltage level v
S_i	: as soon as possible (ASAP) time stamp for the operation o_i
E_i	: as late as possible (ALAP) time stamp for the operation o_i
$P(i, v, f)$	: power consumption of operation o_i at voltage level v and operating frequency f
$x_{i,c,v,f}$	: decision variable which takes the value of 1 if operation o_i is scheduled in control step c using the functional unit $F_{k,v}$ and c has frequency f_c
$y_{i,v,l,m}$	: decision variable which takes the value of 1 if o_i is using the functional unit $F_{k,v}$ and scheduled in control steps $l \rightarrow m$
$L_{i,v}$	: latency for operation o_i using resource operating at voltage v (in terms of number of clock cycles)

Objective Function : The objective is to minimize the peak power and the average power consumption of the whole DFG over all control steps simultaneously. These are already described above in

Eqn. 5.3 and 5.5.

$$Minimize \quad : \quad P_p + P_a \tag{5.10}$$

Using decision variables the objective function can be rewritten as follows :

$$Minimize \quad : \quad P_p + \frac{1}{N} \sum_c \sum_v \sum_{i \in F_{k,v}} \sum_f x_{i,c,v,f} * P(i,v,f) \tag{5.11}$$

It should be noted that the P_p is unknown and has to be minimized. It may be power consumption of any control step in the DFG depending on the scheduled operations and hence is later used as a constraint.

Uniqueness Constraints : These constraints ensure that each operation o_i is scheduled to one unique control step within the mobility range (S_i, E_i) with a particular supply voltage and operating frequency. They are represented as, $\forall i$, $1 \leq i \leq O$,

$$\sum_c \sum_v \sum_f x_{i,c,v,f} \quad = \quad 1 \tag{5.12}$$

Precedence Constraints : These constraints ascertain that for an operation o_i, all its predecessors are scheduled in an earlier control step and its successors are scheduled in an later control step. These are modelled as, $\forall i, j, o_i \in Pred_{o_j}$

$$\sum_v \sum_f \sum_{d=S_i}^{E_i} d * x_{i,d,v,f} - \sum_v \sum_f \sum_{e=S_j}^{E_j} e * x_{j,e,v,f} \quad \leq \quad -1 \tag{5.13}$$

Resource Constraints : These constraints establish that no control step contains more than $F_{k,v}$ operations of type k operating at voltage v. These can be enforced as, $\forall c$, $1 \leq c \leq N$ and $\forall v$,

$$\sum_{i \in F_{k,v}} \sum_f x_{i,c,v,f} \quad \leq \quad M_{k,v} \tag{5.14}$$

Frequency Constraints : This set ensures that if a functional unit is operating at higher voltage level then it can be scheduled in a lower frequency control step, whereas if a functional unit is operating

at lower voltage level then it can not be scheduled in a higher frequency control step. These constraints are written as, $\forall i, 1 \leq i \leq O, \forall c, 1 \leq c \leq N$, if $f < v$, then $x_{i,c,v,f} = 0$.

Peak Power Constraints : These constraints make certain that the maximum power consumption of the DFG does not exceed P_p for any control step. These constraints are applied as follows, $\forall c$, $1 \leq c \leq N$ and $\forall v$,

$$\sum_{i \in F_{k,v}} \sum_{f} x_{i,c,v,f} * P(i,v,f) \quad \leq \quad P_p \tag{5.15}$$

5.2.2 ILP Formulations for Multicycling

In this section, the ILP formulations for simultaneous minimization of both peak and average power consumption of the DFG using multiple supply voltages and multicycling will be discussed.

Objective Function : The objective is to minimize the peak and average power consumption of the whole DFG over all control steps. The expressions given in Eqn. 5.3 and Eqn. 5.5 are still valid here, with only difference being that f_c is the same for all control steps.

$$Minimize \quad : \quad P_p + P_a \tag{5.16}$$

In terms of decision variables, the above is written as :

$$Minimize \quad : \quad P_p + \frac{1}{N} \sum_{l} \sum_{i \in F_{k,v}} \sum_{v} y_{i,v,l,(l+L_{i,v}-1)} * P(i,v,f_{clk}) \tag{5.17}$$

The P_p is used as a constraint later.

Uniqueness Constraints : These constraints confirm that every operation o_i is scheduled in appropriate control steps within the mobility range (S_i, E_i) with a particular supply voltage. It may be operated at more than one clock cycle depending on the supply voltage. These constraints are

represented as, $\forall i$, $1 \leq i \leq O$,

$$\sum_v \sum_{l=S_i}^{S_i+E_i+1-L_{i,v}} y_{i,v,l,(l+L_{i,v}-1)} \quad = \quad 1 \tag{5.18}$$

When the operators are operating at highest voltage, they are scheduled in one unique control step, whereas, when they are to be operated at lower voltages they need more than one clock cycle for completion. Thus, for lower voltage the mobility is restricted.

Precedence Constraints : These constraints guarantee that for an operation o_i, all its predecessors are scheduled in an earlier control step and its successors are scheduled in an later control step. These constraints should also take care of the multicycling operations. These are modeled as, $\forall i, j, o_i \in Pred_{o_j}$,

$$\sum_v \sum_{l=S_i}^{E_i} (l + L_{i,v} - 1) * y_{i,v,l,(l+L_{i,v}-1)} - \sum_v \sum_{l=S_j}^{E_j} l * y_{j,v,l,(l+L_{j,v}-1)} \quad \leq \quad -1 \tag{5.19}$$

Resource Constraints : These constraints make sure that no control step contains more than $F_{k,v}$ operations of type k operating at voltage v. These can be enforced as, $\forall v$ and $\forall l$, $1 \leq l \leq N$,

$$\sum_{i \in F_{k,v}} \sum_l y_{i,v,l,(l+L_{i,v}-1)} \quad \leq \quad M_{k,v} \tag{5.20}$$

Peak Power Constraints : These constraints ensure that the maximum power consumption of the DFG does not exceed P_p for any control step. These constraints are enforced as follows, $\forall l$, $1 \leq l \leq N$,

$$\sum_{i \in F_{k,v}} \sum_v y_{i,v,l,(l+L_{i,v}-1)} * P(i, v, f_{clk}) \quad \leq \quad P_p \tag{5.21}$$

5.3 ILP-Based Scheduler

In this section, we discuss the solutions for the ILP formulations obtained in the previous section. We assume the same target architecture and the characterised datapath components as

used in [55]. In this architecture, level converters are used when a low-voltage functional unit drives a high-voltage functional unit [65]. Peak power consumption of the DFG is minimized by the ILP based scheduler outlined in Fig. 5.1. The first step is to determine the as soon as possible (ASAP) time stamp of each operation. The second step is the determination of the as late as possible (ALAP) time stamp of each vertex for the DFG. The ASAP time stamp is the start time and the ALAP time stamp is the finish time of each operation. These two times provide the mobility of an operation and the operation must be scheduled in this mobile range. This mobility graph needs to be modified for the multicycling scheme. The scheduler is based on the ILP formulations described in Section 5.2. At this point, the operating frequency of a functional unit is assumed as the inverse of its operational delay determined using the delay model given in [48]. The ILP formulations are solved to derive the scheduled DFG. The scheduler decides the cycle frequencies based on the formulas given in [48]. Finally, the power consumption of the scheduled DFG is estimated.

Step 1	: Find ASAP schedule of the UDFG.
Step 2	: Find ALAP schedule of the UDFG.
Step 3	: Determine the mobility graph of each node.
Step 4	: Modify the mobility graph for multicycling.
Step 5	: Construct the ILP formulations.
Step 6	: Solve the ILP formulations using LP-Solve.
Step 7	: Find the scheduled DFG.
Step 8	: Determine the cycle frequencies for DFC scheme.
Step 9	: Estimate the power consumptions of the DFG.

Figure 5.1. ILP-Based Scheduler

5.3.1 Scheduler using Multiple Voltages and Dynamic Frequency Clocking

The intermediate steps in the solution for the ILP formulations for the multiple supply voltages and dynamic frequency clocking is illustrated using the DFG shown in Fig. 5.2. The ASAP schedule is shown in Fig. 5.2(a) and the ALAP schedule is shown in Fig. 5.2(b). From the ASAP and ALAP schedules the mobility graph shown in Fig. 5.2(c) is determined. We have shown one such ILP formulations in Fig. 5.3 for the resource constraint (RC3), two multipliers at $2.4V$ and two ALU operating at $3.3V$ using switching activity of 0.5. In Fig. 5.3, we used the following

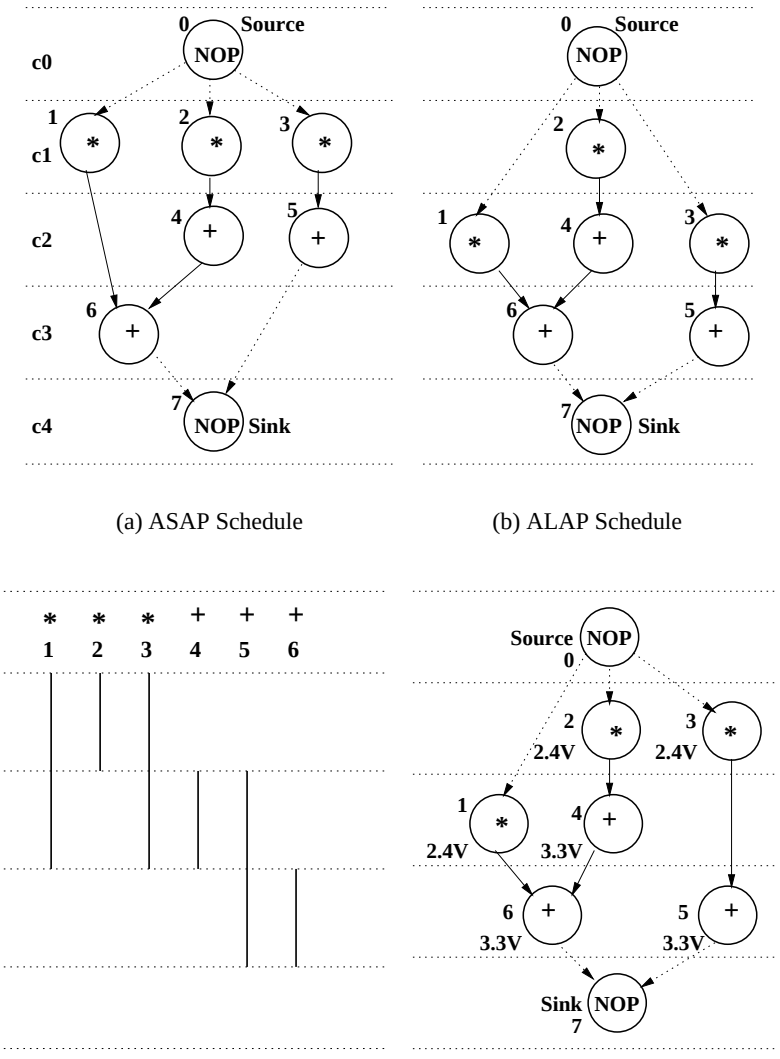
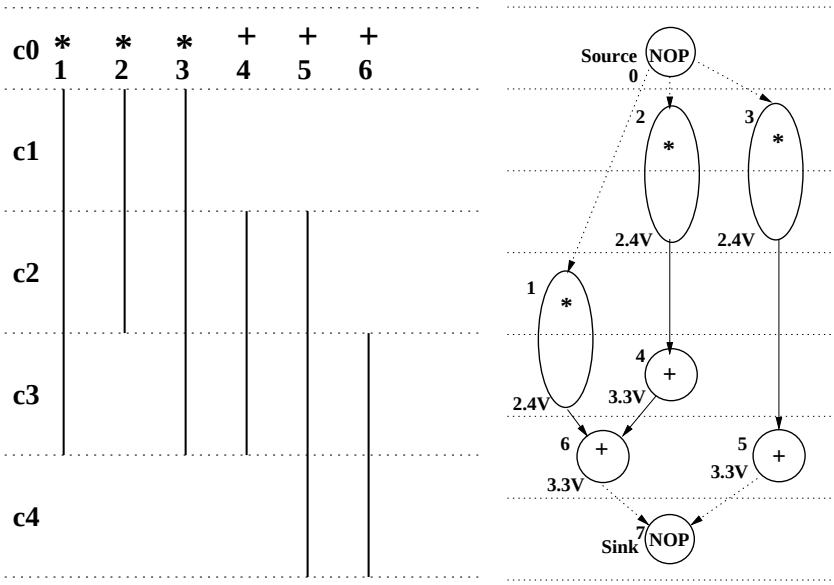


Figure 5.2. Example DFG for Resource Constraint RC3; using Multiple Supply Voltages and Dynamic Frequency Clocking

```
/* ILP Formulation for Simultaneous Peak and Average Power Minimization for MVDFC scheme */

/* Objective function */
min : 2.89 x1111 + 1.44 x1112 + 1.52 x1121 + 0.76 x1122 + 2.89 x2111 + 1.44 x2112
+ 1.52 x2121 + 0.76 x2122 + 2.89 x3111 + 1.44 x3112 + 1.52 x3121 + 0.76 x3122
+ 2.89 x1211 + 1.44 x1212 + 1.52 x1221 + 0.76 x1222 + 2.89 x3211 + 1.44 x3212
+ 1.52 x3221 + 0.76 x3222 + 0.08 x4211 + 0.04 x4212 + 0.04 x4221 + 0.02 x4222
+ 0.08 x5211 + 0.04 x5212 + 0.04 x5221 + 0.02 x5222 + 0.08 x5311 + 0.04 x5312
+ 0.04 x5321 + 0.02 x5322 + 0.08 x6311 + 0.04 x6312 + 0.04 x6321 + 0.02 x6322 + PP;
/* Uniqueness Constraints */
x1111 + x1112 + x1121 + x1122 + x1211 + x1212 + x1221 + x1222 = 1;
x2111 + x2112 + x2121 + x2122 = 1;
x3111 + x3112 + x3121 + x3122 + x3211 + x3212 + x3221 + x3222 = 1;
x4211 + x4212 + x4221 + x4222 = 1;
x5211 + x5212 + x5221 + x5222 + x5311 + x5312 + x5321 + x5322 = 1;
x6311 + x6312 + x6321 + x6322 = 1;
/* Precedence Constraints */
3 x6311 + 3 x6312 + 3 x6321 + 3 x6322 - 2 x1211 - 2 x1212 - 2 x1221 - 2 x1222
                - x1111 - x1112 - x1121 - x1122 ≥ 1;
2 x4211 + 2 x4212 + 2 x4221 + 2 x4222 - x2111 - x2112 - x2121 - x2122 ≥ 1;
3 x6311 + 3 x6312 + 3 x6321 + 3 x6322 - 2 x4211 - 2 x4212 - 2 x4221 - 2 x4222 ≥ 1;
3 x5311 + 3 x5312 + 3 x5321 + 3 x5322 + 2 x5211 + 2 x5212 + 2 x5221 + 2 x5222
                - 2 x3211 - 2 x3212 - 2 x3221 - 2 x3222 - x3111 - x3112 - x3121 - x3122 ≥ 1;
/* Resource Constraints */
x1111 + x2111 + x3111 + x1112 + x2112 + x3112 ≤ 0; /* Mmult1 */
x1121 + x2121 + x3121 + x1122 + x2122 + x3122 ≤ 2; /* Mmult2 */
x1211 + x3211 + x1212 + x3212 ≤ 0; /* Mmult1 */
x1221 + x3221 + x1222 + x3222 ≤ 2; /* Mmult2 */
x4211 + x5211 + x4212 + x5212 ≤ 2; /* Malu1 */
x4221 + x5221 + x4222 + x5222 ≤ 0; /* Malu2 */
x5311 + x6311 + x5312 + x6312 ≤ 2; /* Malu1 */
x5321 + x6321 + x5322 + x6322 ≤ 0; /* Malu2 */
/* Frequency Constraints */
x1121 = 0; x1221 = 0; x2121 = 0; x3121 = 0; x3221 = 0; x4221 = 0; x5221 = 0; x5321 = 0; x6321 = 0;
/* Peak Power Constraints */
8.64 x1111 + 4.32 x1112 + 4.56 x1121 + 2.28 x1122 + 8.64 x2111 + 4.32 x2112 + 4.56 x2121
                + 2.28 x2122 + 8.64 x3111 + 4.32 x3112 + 4.56 x3121 + 2.28 x3122 ≤ PP;
8.64 x1211 + 4.32 x1212 + 4.56 x1221 + 2.28 x1222 + 8.64 x3211 + 4.32 x3212 + 4.56 x3221
                + 2.28 x3222 + 0.23 x4211 + 0.11 x4212 + 0.12 x4221 + 0.06 x4222
                + 0.23 x5211 + 0.11 x5212 + 0.12 x5221 + 0.06 x5222 ≤ PP;
0.23 x5311 + 0.11 x5312 + 0.12 x5321 + 0.06 x5322 + 0.23 x6311 + 0.11 x6312 + 0.12 x6321
                + 0.06 x6322 ≤ PP;
/* Integer Constraints */
INT x1111, x1112, x1121, x1122, x1211, x1212, x1221, x1222, x2111, x2112, x2121, x2122, x3111,
x3112, x3121, x3122, x3211, x3212, x3221, x3222, x4211, x4212, x4221, x4222, x5211, x5212,
x5221, x5222, x5311, x5312, x5321, x5322, x6311, x6312, x6321, x6322;
```

Figure 5.3. ILP Formulation for Example DFG using DFC, for RC3 and Switching Activity = 0.5



(a) Mobility Graph (b) Final Schedule

Figure 5.4. Example DFG for Resource Constraint RC3; using Multiple Supply Voltages and Multicycling

additional notations : (i) PP : peak power, (ii) $Mmult1$: number of multipliers at voltage level 1, (iii) $Mmult2$: number of multipliers at voltage level 2, (iv) $Malu1$: number of ALUs at voltage level 1, and (v) $Malu2$: number of ALUs at voltage level 2. The ILP formulations are solved using LP-solve and the scheduled DFG is shown in Fig. 5.2(d).

5.3.2 Scheduler using Multiple Supply Voltages and Multicycling

The solution for the ILP formulation for multiple supply voltages and multicycling is illustrated using the DFG shown in Fig. 5.4. The ASAP schedule is shown in Fig. 5.2 and the ALAP schedule is shown in Fig. 5.2(a). From the ASAP and ALAP schedules the mobility graph shown in Fig. 5.4(a) is obtained. This mobility graph is different from that shown in Fig. 5.2(c); The mobility graph in Fig. 5.4(a) considers the multicycle operations. Two operating voltage levels are assumed in Fig. 5.4(a). The multipliers take two clock cycles when operated at low voltage level. For the characterised cells used in our experiment [55], the operating clock frequency, f_{clk} is $9MHz$. The ILP formulations are obtained using this mobility graph. We have shown one such ILP formulation

```
/* ILP Formulation for Simultaneous Peak and Average Power Minimization for MVMC scheme */

/* Objective function */
min: 1.7 x1111 + 0.9 x1212 + 1.7 x2111 + 0.9 x2212 + 1.7 x3111 + 0.9 x3212 + 1.7 x1122 + 0.9 x1212
+ 0.9 x1223 + 1.7 x2122 + 0.9 x2212 + 0.9 x2223 + 1.7 x3122 + 0.9 x3212 + 0.9 x3223 + 0.05 x4122
+ 0.02 x4222 + 0.05 x5122 + 0.02 x5222 + 1.7 x1133 + 0.9 x1223 + 0.9 x1234 + 1.7 x2133 + 0.9 x2223
+ 1.7 x3133 + 0.9 x3223 + 0.9 x3234 + 0.05 x4133 + 0.02 x4233 + 0.05 x5133 + 0.02 x5233
+ 0.05 x6133 + 0.02 x6233 + 1.7 x1144 + 0.9 x1234 + 1.7 x3144 + 0.9 x3234 + 0.05 x4144
+ 0.02 x4244 + 0.05 x5144 + 0.02 x5244 + 0.05 x6144 + 0.02 x6244 + 0.05 x5155 + 0.02 x5255
+ 0.05 x6155 + 0.02 x6255 + PP;
/* Uniqueness Constraints */
x1111 + x1122 + x1133 + x1144 + x1212 + x1223 + x1234 = 1;
x2111 + x2122 + x2133 + x2212 + x2223 = 1;
x3111 + x3122 + x3133 + x3144 + x3212 + x3223 + x3234 = 1;
x4122 + x4133 + x4144 + x4222 + x4233 + x4244 = 1;
x5122 + x5133 + x5144 + x5155 + x5222 + x5233 + x5244 + x5255 = 1;
x6133 + x6144 + x6155 + x6233 + x6244 + x6255 = 1;
/* Peak Power Constraints */
8.6 x1111 + 4.6 x1212 + 8.6 x2111 + 4.6 x2212 + 8.6 x3111 + 4.6 x3212 ≤ PP;
8.6 x1122 + 4.6 x1212 + 4.6 x1223 + 8.6 x2122 + 4.6 x2212 + 4.6 x2223 + 8.6 x3122
                + 4.6 x3212 + 4.6 x3223 + 0.2 x4122 + 0.1 x4222 + 0.2 x5122 + 0.1 x5222 ≤ PP;
8.6 x1133 + 4.6 x1223 + 4.6 x1234 + 8.6 x2133 + 4.6 x2223 + 8.6 x3133 + 4.6 x3223 + 4.6 x3234
                + 0.2 x4133 + 0.1 x4233 + 0.2 x5133 + 0.1 x5233 + 0.2 x6133 + 0.1 x6233 ≤ PP;
8.6 x1144 + 4.6 x1234 + 8.6 x3144 + 4.6 x3234 + 0.2 x4144 + 0.1 x4244 + 0.2 x5144 + 0.1 x5244
                + 0.2 x6144 + 0.1 x6244 ≤ PP;
0.2 x5155 + 0.1 x5255 + 0.2 x6155 + 0.1 x6255 ≤ PP;
/* Resource Constraints */
x1111 + x2111 + x3111 ≤ 0; /* Mmult1 */      x1212 + x2212 + x3212 ≤ 2; /* Mmult2 */
x1122 + x2122 + x3122 ≤ 0; /* Mmult1 */
x1212 + x1223 + x2212 + x2223 + x3212 + x3223 ≤ 2; /* Mmult2 */
x1133 + x2133 + x3133 ≤ 0; /* Mmult1 */   x1223 + x1234 + x2223 + x3223 + x3234 ≤ 2; /* Mmult2 */
x1144 + x3144 ≤ 0; /* Mmult1 */      x1234 + x3234 ≤ 2; /* Mmult2 */
x4122 + x5122 ≤ 2; /* Malu1 */      x4222 + x5222 ≤ 0; /* Malu2 */
x4133 + x5133 + x6133 ≤ 2; /* Malu1 */      x4233 + x5233 + x6233 ≤ 0; /* Malu2 */
x4144 + x5144 + x6144 ≤ 2; /* Malu1 */      x4244 + x5244 + x6244 ≤ 0; /* Malu2 */
x5155 + x6155 ≤ 2; /* Malu1 */      x5255 + x6255 ≤ 0; /* Malu2 */
/* Precedence Constraints */
5 x6155 + 5 x6255 + 4 x6144 + 4 x6244 + 3 x6133 + 3 x6233 - 4 x1144 - 4 x1234 - 3 x1133
                - 3 x1223 - 2 x1122 - 2 x1212 - x1111 ≥ 1;
5 x6155 + 5 x6255 + 4 x6144 + 4 x6244 + 3 x6133 + 3 x6233 - 4 x4144 - 4 x4244 - 3 x4133
                - 3 x4233 - 2 x4122 - 2 x4222 ≥ 1;
4 x4144 + 4 x4244 + 3 x4133 + 3 x4233 + 2 x4122 + 2 x4222 - 3 x2133 - 3 x2223 - 2 x2122
                - 2 x2212 - x2111 ≥ 1;
5 x5155 + 5 x5255 + 4 x5144 + 4 x5244 + 3 x5133 + 3 x5233 + 2 x5122 + 2 x5222 - 4 x3144
                - 4 x3234 - 3 x3133 - 3 x3223 - 2 x3122 - 2 x3212 - x3111 ≥ 1;
/* Integer Constraints */
INT x1111, x1122, x1133, x1144, x1212, x1223, x1234, x2111, x2122, x2133, x2212, x2223, x3111,
x3122, x3133, x3144, x3212, x3223, x3234, x4122, x4133, x4144, x4222, x4233, x4244, x5122, x5133,
x5144, x5155, x5222, x5233, x5244, x5255, x6133, x6144, x6155, x6233, x6244, x6255;
```

Figure 5.5. ILP Formulation for Example DFG using Multicycling, for RC3 and Switching Activity = 0.5

in Fig. 5.5 for the resource constraint (RC3), two multipliers at $2.4V$ two ALUs at $3.3V$, and switching activity $= 0.5$. In Fig. 5.5, the notations, such as, PP, $Mmult1$, $Mmult2$, $Malu1$ and $Malu2$ have same meaning as that of the DFC case shown in Fig. 5.3. The ILP formulations are solved using LP-solve and the scheduled DFG is shown in Fig. 5.4(b).

5.4 Experimental Results

The ILP-based schedulers for both multiple supply voltages and dynamic clocking frequency, and multiply supply voltages and multicycling schemes were tested with five high-level synthesis benchmark circuits : (1) Example circuit (EXP), (2) FIR filter, (3) IIR filter, (4) HAL differential equation solver and (5) Auto-Regressive filter (ARF). The notations used to express the various results are given in Table 5.3.

The schedulers were tested using different sets of resource constraints (RC1,RC2,RC3,RC4,RC5) shown in Table 5.4 for each benchmark circuit. The experimental results for various benchmark circuits are reported in Table 5.5 for both dynamic frequency clocking and multicycling schemes. The power is estimated including the overheads, such as level converters (used in both the schemes) and dynamic clocking units (needed for dynamic frequency clocking case). It is assumed that each resource has equal switching activity ($\alpha_{i,c}$). The results are reported for two supply voltages and for switching $= 0.5$.

To get a visual picture of the experimental results, we plotted the peak power reductions, average power reduction and the PDP reductions averaged over the different sets of resource constraints. Fig. 5.6 shows the average reductions for different benchmarks averaged over all resource constraints. It is obvious from the figure that the reductions using combined multiple supply voltages and dynamic frequency clocking are appreciable. It is observed that the power consumption increases for higher switching and decreases for lower switching activity. The power reductions for the proposed scheduling scheme are listed alongwith other scheduling algorithms dealing with peak power reduction in Table 5.6. The table is not to provide an exact comparison, but to provide a general idea of relative performance.

Table 5.3. Notations used in Expressing Results

P_{pS}	: the peak power consumption (in mW) for single supply voltage and single frequency operation
P_{pD}	: the peak power consumption (in mW) for multiple supply voltages and dynamic frequency operation
P_{pM}	: the peak power consumption (in mW) for multiple supply voltages and multicycle operation
P_{aS}	: the average power consumption (in mW) for single supply voltage and single frequency operation
P_{aD}	: the average power consumption (in mW) for multiple supply voltages and dynamic frequency operation
P_{aM}	: the average power consumption (in mW) for multiple supply voltages and multicycle operation
T_S	: the critical path delay for single supply voltage and single frequency operation
T_D	: the critical path delay for multiple supply voltages and dynamic frequency operation
T_M	: the critical path delay for multiple supply voltages and multicycle operation
PDP_S	: the power delay product (in nJ) for single supply voltage and single frequency operation $(= P_{aS} * T_S)$
PDP_D	: the power delay product (in nJ) for multiple supply voltage and dynamic frequency clocking operation $(= P_{aD} * T_D)$
PDP_M	: the power delay product (in nJ) for multiple supply voltage and multicycle operation $(= P_{aM} * T_M)$
ΔP_{pD}	: the percentage peak power reduction using the multiple supply voltages and dynamic frequency scheme $\left(= \frac{(P_{pS} - P_{pD})}{P_{pS}} * 100\right)$
ΔP_{pM}	: the percentage peak power reduction using the multiple supply voltages and multicycle scheme $\left(= \frac{(P_{pS} - P_{pM})}{P_{pS}} * 100\right)$
ΔPDP_D	: the percentage PDP reduction using the multiple supply voltages and dynamic frequency scheme $\left(= \frac{(PDP_S - PDP_D)}{PDP_S} * 100\right)$
ΔPDP_M	: the percentage PDP reduction using the multiple supply voltages and multicycle scheme $\left(= \frac{(PDP_S - PDP_M)}{PDP_S} * 100\right)$

Table 5.4. Resource Constraints used for our Experiement

Resource Constraints				Resource Constraint Labels
Multipliers		ALUs		
2.4 V	3.3 V	2.4 V	3.3 V	
2	1	1	1	RC1
3	0	1	1	RC2
2	0	0	2	RC3
1	1	0	1	RC4
2	0	0	1	RC5

5.5 Peak Power Minimization

In the previous few sections we have presented the formulations for simultaneous minimization of peak and average power of a datapath circuit. In this section we discuss the ILP-based scheduling scheme that minimizes peak power only without explicitly considering the average power [45, 165]. The peak power consumption presented in Eqn. 5.2 serves as the objective function. The peak power consumption Eqn. has been reproduced here for quick reference, where the notations are the same meaning as used before.

$$\begin{aligned} P_p &= Max\left(P_c\right)_{\forall c=1,2,....N} \\ &= Max\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right)_{\forall c=1 \rightarrow N} \end{aligned} \tag{5.22}$$

The above equation can be rewritten as follows for multiple supply voltages and multicycling operation scenario; clock frequency is the same for all control steps and denoted as f_{clk}.

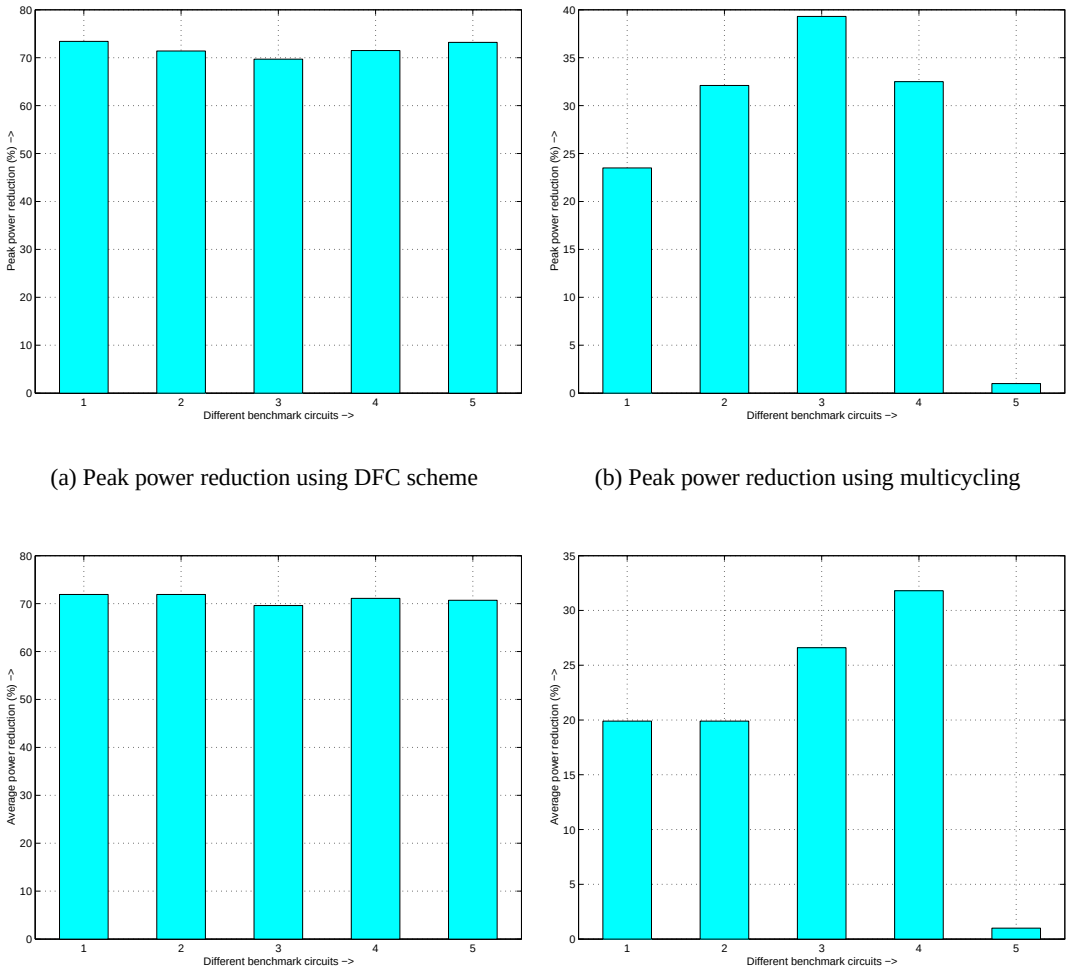
$$\begin{aligned} P_p &= Max\left(P_c\right)_{\forall c=1,2,....N} \\ &= Max\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_{clk}\right)_{\forall c=1 \rightarrow N} \end{aligned} \tag{5.23}$$

5.5.1 ILP Formulations

In this section, we formulate the ILP models for peak power minimization for both MVDFC and MVMC scenario. The ILP models ensure that the dependency constraints and resource constraints are satisfied. The level converters are considered as resources operating in the control step in which

Table 5.5. Peak Power, Average Power and PDP Estimates for Benchmarks using Scheduling Schemes

	R		Peak Power (mW)				Average Power (mW)					PDP Estimates (nJ)				
	C	P_{pS}	P_{pD}	ΔP_{pD}	P_{pM}	ΔP_{pM}	P_{aS}	P_{aD}	ΔP_{aD}	P_{aM}	ΔP_{aM}	PDP_S	PDP_D	ΔPDP_D	PDP_M	ΔPDP_M
(1)	1	17.28	4.56	73.6	8.76	49.3	8.86	2.41	72.8	6.57	25.8	2.95	1.33	54.9	2.92	0
e	2	17.28	4.56	73.6	13.68	20.8	8.86	2.41	72.8	6.98	21.2	2.95	1.33	54.9	3.1	0
x	3	17.28	4.56	73.6	9.12	47.2	8.86	2.61	70.5	5.58	37.0	2.95	1.30	55.9	3.1	0
p	4	8.86	2.39	73.0	8.86	0	6.65	1.88	71.7	6.65	0	2.96	1.36	54.1	2.95	0
	Average values			73.5		29.3			72.0		21.0			55.0		0
(2)	1	17.28	4.56	73.6	8.76	49.3	8.82	2.34	73.5	7.28	17.5	4.9	2.34	52.5	4.85	0
f	2	17.28	4.56	73.6	13.68	20.8	8.82	2.35	73.4	7.68	12.9	4.9	2.35	52.0	5.12	0
i	3	17.28	4.56	73.6	13.68	20.8	8.82	2.44	72.3	6.64	24.7	4.9	2.30	53.0	5.12	0
r	4	17.28	6.60	61.8	8.86	48.7	8.82	2.84	67.8	7.35	16.7	4.9	2.68	45.3	4.9	0
	Average values			70.7		34.9			71.8		18.0			50.7		0
(3)	1	25.92	8.88	65.7	17.76	31.5	11.03	3.49	68.4	8.95	18.9	4.9	2.32	52.7	4.97	0
i	2	25.92	6.84	73.6	13.68	47.2	11.03	2.98	73.0	7.68	30.4	4.9	1.98	59.6	5.12	0
i	3	17.28	4.56	73.6	9.12	47.2	8.82	2.45	72.2	5.24	40.6	4.9	2.0	59.2	4.66	4.9
r	4	17.28	6.60	61.8	13.20	23.6	8.82	3.31	62.5	8.05	8.7	4.9	2.57	47.6	5.37	0
	Average values			68.7		37.4			69.0		24.7			54.8		1.0
(4)	1	17.51	4.62	74.7	13.32	23.9	13.25	3.55	73.2	8.82	33.4	5.89	2.76	53.1	5.88	0.2
h	2	17.51	4.62	74.7	13.68	21.9	13.25	3.55	73.2	9.23	30.3	5.89	2.76	53.1	6.15	0
a	3	17.51	4.67	73.3	9.34	46.7	13.25	3.73	71.8	7.98	39.8	5.89	2.69	54.3	6.20	0
l	4	17.51	6.71	61.7	13.42	23.4	10.59	3.73	64.8	8.90	16.0	5.88	3.52	40.1	5.93	0
	Average values			71.1		29.0			70.8		29.9			50.2		0.7
(5)	1	8.86	2.34	73.6	8.64	2.5	4.50	1.20	73.3	3.40	24.4	5.00	2.00	60.0	4.85	3.0
a	2	8.86	2.34	73.6	8.64	2.5	4.50	1.20	73.3	3.58	24.4	5.00	2.00	60.0	4.85	3.0
r	3	8.86	2.39	73.0	8.76	1.1	4.50	1.40	68.9	3.65	18.9	5.00	1.90	62.0	5.0	0
f	4	8.86	2.39	73.0	8.76	1.1	4.50	1.40	68.9	3.46	23.1	5.00	1.90	62.0	5.0	0
	Average values			73.3		1.8			71.1		22.7			61.0		1.1
Average over all benchmarks				71.5		26.5			71.0		23.3			54.3		0.5



(a) Peak power reduction using DFC scheme

(b) Peak power reduction using multicycling

(c) Average power reduction using DFC scheme

(d) Average power reduction using multicycling

Figure 5.6. Average Reduction for Different Bechmarks

it needs to step up signal. The dynamic clocking unit (DCU) that generates dynamic frequency is considered as a resource operating in all the control steps. The power dissipation of the level converters and DCU are included. In order to formulate an ILP based model for Eqn. 5.22 and hence a scheduling scheme for the DFG, we use the same notations given in Table 5.2.

5.5.1.1 Multiple Supply Voltages and Dynamic Frequency Clocking (MVDFC)

In this subsection, we describe the ILP formulation for peak power minimization using multiple supply voltages and dynamic frequency clocking. In dynamic frequency clocking, the clock

Table 5.6. Peak and Average Power Reduction for Various Scheduling Schemes

Bench-mark Circuits	Percentage average data for various schemes									
	DFC based		Shiue [119]		Martin [44]		Raghunathan [47]		Mohanty [48]	
	ΔP_p	ΔP_a	ΔP_p	ΔP_a	ΔP_p	ΔP_a	ΔP_p	ΔP_a	ΔP_p	ΔP_a
EXP(1)	73	72	-	-	-	-	-	-	-	-
FIR(2)	71	72	63	NA	40	NO	23	38	71	53
IIR(3)	69	69	-	-	-	-	-	-	-	-
HAL(4)	71	71	28	NA	-	-	-	-	73	70
ARF(5)	73	71	50	NA	-	-	-	-	68	67

frequency is varied on-the-fly based on the functional units active in that cycle. In this clocking scheme, all the units are clocked by a single clock line which switches at run-time. The frequency reduction creates an opportunity to operate the different functional units at different voltages, which in turn, helps in further reduction of power.

In this case the objective is to minimize the peak power consumption of the whole DFG over all control steps described in Eqn. 5.22 without explicitly considering the average power minimzation. Thus the objective function changes into the equation given below.

$$Minimize \quad : \quad P_p \quad \left(= Max \left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c \right)_{\forall c=1 \rightarrow N} \right) \tag{5.24}$$

It should be noted that the P_{peak} is an unknown which has to be minimized. It may be power consumption of any control step in the DFG depending on the scheduled operations and hence is later used as a constraint. The constraints of the formulation, such as uniqueness constraints, precedence constraints, resource constraints, frequency constraints, and peak power constraints remains the same as before.

5.5.1.2 Multiple Supply Voltages and Multicycling (MVMC)

In this subsection, we describe the ILP formulation for peak power minimization using multiple supply voltages and multicycling. In this scheme, the functional units are operated at multiple supply voltages and the lower operating voltage functional units are scheduled in consecutive control steps. In this case the objective is to minimize the peak power consumption of the whole

DFG over all control steps described in Eqn. 5.23 without explicitly considering the average power minimization. Thus the ILP formulation becomes as the one given below.

$$Minimize \quad : \quad P_p \quad \left(= Max\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_{clk}\right)_{\forall c=1 \rightarrow N}\right) \tag{5.25}$$

It should be noted that the P_{peak} is an unknown which has to be minimized. It may be power consumption of any control step in the DFG depending on the scheduled operations and hence is later used as a constraint. The constraints of the formulation, such as uniqueness constraints, precedence constraints, resource constraints, and peak power constraints remains the same as before.

5.5.2 ILP-Based Scheduler

In this section, we will discuss the solutions for the ILP formulations obtained in the previous section. The target architecture and characterised datapath components are from [55]. The ILP based scheduler which minimizes peak power consumption of the DFG has basically the same steps as the one presented for simultaneous peak and average presented in Fig. 5.1. The first step is to determine the as soon as possible (ASAP) time stamp of each operation. The second step is the determination of the as late as possible (ALAP) time stamp of each vertex for the DFG. The ASAP time stamp is the start time and the ALAP time stamp is the finish time of each operation. These two times provide the mobility of an operation and the operation must be scheduled in this mobile range. This mobility graph needs to be modified for the MVMC scheme. Then the scheduler determines the ILP formulations based on the models described in Section 5.5.1. After the ILP formulation is solved (using LP-Solve) the scheduled DFG is obtained. The scheduler determines the cycle frequencies for the scheduled DFG for the MVDFC scheme.

5.5.2.1 Scheduling for MVDFC

We illustrate the solution for the ILP formulation in the MVDFC case, with the help of the DFG shown in Fig. 5.7. The ASAP schedule is shown in Fig. 5.7(a) and the ALAP schedule is shown in Fig. 5.7(b). From the ASAP and ALAP schedules we obtain the mobility graph as in

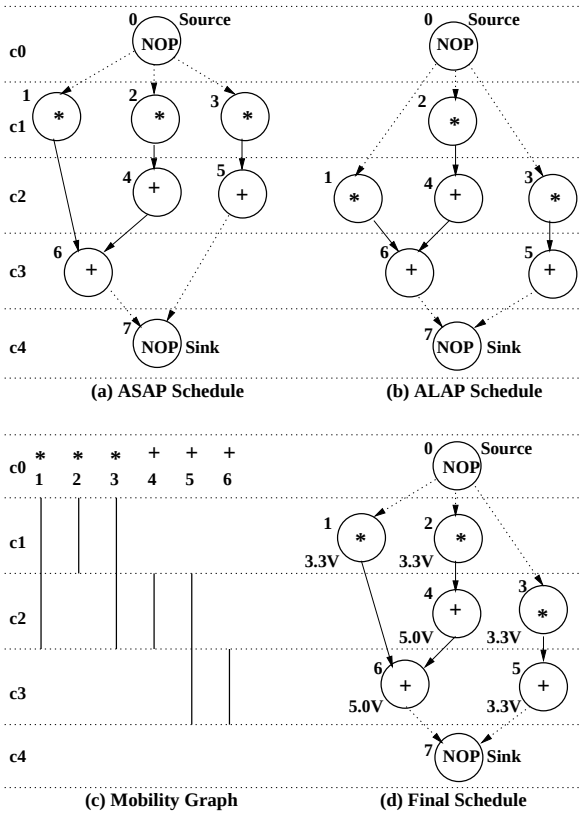


Figure 5.7. Example DFG (for RC1) (MVDFC)

Fig. 5.7(c). Using this mobility graph, we have the ILP formulations shown in Fig. 5.8 for the resource constraint (RC1) : two multipliers at $3.3V$, one multiplier at $5.0V$, one ALU at $3.3V$ and one ALU operating at $5.0V$. We solved the formulation using LP-solve and based on the results, we obtained the scheduled DFG shown is Fig. 5.7(d). In Fig. 5.8, we used the following additional notations, PP : peak power, $Mmult1$: number of multipliers at voltage level 1, $Mmult2$: number of multipliers at voltage level 2, $Malu1$: number of ALUs at voltage level 1, and $Malu2$: number of ALUs at voltage level 2. The corresponding formulation expressed in AMPL [166] is given in Fig. 5.9.

5.5.2.2 Scheduling for MVMC

We illustrate solution for the ILP formulation of the MVMC case, with the help of the DFG shown in Fig. 5.10. The ASAP schedule is shown in Fig. 5.10(a) and the ALAP schedule is

```
/* ILP Formulation for Peak Power Minimization for MVDFC scheme */

/* Objective Function */

min: PP;

/* Uniqueness Constraints */
x1111 + x1112 + x1121 + x1122 + x1211 + x1212 + x1221 + x1222 = 1;
x2111 + x2112 + x2121 + x2122 = 1;
x3111 + x3112 + x3121 + x3122 + x3211 + x3212 + x3221 + x3222 = 1;
x4211 + x4212 + x4221 + x4222 = 1;
x5211 + x5212 + x5221 + x5222 + x5311 + x5312 + x5321 + x5322 = 1;
x6311 + x6312 + x6321 + x6322 = 1;

/* Precedence Constraints */
3x6311 + 3 x6312 + 3 x6321 + 3 x6322 - 2 x1211 - 2 x1212 - 2 x1221 - 2 x1222
        - x1111 - x1112 - x1121 - x1122 ≥ 1;
2 x4211 + 2 x4212 + 2 x4221 + 2 x4222 - x2111 - x2112 - x2121 - x2122 ≥ 1;
3 x6311 + 3 x6312 + 3 x6321 + 3 x6322 - 2 x4211 - 2 x4212 - 2 x4221 - 2 x4222 ≥ 1;
3 x5311 + 3 x5312 + 3 x5321 + 3 x5322 + 2 x5211 + 2 x5212 + 2 x5221 + 2 x5222 - 2 x3211
        - 2 x3212 - 2 x3221 - 2 x3222 - x3111 - x3112 - x3121 - x3122 ≥ 1;

/* Resource Constraints */
x1111 + x2111 + x3111 + x1112 + x2112 + x3112 ≤ 1; /* Mmult1 */
x1121 + x2121 + x3121 + x1122 + x2122 + x3122 ≤ 2; /* Mmult2 */
x1211 + x3211 + x1212 + x3212 ≤ 1; /* Mmult1 */
x1221 + x3221 + x1222 + x3222 ≤ 2; /* Mmult2 */
x4211 + x5211 + x4212 + x5212 ≤ 1; /* Malu1 */
x4221 + x5221 + x4222 + x5222 ≤ 1; /* Malu2 */
x5311 + x6311 + x5312 + x6312 ≤ 1; /* Malu1 */
x5321 + x6321 + x5322 + x6322 ≤ 1; /* Malu2 */

/* Frequency Constraints */
x1121 = 0; x1221 = 0; x2121 = 0; x3121 = 0; x3221 = 0; x4221 = 0; x5221 = 0; x5321 = 0; x6321 = 0;

/* Peak Power Constraints */
39.6 x1111 + 19.8 x1112 + 17.3 x1121 + 8.6 x1122 + 39.6 x2111 + 19.8 x2112 + 17.3 x2121
        + 8.6 x2122 + 39.6 x3111 + 19.8 x3112 + 17.3 x3121 + 8.6 x3122 ≤ PP;
39.6 x1211 + 19.8 x1212 + 17.3 x1221 + 8.6 x1222 + 39.6 x3211 + 19.8 x3212
        + 17.3 x3221 + 8.6 x3222 + 1.0 x4211 + 0.5 x4212 + 0.5 x4221 + 0.2 x4222
        + 1.0 x5211 + 0.5 x5212 + 0.5 x5221 + 0.2 x5222 ≤ PP;
1.0 x5311 + 0.5 x5312 + 0.5 x5321 + 0.2 x5322 + 1.0 x6311 + 0.5 x6312
        + 0.5 x6321 + 0.2 x6322 ≤ PP;

/* Integer Constraints */
INT x1111, x1112, x1121, x1122, x1211, x1212, x1221, x1222, x2111, x2112, x2121, x2122, x3111,
x3112, x3121, x3122, x3211, x3212, x3221, x3222, x4211, x4212, x4221, x4222, x5211, x5212,
x5221, x5222, x5311, x5312, x5321, x5322, x6311, x6312, x6321, x6322;
```

Figure 5.8. ILP Formulation for Example DFG (MVDFC)

```
    /* ILP Formulation for Peak Power Minimization for MVDFC scheme */

  param TASK;                                  # number of Tasks
  param LEVEL;                                 # number of levels in DFG
  param VOLT;                                  # number of voltage levels
  param FREQ;                                  # number of frequency levels
  param ASAP {1..TASK} > 0, ≤ LEVEL;           #ASAP Schedule for each Task
  param ALAP {1..TASK} > 0, ≤ LEVEL;           #ALAP Schedule for each Task
  param OP {1..TASK};                          #Type of Functional Unit
  param POWER {1..2, 1..VOLT, 1..FREQ};        #Power Consumption of each Functional Unit
  param M {1..2, 1..VOLT};                     #Resource Constraints

var PP;
var X {i in 1..TASK, j in ASAP[i]..ALAP[i], v in 1..VOLT, f in 1..FREQ} binary;

#Objective Function
minimize peak_power : PP;

# Uniqueness Constraints
subject to uniq_cons {i in 1..TASK}:
      sum {j in ASAP[i]..ALAP[i], v in 1..VOLT, f in 1..FREQ} X[i, j, v, f] = 1;

# Precedence Constraints
subject to pred_cons1:
      sum {j in ASAP[6]..ALAP[6], v in 1..VOLT, f in 1..FREQ} j * X[6, j, v, f]
            - sum {j in ASAP[1]..ALAP[1], v in 1..VOLT, f in 1..FREQ} j * X[1, j, v, f] ≥ 1;
subject to pred_cons2:
      sum {j in ASAP[4]..ALAP[4], v in 1..VOLT, f in 1..FREQ} j * X[4, j, v, f]
            - sum {j in ASAP[2]..ALAP[2], v in 1..VOLT, f in 1..FREQ} j * X[2, j, v, f] ≥ 1;
subject to pred_cons3:
      sum {j in ASAP[6]..ALAP[6], v in 1..VOLT, f in 1..FREQ} j * X[6, j, v, f]
            - sum {j in ASAP[4]..ALAP[4], v in 1..VOLT, f in 1..FREQ} j * X[4, j, v, f] ≥ 1;
subject to pred_cons4:
      sum {j in ASAP[5]..ALAP[5], v in 1..VOLT, f in 1..FREQ} j * X[5, j, v, f]
            - sum {j in ASAP[3]..ALAP[3], v in 1..VOLT, f in 1..FREQ} j * X[3, j, v, f] ≥ 1;

# Resource Constraints
subject to res_cons_mult {j in 1..LEVEL, v in 1..VOLT}:
      sum {f in 1..FREQ, i in 1..TASK: ASAP[i] ≤ j ≤ ALAP[i] && OP[i] = 2} X[i, j, v, f] ≤ M[2, v];
subject to res_cons_alu j in 1..LEVEL, v in 1..VOLT:
      sum {f in 1..FREQ, i in 1..TASK: ASAP[i] ≤ j ≤ ALAP[i] && OP[i] = 1} X[i, j, v, f] ≤ M[1, v];

# Peak Power Constraints
subject to pp_cons {j in 1..LEVEL}:
      sum {v in 1..VOLT, f in 1..FREQ, i in 1..TASK: ASAP[i] ≤ j ≤ ALAP[i]} POWER[OP[i], v, f]
      * X[i, j, v, f] ≤ PP;

#Frequency Constraints
subject to freq_cons {i in 1..TASK, j in ASAP[i]..ALAP[i]}:    X[i, j, 2, 1] = 0;
```

Figure 5.9. ILP Formulation for Example DFG (MVDFC) in AMPL

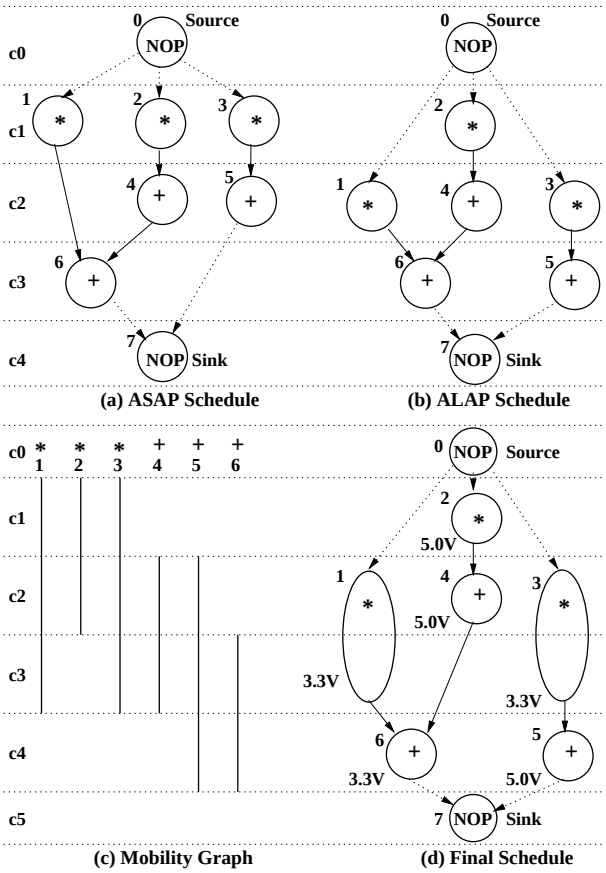


Figure 5.10. Example DFG (for RC1) (MVMC)

shown in Fig. 5.10(b). From the ASAP and ALAP schedules we obtain the mobility graph which is Fig.5.10(c). This mobility graph is different from that shown in Fig. 5.10(c). In the MVMC case, the mobility graph considers the multicycle operations. We assume two operating voltage levels, and when the multipliers are operated at lower voltage, they take two clock cycles. For the characterised cells used in our experiment [55], the operating clock frequency, f_{clk} is $18MHz$. Using this mobility graph, we have the ILP formulations shown in Fig. 5.11 for the resource constraint (RC1), two multipliers at $3.3V$, one multipliers at $5.0V$, one ALU at $3.3V$ and one ALUs operating at $5.0V$. The corresponding formulation expressed in AMPL [166] is given in Fig. 5.12. We solved the formulation using LP-solve and based on the results we obtained the scheduled DFG shown is Fig. 5.10(d). In Fig. 5.11, the notations, such as, PP, $Mmult1$, $Mmult2$, $Malu1$ and $Malu2$ have same meaning as that of the MVDFC case shown in Fig. 5.8.

```
/* ILP Formulation for Peak Power Minimization for MVMC scheme */

/* Objective Function */
min: PP;

/* Uniqueness Constraints */
x1212 + x1223 + x1111 + x1122 + x1133 = 1;
x2212 + x2111 + x2122 = 1;
x3111 + x3122 + x3133 + x3212 + x3223 = 1;
x4122 + x4133 + x4222 + x4233 = 1;
x5122 + x5133 + x5144 + x5222 + x5233 + x5244 = 1;
x6133 + x6144 + x6233 + x6244 = 1;

/* Peak Power Constraints */
39.6 x1111 + 8.6 x1212 + 39.6 x2111 + 8.6 x2212 + 39.6 x3111 + 8.6 x3212 ≤ PP;
39.6 x1122 + 8.6 x1212 + 8.6 x1223 + 39.6 x2122 + 8.6 x2212 + 39.6 x3122 + 8.6 x3212
        + 8.6 x3223 + 1.0 x4122 + 0.5 x4222 + 1.0 x5122 + 0.5 x5222 ≤ PP;
39.6 x1133 + 8.6 x1223 + 39.6 x3133 + 8.6 x3223 + 1.0 x4133 + 0.5 x4233 + 1.0 x5133
        + 0.5 x5233 + 1.0 x6133 + 0.5 x6233 ≤ PP;
1.0 x5144 + 0.5 x5244 + 1.0 x6144 + 0.5 x6244 ≤ PP;

/* Resource Constraints */
x1111 + x2111 + x3111 ≤ 1; /* Mmult1 */
x1212 + x2212 + x3212 ≤ 2; /* Mmult2 */
x1122 + x2122 + x3122 ≤ 1; /* Mmult1 */
x1212 + x1223 + x2212 + x3212 + x3223 ≤ 2; /* Mmult2 */
x1133 + x3133 ≤ 1; /* Mmult1 */
x1223 + x3223 ≤ 2; /* Mmult2 */
x4122 + x5122 ≤ 1; /* Malu1 */
x4222 + x5222 ≤ 1; /* Malu2 */
x4133 + x5133 + x6133 ≤ 1; /* Malu1 */
x4233 + x5233 + x6233 ≤ 1; /* Malu2 */
x5144 + x6144 ≤ 1; /* Malu1 */
x5244 + x6244 ≤ 1; /* Malu2 */

/* Precedence Constraints */
4 x6144 + 4 x6244 + 3 x6133 + 3 x6233 - 3 x1133 - 3 x1223 - 2 x1122 - 2 x1212 - x1111 ≥ 1;
4 x6144 + 4 x6244 + 3 x6133 + 3 x6233 - 3 x4133 - 3 x4233 - 2 x4122 - 2 x4222 ≥ 1;
3 x4133 + 3 x4233 + 2 x4122 + 2 x4222 - 2 x2122 - 2 x2212 - x2111 ≥ 1;
4 x5144 + 4 x5244 + 3 x5133 + 3 x5233 + 2 x5122 + 2 x5222 - 3 x3133
        - 3 x3223 - 2 x3122 - 2 x3212 - x3111 ≥ 1;

/* Integer Constraints */
INT x1111, x1122, x1133, x1212, x1223, x2111, x2122, x2212, x3111, x3122, x3133,
x3212, x3223, x4122, x4133, x4222, x4233, x5122, x5133, x5144, x5222,
x5233, x5244, x6133, x6144, x6233, x6244;
```

Figure 5.11. ILP Formulation for Example DFG (MVMC)

```
    /* ILP Formulation for Peak Power Minimization for MVMC scheme */

  param TASK;                          # Number of Tasks
  param LEVEL;                         # Number of Levels in DFG
  param VOLT;                          # Number of Voltage Levels
  param ASAP {1..TASK} > 0;            #ASAP Schedule for each Task
  param ALAP {1..TASK} > 0;            #ALAP Schedule for each Task
  param OP {1..TASK};                  #Type of Functional Unit
  param M {1..2, 1..VOLT};             #Resource Constraints
  param POWER {1..2, 1..VOLT};         #Power consumption of the Functional Unit

var PP;
var X {i in 1..TASK, v in 1..VOLT, j in ASAP[i]..ALAP[i], k in ASAP[i]..ALAP[i]} binary;

#Objective Function
minimize peak_power: PP;
# Uniqueness Constraints
subject to uniq_cons {i in 1..TASK}:
    sum{j in ASAP[i]..ALAP[i]} X[i, 1, j, j] + (if OP[i] = 2 then sum{j in ASAP[i]..ALAP[i]-1}
       X[i, 2, j, j+1] else sum{j in ASAP[i]..ALAP[i]} X[i, 2, j, j]) = 1;
# Precedence Constraints
subject to pred_cons1:
    sum {v in 1..VOLT, j in ASAP[6]..ALAP[6]} j * X[6, v, j, j] - sum {j in ASAP[1]..ALAP[1]} j
       * X[1, 1, j, j] - sum {j in ASAP[1]..ALAP[1]-1} (j+1) * X[1, 2, j, j+1] ≥ 1;
subject to pred_cons2:
    sum {v in 1..VOLT, j in ASAP[6]..ALAP[6]} j * X[6, v, j, j] - sum {v in 1..VOLT,
       j in ASAP[4]..ALAP[4]} j * X[4, v, j, j] ≥ 1;
subject to pred_cons3:
    sum {v in 1..VOLT, j in ASAP[4]..ALAP[4]} j * X[4, v, j, j] - sum {j in ASAP[2]..ALAP[2]} j
       * X[2, 1, j, j] - sum {j in ASAP[2]..ALAP[2]-1} (j+1) * X[2, 2, j, j+1] ≥ 1;
subject to pred_cons4:
    sum {v in 1..VOLT, j in ASAP[5]..ALAP[5]} j * X[5, v, j, j] - sum {j in ASAP[3]..ALAP[3]} j
       * X[3, 1, j, j] - sum {j in ASAP[3]..ALAP[3]-1} (j+1) * X[3, 2, j, j+1] ≥ 1;
# Resource Constraints
subject to res_cons_mult {j in 1..LEVEL, v in 1..VOLT}:
    if v = 1 then sum {i in 1..TASK: ASAP[i] ≤ j ≤ ALAP[i] && OP[i] = 2} X[i, 1, j, j]
    else sum {i in 1..TASK: ASAP[i] < j < ALAP[i] && OP[i] = 2} (X[i, 2, j-1, j] + X[i, 2, j, j+1]) +
    sum {i in 1..TASK: ALAP[i] = j && OP[i] = 2} X[i, 2, j-1, j] + sum {i in 1..TASK: ASAP[i] = j
       && OP[i] = 2} X[i, 2, j, j+1] ≤ M[2, v];
subject to res_cons_alu {j in 1..LEVEL, v in 1..VOLT}:
    sum {i in 1..TASK: ASAP[i] ≤ j ≤ ALAP[i] && OP[i] = 1} X[i, v, j, j] ≤ M[1, v];
# Peak Power Constraints
subject to pp_cons {j in 1..LEVEL-1}:
  sum {i in 1..TASK: ASAP[i] ≤ j ≤ ALAP[i]} X[i, 1, j, j] * POWER[OP[i], 1]
  + sum {i in 1..TASK: ASAP[i] < j < ALAP[i] && OP[i] = 2} (X[i, 2, j-1, j]
  * POWER[OP[i], 2] + X[i, 2, j, j+1] * POWER[OP[i], 2])
  + sum {i in 1..TASK: j = ALAP[i] && OP[i] = 2} X[i, 2, j-1, j] * POWER[OP[i], 2]
  + sum {i in 1..TASK: ASAP[i] = j && OP[i] = 2} X[i, 2, j, j+1] * POWER[OP[i], 2]
  + sum {i in 1..TASK: ASAP[i] ≤ j ≤ ALAP[i] && OP[i] = 1} X[i, 2, j, j] * POWER[OP[i], 2] ≤ PP;
```

Figure 5.12. ILP Formulation for Example DFG (MVMC) in AMPL

5.5.3 Experimental Results

The ILP based MVDFC and MVMC schedulers were tested with five benchmark circuits : Example circuit (exp), FIR filter, IIR filter, HAL differential equation solver, and Auto-Regressive filter (arf). The functional units used are ALUs and multipliers. The characterised datapath cells are used from [55]. The scheduling algorithms were tested using the different sets of resource constraints (RC1, RC2, RC3, RC4, RC5) shown in Table 5.7. The experimental results for various benchmark circuits are reported in Table 5.8 for both MVDFC and MVMC case. The power estimation includes the power consumption of the overheads, such as level converters (used in both MVDFC and MVMC schemes) and dynamic clocking units (needed for MVDFC case). It is assumed that each resource has equal switching activity ($\alpha_{i,c}$). The results are reported for two supply voltages and for switching $= 0.5$.

Table 5.7. Resource Constraints used for our Experiment

Resource Constraints Details				Resource Constraint Label
Multipliers		ALUs		
3.3 V	5.0 V	3.3 V	5.0 V	
2	1	1	1	RC1
3	0	1	1	RC2
2	0	0	2	RC3
1	1	0	1	RC4
2	0	0	1	RC5

To get a visual picture of the experimental results, we plotted the peak power reductions and the PDP reductions averaged over all resource constraints. Fig. 5.13 shows the average reductions for different benchmarks averaged over all resource constraints. It is obvious from the figure that the reductions are appreciable. It is observed that the power consumption increases for higher switching and decreases for lower switching activity. The peak power reductions for the proposed scheduling schemes are listed alongwith other scheduling algorithms dealing with peak power reduction in Table 5.5.3. The table is not to provide an exact comparison, but to provide a general idea of relative performances.

Table 5.8. Power Estimates for MVDFC and MVMC Scheduling Schemes

	RC	Peak Power Estimate in mW					PDP Estimates in nJ				
		P_{pS}	P_{pD}	ΔP_{pD}	P_{pM}	ΔP_{pM}	PDP_S	PDP_D	ΔPDP_D	PDP_M	ΔPDP_M
1	2	3	4	5	6	7	8	9	10	11	12
exp	1	79.2	17.3	78.2	35.6	55.1	20.3	7.8	61.9	17.0	16.1
	2	79.2	17.3	78.2	51.8	34.6	20.3	7.8	61.9	12.0	41.1
	3	79.2	17.3	78.2	34.6	56.4	20.3	7.6	62.5	15.3	24.8
	4	40.7	9.2	77.5	40.7	0	27.1	10.5	61.4	27.1	0
	5	40.7	9.2	77.5	34.6	15.1	27.1	10.5	61.4	15.1	44.3
	Average values			77.9		32.2			61.8		25.3
fair	1	79.2	17.3	78.2	40.7	48.6	56.2	21.8	61.1	51.8	7.8
	2	79.2	17.3	78.2	51.3	35.2	56.2	21.8	61.1	49.3	12.3
	3	79.2	17.3	78.2	35.6	55.1	56.2	22.0	60.9	34.3	39.0
	4	79.2	40.6	48.7	40.7	48.61	56.2	46.6	17.1	67.5	-20.1
	5	79.2	17.3	78.2	35.6	55.1	56.2	22.1	60.7	35.2	37.4
	Average values			72.3		48.5			52.2		15.3
iir	1	118.9	37.1	68.8	74.2	37.6	45.0	17.8	60.5	43.3	3.8
	2	118.9	25.9	78.2	51.9	56.4	45.0	14.4	68.0	29.8	33.8
	3	79.3	17.3	78.2	34.6	56.4	56.2	19.4	65.5	40.2	28.5
	4	80.3	29.0	63.9	56.9	29.1	56.2	34.0	39.4	60.0	6.8
	5	80.3	17.8	77.9	34.6	56.9	56.2	18.8	66.5	40.2	28.5
	Average values			73.4		47.2			60.0		20.3
hal	1	80.3	17.5	78.2	56.9	29.1	54.0	21.0	61.1	73.0	-35.2
	2	80.3	17.5	78.2	51.8	35.5	54.0	21.0	61.1	35.9	33.5
	3	80.3	17.8	77.8	35.6	55.7	54.0	20.8	61.5	42.3	21.7
	4	80.3	29.0	63.9	58.0	27.8	67.5	45.7	32.2	73.5	-8.9
	5	80.3	17.8	77.9	35.6	55.7	67.5	26.4	60.9	48.4	28.3
	Average values			75.2		40.8			55.4		7.9
arf	1	40.7	8.9	78.2	35.0	14.0	114.7	31.5	72.5	66.2	42.3
	2	40.7	8.9	78.2	35.0	14.0	114.7	31.5	72.5	66.7	41.8
	3	40.7	9.1	77.5	35.6	12.5	114.7	38.2	66.7	68.3	40.5
	4	40.7	9.1	77.5	39.6	2.7	114.7	39.0	66.0	132.9	-15.9
	5	40.7	9.1	77.5	35.6	12.5	114.7	38.2	66.7	68.3	40.5
	Average values			77.8		11.1			68.9		29.8
Overall Average				75.3		36.0			59.7		19.7

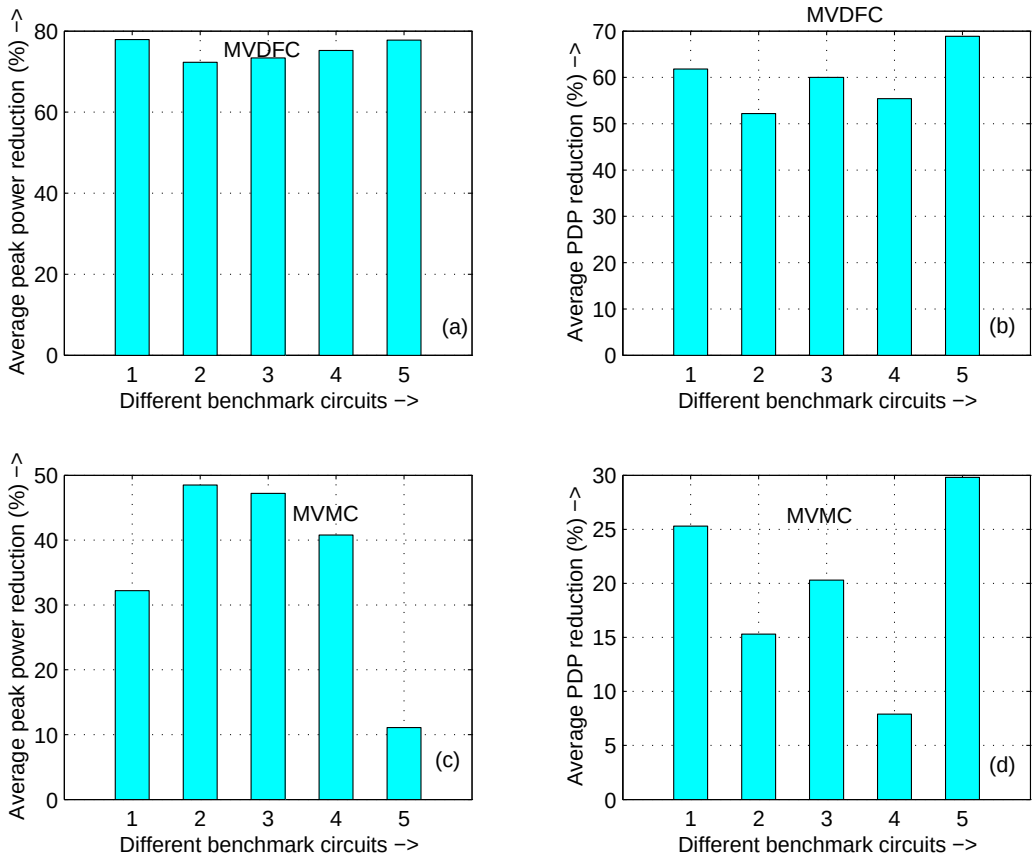


Figure 5.13. Average Reductions for Benchmarks

Table 5.9. Power Reduction for Various Scheduling Schemes

Benchmark Circuits	% Estimated average peak power reduction						
	This work				Shiue [119]	Martin [44]	Raghunathan [47]
	MVDFC		MVMC				
	ΔP_{DFC}	ΔPDP_{DFC}	ΔP_{MC}	ΔPDP_{MC}	ΔP	ΔP	ΔP
(1)exp	77.9	61.8	32.2	25.3	-	-	-
(2)fir	72.3	52.2	48.5	15.3	63.0	40.3	23.1
(3)iir	73.4	60.0	47.2	20.3	-	-	-
(4)hal	75.2	55.4	40.8	7.9	28.0	-	-
(5)arf	77.8	68.9	11.1	29.8	50.0	-	-

5.6 Conclusions

Reduction of both peak power and average power consumption of a CMOS circuit is important. This chapter addressed reduction of peak power and average power at behavioral level using low power datapath scheduling techniques. Datapath scheduling schemes, one using multiple supply voltage and dynamic clocking and another using multiple supply voltage and multicycling have been introduced. ILP based optimization techniques were used for the above two modes of datapath operations. Significant amount of peak and average power reduction over the single supply voltage and single frequency scenario could be achieved in both the cases by the proposed scheduling algorithm. The reductions attained in peak power, average power and power delay product by using combined multiple supply voltage and dynamic frequency clocking were noteworthy. *The results clearly indicate that the dynamic frequency clocking is a better scheme than the multicycling approach for power minimization.*

CHAPTER 6

ENERGY AND TRANSIENT POWER MINIMIZATION

In battery driven portable applications, the minimization of energy, average power, peak power, and peak power differential are equally important to improve reliability and efficiency. The peak power and peak power differential drive the transient characteristics of a CMOS circuit. In this chapter, we propose a framework for simultaneous reduction of the energy and transient power during behavioral synthesis. A new parameter called "Cycle Power Function" (CPF) is defined which captures the transient power characteristics as an equally weighted sum of normalized mean cycle power and normalized mean cycle differential power. Minimizing this parameter using multiple supply voltages and dynamic frequency clocking results in reduction of both energy and transient power [48]. The cycle differential power can be modeled either as the mean deviation from the average power or as the cycle-to-cycle power gradient. The switching activity information is obtained from behavioral simulations. Based on the above we develop a new datapath scheduling algorithm called CPF-scheduler which attempts at power and energy minimization by minimizing the CPF parameter by the scheduling process. The type and number of functional units available becomes the set of resource constraints for the scheduler. Experimental results indicate that the scheduler that minimizes CPF instead of conventional energy or average power as objective function could achieve significant reductions in power and energy. The rest of the chapter is organized as follows. The derivation of the CPF function based on the two models are presented in section 6.1. The proposed scheduling algorithm are presented in section 6.2. The subsequent sections present the experimental results and conclusions.

6.1 Cycle Power Function (CPF)

In this section, we introduce the different notations and terminology required for defining the cycle power function (CPF). The CPF must be defined such that it can capture simultaneously the average power, the peak power and the peak power differential of the datapath. The peak power and peak power differential determine the transient power characteristics of the circuit. Minimization of the CPF using multiple voltages results in minimization of energy as well. The datapath is represented as a sequencing data flow graph (DFG). The notations and terminology needed for the proposed models are given in Table 6.1.

Table 6.1. List of Notataions and Terminology used in CPF Modeling

N	: total number of control steps in the DFG
O	: total number of operations in the DFG
c	: a control step or a clock cycle in the DFG
o_i	: any operation i, where $1 \leq i \leq O$,
P_c	: the total power consumption of all functional units active in control step c (cycle power consumption)
P_{peak}	: peak power consumption for the DFG equal to $max(P_c)_{\forall c}$
P	: mean power consumption of the DFG (average P_c over all control steps)
P_{norm}	: normalised mean power consumption of the DFG
DP_c	: cycle difference power (for cycle c; a measure of cycle power fluctuation)
DP_{peak}	: peak differential power consumption for the DFG equal to $max(DP_c)_{\forall c}$
DP	: mean of the cycle difference powers for all control steps in DFG
DP_{norm}	: normalised mean of the mean difference powers for all steps in DFG
CPF	: cycle power function
$FU_{k,v}$	: any functional unit of type k operating at voltage level v
FU_i	: any functional unit $FU_{k,v}$ needed by o_i for its execution ($o_i \in FU_{k,v}$)
$FU_{i,c}$	: any functional unit FU_i active in control step c
R_c	: total number of functional units active in step c (same as the number of operations scheduled in c)
$\alpha_{i,c}$	: switching activity of resource $FU_{i,c}$
$V_{i,c}$	: operating voltage of resource $FU_{i,c}$
$C_{i,c}$	: load capacitance of resource $FU_{i,c}$
f_c	: frequency of control step c

The CPF is defined to consist of two main components: the normalized mean cycle power and the normalized mean cycle difference power. The normalized mean cycle power (P_{norm}) is the mean cycle power (P) normalized with respect to the peak power consumption (P_{peak}) of the

DFG. The normalized mean cycle difference power (DP_{norm}) is the mean cycle difference power (DP) normalized with respect to the peak power differential of the DFG. The second component varies between the two models. The mean difference power is the mean of the cycle difference power DP_c over the control steps. In model 1, the cycle difference power DP_c is defined as the absolute deviation of the cycle power from the mean cycle power. Then, the mean cycle difference power DP is the mean deviation of the cycle power from the mean cycle power. On other hand, in model 2, the cycle difference power DP_c of a current cycle is modelled as the cycle-to-cycle power gradient. In other words, the cycle difference power DP_c of a current control step c is the difference (or gradient) of the current cycle power and the previous cycle power. This can be expressed mathematically as, $DP_c = P_c - P_{c-1}$ or $DP_{c+1} = P_{c+1} - P_c$. In this case, the mean cycle difference power DP is the mean difference (or the gradient). The two models are further elaborated and used in defining the CPF.

6.1.1 Model 1 : CPF using Mean Deviation

For a set of n observations, $x_1, x_2, ..., x_n$ from a given distribution, the sample mean (which is an unbiased estimator for the population mean, μ) is $m = \frac{1}{n}\sum_{i=1}^{n} x_i$. The absolute deviation of these observations is defined as $\Delta x_i = |x_i - m|$. The mean deviation of the observations is given by $MD = \frac{1}{n}\sum_{i=1}^{n} |x_i - m|$. In this case, we model the cycle difference power DP_c as the absolute deviation of cycle power P_c from the mean cycle power P. Similarly, the mean difference power DP is modelled as mean deviation of the cycle power P_c. The mean cycle power P is an unbiased estimate of the average power consumption of the DFG.

The power consumption for any control step c is given by Eqn. 6.1. This is the total power consumption of all functional units active in control step c. This also includes the power consumption of the level converters where the level converters are considered as resources operating in a cycle c, if the current resource is driven by a resource operating at lower voltage.

$$P_c = \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c \tag{6.1}$$

The peak power consumption of the DFG is the maximum power consumption over all the N control steps which can be expressed as below.

$$P_{peak} = max(P_c)_{\forall c=1,2,....N} = max\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right)_{\forall c=1,2,....N} \quad (6.2)$$

The mean cycle power consumption of the DFG (P) is defined as,

$$P = \frac{1}{N}\sum_{c=1}^{N} P_c = \frac{1}{N}\sum_{c=1}^{N}\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right) \quad (6.3)$$

The mean cycle power P is an unbiased estimate of the average power consumption of the DFG. The true average power consumption of the DFG is the total energy consumption of the DFG per clock cycle or per second. The normalised mean cycle power (P_{norm}) is obtained by dividing P by maximum cycle power (P_{peak}).

$$P_{norm} = \frac{P}{P_{peak}} = \frac{\frac{1}{N}\sum_{c=1}^{N}\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c}{max\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right)_{\forall c=1,2,....N}} \quad (6.4)$$

Thus, the normalised mean cycle power (P_{norm}) is an unitless quantitity in the range [0,1].

The cycle difference power (DP_c) for any control step can be defined as follows. This is the absolute deviation of the cycle power from the mean cycle power consumption of the DFG. This is a measure of the cycle power fluctuation of the DFG.

$$DP_c = |P - P_c| = \left|\frac{1}{N}\sum_{c=1}^{N}\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right) - \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right| \quad (6.5)$$

The peak differential power which characterizes the maximum power fluctuation of the DFG is given by (DP_{peak}). This characterizes the maximum power fluctuation or the transient of the DFG over the entire set of control steps.

$$\begin{aligned} DP_{peak} &= max\left(|P - P_c|\right)_{\forall c=1,2,....N} \\ &= max\left(\left|\frac{1}{N}\sum_{c=1}^{N}\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right) - \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right|\right)_{\forall c=1,2,...N} \end{aligned} \quad (6.6)$$

The mean cycle difference power (DP) is calculated as the sample mean of DP_c. This is a measure of the power spread or distribution of the cycle power over all control steps of the DFG.

$$\begin{aligned} DP &= \frac{1}{N}\sum_{c=1}^{N} DP_c \\ &= \frac{1}{N}\sum_{c=1}^{N} |P - P_c| \\ &= \frac{1}{N}\sum_{c=1}^{N}\left(\left|\frac{1}{N}\sum_{c=1}^{N}\left(\sum_{i=1}^{R_c}\alpha_{i,c}C_{i,c}V_{i,c}^2 f_c\right) - \sum_{i=1}^{R_c}\alpha_{i,c}C_{i,c}V_{i,c}^2 f_c\right|\right) \end{aligned} \tag{6.7}$$

The normalised mean cycle difference power (DP_{norm}) can be written as given below.

$$DP_{norm} = \frac{DP}{DP_{peak}} = \frac{\frac{1}{N}\sum_{c=1}^{N}\left(\left|\frac{1}{N}\sum_{c=1}^{N}\left(\sum_{i=1}^{R_c}\alpha_{i,c}C_{i,c}V_{i,c}^2 f_c\right) - \sum_{i=1}^{R_c}\alpha_{i,c}C_{i,c}V_{i,c}^2 f_c\right|\right)}{max\left(\left|\frac{1}{N}\sum_{c=1}^{N}\left(\sum_{i=1}^{R_c}\alpha_{i,c}C_{i,c}V_{i,c}^2 f_c\right) - \sum_{i=1}^{R_c}\alpha_{i,c}C_{i,c}V_{i,c}^2 f_c\right|\right)_{\forall c}} \tag{6.8}$$

The above normalised mean cycle difference power DP_{norm} is a unitless quantity in the range [0,1].

The cycle power function CPF which is modelled as the equally weighted sum of the normalized mean cycle power (P_{norm}) and the normalized mean cycle difference power (DP_{norm}) is given below.

$$CPF(P_{norm}, DP_{norm}) = P_{norm} + DP_{norm} \tag{6.9}$$

Thus, the CPF will have a value in the range [0,2]. The CPF can be impacted by various constraints, including the resource constraints. In terms of peak cycle power (P_{peak}) and peak cycle difference power (DP_{peak}), the CPF can be expressed as :

$$CPF = \frac{P}{P_{peak}} + \frac{DP}{DP_{peak}} = \frac{\frac{1}{N}\sum_{c=1}^{N} P_c}{P_{peak}} + \frac{\frac{1}{N}\sum_{c=1}^{N}|P - P_c|}{DP_{peak}} \tag{6.10}$$

Using Eqn. 6.4 and 6.8, the cycle power function (CPF) can be written as follows.

$$CPF = \frac{\frac{1}{N}\sum_{c=1}^{N}\sum_{i=1}^{R_c}\alpha_{i,c}C_{i,c}V_{i,c}^2 f_c}{max\left(\sum_{i=1}^{R_c}\alpha_{i,c}C_{i,c}V_{i,c}^2 f_c\right)_{\forall c}} + \frac{\frac{1}{N}\sum_{c=1}^{N}\left(\left|\frac{1}{N}\sum_{c=1}^{N}\left(\sum_{i=1}^{R_c}\alpha_{i,c}C_{i,c}V_{i,c}^2 f_c\right) - \sum_{i=1}^{R_c}\alpha_{i,c}C_{i,c}V_{i,c}^2 f_c\right|\right)}{max\left(\left|\frac{1}{N}\sum_{c=1}^{N}\left(\sum_{i=1}^{R_c}\alpha_{i,c}C_{i,c}V_{i,c}^2 f_c\right) - \sum_{i=1}^{R_c}\alpha_{i,c}C_{i,c}V_{i,c}^2 f_c\right|\right)_{\forall c}} \tag{6.11}$$

6.1.2 Model 2 : CPF using Cycle-to-Cycle Gradient

For a set $x_1, x_2, ..., x_n$ of n observations from a given distribution, the observation-to-observation gradient can be defined as, $|x_{i+1} - x_i|$, where $1 \leq i \leq n-1$. The mean gradient is given by $\frac{1}{n-1}\sum_{i=1}^{n-1}|x_{i+1} - x_i|$. It should be noted that there are $n-1$ gradients for n observations. In this case, we model the cycle difference power DP_c as the cycle-to-cycle power gradient and the mean difference power DP as the mean gradient. The models for the mean cycle power or the average power (Eqn. 6.1 - 6.3) remains the same as before.

The cycle difference power (DP_c) for any control step is defined as the difference in the power consumption of the current to the previous control step, as given below.

$$DP_{c+1} = |P_{c+1} - P_c| = \left|\sum_{i=1}^{R_{c+1}} \alpha_{i,c+1} C_{i,c+1} V_{i,c+1}^2 f_{c+1} - \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right| \quad (6.12)$$

The peak differential power is characterized by (DP_{peak}) :

$$\begin{aligned} DP_{peak} &= max\left(|P_{c+1} - P_c|\right)_{\forall c=1,2,....N-1} \\ &= max\left(\left|\sum_{i=1}^{R_{c+1}} \alpha_{i,c+1} C_{i,c+1} V_{i,c+1}^2 f_{c+1} - \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right|\right)_{\forall c=1,2,...N-1} \end{aligned} \quad (6.13)$$

The mean cycle difference power (DP) is calculated as,

$$\begin{aligned} DP &= \frac{1}{N-1}\sum_{c=1}^{N-1} DP_{c+1} \\ &= \frac{1}{N-1}\sum_{c=1}^{N-1} |P_{c+1} - P_c| \\ &= \frac{1}{N-1}\sum_{c=1}^{N-1}\left(\left|\sum_{i=1}^{R_{c+1}} \alpha_{i,c+1} C_{i,c+1} V_{i,c+1}^2 f_{c+1} - \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right|\right) \end{aligned} \quad (6.14)$$

The normalised mean cycle difference power (DP_{norm}) can be written as given below.

$$DP_{norm} = \frac{DP}{DP_{peak}} = \frac{\frac{1}{N-1}\sum_{c=1}^{N-1}\left(\left|\sum_{i=1}^{R_{c+1}} \alpha_{i,c+1} C_{i,c+1} V_{i,c+1}^2 f_{c+1} - \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right|\right)}{max\left(\left|\sum_{i=1}^{R_{c+1}} \alpha_{i,c+1} C_{i,c+1} V_{i,c+1}^2 f_{c+1} - \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right|\right)_{\forall c=1,2,...N-1}} \quad (6.15)$$

Using Eqn. 6.4 and 6.15, the cycle power function (CPF) can be written as follows.

$$
\begin{aligned}
CPF &= P_{norm} + DP_{norm} \\
&= \frac{P}{P_{peak}} + \frac{DP}{DP_{peak}} \\
&= \frac{\frac{1}{N}\sum_{c=1}^{N} P_c}{P_{peak}} + \frac{\frac{1}{N-1}\sum_{c=1}^{N-1} |P_{c+1} - P_c|}{DP_{peak}} \\
&= \frac{\frac{1}{N}\sum_{c=1}^{N}\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c}{max\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right)_{\forall c}} + \frac{\frac{1}{N-1}\sum_{c=1}^{N-1}\left(\left|\sum_{i=1}^{R_{c+1}} \alpha_{i,c+1} C_{i,c+1} V_{i,c+1}^2 f_{c+1} - \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right|\right)}{max\left(\left|\sum_{i=1}^{R_{c+1}} \alpha_{i,c+1} C_{i,c+1} V_{i,c+1}^2 f_{c+1} - \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right|\right)_{\forall c=1,2,\ldots N-1}}
\end{aligned}
\tag{6.16}
$$

The above function (Eqn. 6.11 or 6.16) can be used as the objective function for low power datapath scheduling. *The minimization of this objective function using multiple supply voltages, dynamic frequency clocking and multicycling will lead to the reduction of energy and power parameters.* From the equations, 6.10, 6.11, and 6.16 we make the following observations about the cycle power function (CPF). The CPF is a *non-linear* function. It is a function of four parameters, such as, average power (P), peak power (P_{peak}), average difference power (DP) and peak difference power (DP_{peak}). Each of the above power parameters are dependent on switching activity, capacitance, operating voltage and operating frequency. The absolute function (abs or $|\ |$) in the numerator (of Eqn. 6.11 or 6.16) contributes to the nonlinearity. The complex behavior of the function is also contributed by the denominator parameters, P_{peak} and DP_{peak}.

The power models expressed in equation 6.16 and 6.11 for the CPF use generic parameters, such as $\alpha_{i,c}, C_{i,c}, V_{i,c}$ and f_c. The intention of using such paramaters is to make the CPF model a general one, independent of any specific energy or power models. It can accomodate both the look-up table based energy (power) models and energy (power) macro-models. The generic model can also help in easy integration of the CPF model in a behavioral synthesis tool that uses both behavioral power estimator and datapath scheduler. Using the dynamic energy model proposed in [51], we can express the effective switching capacitance of our proposed model as,

$$\alpha_i C_i = C_{swi}({\alpha_i}^1, {\alpha_i}^2) \tag{6.17}$$

Here, the α_i and C_i are the parameters corresponding to the functional unit FU_i. The C_{swi} is a measure of the effective switching capacitance of resource (functional unit) FU_i, which is a function of ${\alpha_i}^1$ and ${\alpha_i}^2$; where ${\alpha_i}^1$ and ${\alpha_i}^2$ are the average switching activity values on the first and second input operands of resource FU_i. It should be noted that the above switching model (in Eqn. 6.17) handles input pattern dependencies. Moreover, the generic CPF model can be easily tuned to handle any of the four modes of datapath circuit operation, such as, (i) single supply voltage and single frequency, (ii) multiple supply voltages and single frequency, (iii) multiple supply voltages and dynamic frequency and (iv) multiple supply voltage and multicycling. For example, for single supply voltage and single frequency scheme, $V_{i,c}$ and f_c are same for all c, for multiple supply voltage and multicycling f_c is same for all c. Using Eqn. 6.17 we rewrite Eqn. 6.11 as,

$$CPF = \frac{\frac{1}{N}\sum_{c=1}^{N}\sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c}{max\left(\sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c\right)_{\forall c}} + \frac{\frac{1}{N}\sum_{c=1}^{N}\left(\left|\frac{1}{N}\sum_{c=1}^{N}\left(\sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c\right) - \sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c\right|\right)}{max\left(\left|\frac{1}{N}\sum_{c=1}^{N}\left(\sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c\right) - \sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c\right|\right)_{\forall c}} \quad (6.18)$$

Similarly, using Eqn. 6.17 we rewrite Eqn. 6.16 as,

$$CPF = \frac{\frac{1}{N}\sum_{c=1}^{N}\sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c}{max\left(\sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c\right)_{\forall c}} + \frac{\frac{1}{N-1}\sum_{c=1}^{N-1}\left(\left|\sum_{i=1}^{R_{c+1}} C_{swi,c+1}V_{i,c+1}^2 f_{c+1} - \sum_{i=1}^{R_c} C_{swi,c+1}V_{i,c}^2 f_c\right|\right)}{max\left(\left|\sum_{i=1}^{R_{c+1}} C_{swi,c+1}V_{i,c+1}^2 f_{c+1} - \sum_{i=1}^{R_c} C_{swi,c+1}V_{i,c}^2 f_c\right|\right)_{\forall c=1\rightarrow N-1}} \quad (6.19)$$

The notation $C_{swi,c}$ represents C_{swi} for the functional unit FU_i active in control step c. The above two function (Eqn. 6.18 and Eqn. 6.19) are used as objective functions for our scheduling algorithm. ${\alpha_i}^1$ and ${\alpha_i}^2$ are estimated using behavioral simulation of a DFG [167, 168, 169]. A look-up table constructed to store the C_{sw} values for different combinations of (α^1 and α^2) for different types of functional units, such as multipliers and ALUs. We use interpolation technique to determine the C_{sw} values for the (α^1 and α^2) combinations that are not available in the look-up table.

6.2 CPF-Scheduler Algorithm

In this section, we develop a scheduling algorithm that minimizes the objective functions (Eqn. 6.18 or 6.19) using multiple voltages and dynamic clocking to reduce energy and the powers.

We assume the availability of different functional units operating at different supply voltages. In dynamic frequency clocking or frequency scaling, all the units are clocked by a single clock line which can switch frequencies at run-time [60, 62, 63]. In such systems, a dynamic clocking unit (DCU) generates different clocks using a clock dividing strategy. It should be noted that frequency scaling helps in reducing power, but not energy. Moreover, the frequency reduction facilitates the the operations of the different functional units at different voltages, which in turn helps in energy reduction.

The target architecture model assumed for the scheduling is from [65]. Each functional unit is associated with a register and a multiplexor. The register and the multiplexor will operate at the same voltage level as that of the functional units. Level converters are used when a low-voltage functional unit is driving a high-voltage functional unit [65, 95]. A controller decides which of the functional units are active in each control step and those that are not active are disabled using the multiplexors. The controller will have a storage unit to store the cycle frequency index (cfi_c) values obtained from the scheduling, used as the clock dividing factor for the dynamic clocking unit. The cycle frequency f_c is generated dynamically and a corresponding functional unit is activated.

The delay for a control step is dependent on the delays of the functional units (d_{FU}), multiplexor (d_{Mux}), register (d_{Reg}) and level converters (d_{Conv}) as expressed in following equation.

$$d_c = d_{FU} + d_{Mux} + d_{Reg} + d_{Conv} \tag{6.20}$$

where, d_c is the delay of control step c, d_{FU} is the delay of the slowest FU in the control step c and the register delays include the set-up and propagation delays. Using the above delay model, the worst case delays of the library components are estimated. For a given base frequency (f_{base}), maximum frequencies of each FU are scaled down to operating frequencies (f_c). These parameters are determined as follows :

$$\begin{aligned} f_{base} &= \left\lfloor \frac{\lfloor 1/d_c^{min} \rfloor}{2^{L_f}} \right\rfloor 2^{L_f} \\ cfi_c &= \left\lceil \frac{\lfloor d_c/d_c^{min} \rfloor}{2^n} \right\rceil 2^n \\ f_c &= \frac{f_{base}}{cfi_c} \end{aligned} \tag{6.21}$$

Input	: UDFG, resource constraints, L_V, L_f, all $V_i \in V_{L_V}$, d_{FU}, d_{Mux}, d_{Reg}, d_{Conv}
Output	: scheduled DFG, f_{base}, N, cfi_c, power, energy and delay estimates
Step 1	: Calculate the switching activity at the inputs of each node through behavioral simulation of the DFG.
Step 2	: Construct a look-up table of effective switching capacitance, switching activity pairs.
Step 3	: Find ASAP and ALAP schedules of the UDFG.
Step 4	: Determine the number of multipliers and ALUs at different operating voltages.
Step 5	: Modify both ASAP and ALAP schedules obtained in Step 1 using the number of resources found in Step 2 as initial resource constraint.
Step 6	: Calculate the total number of control steps which is the maximum of ASAP and ALAP schedules from Step 5.
Step 7	: Find the vertices having non-zero mobility and vertices with zero mobility.
Step 8	: Use the CPF-Scheduler-Heuristics to assign the time stamp and operating voltage for the vertices, and the cycle frequencies such that CPF and time penalty are minimum.
Step 9	: Find base frequency f_{base} and cycle frequency index cfi_c.
Step 10	: Calculate power, energy and delay details.

Figure 6.1. The CPF-Scheduler Algorithm Flow

where, d_c^{min} is the minimum of the control step delays and L_f is the number of allowable frequencies. The value of n is chosen in such a way that cfi_c is closest value greater than or equal to $\lceil \frac{d_c}{d_c^{min}} \rceil$.

The inputs to the algorithm are an unscheduled data flow graph (UDFG), the resource constraints, the number of allowable voltage levels (L_V), the number of allowable frequencies (L_f), delay of each resource (d_{FU}), multiplexor (d_{Mux}), register (d_{Reg}) at different voltage levels. The delays of level converters (d_{Conv}) are represented in the form of a matrix that shows the delay for converting one voltage level V_i to another voltage level V_j (where, both $V_i, V_j \in V_{L_V}$). The resource constraint includes the number of ALUs and multipliers at different voltage levels V_i (where, $V_i \in V_{L_V}$). The scheduling algorithm determines the proper time stamp for each operation, f_{base}, cfi_c and the voltage level such that the objective function in Eqn. 6.18 or 6.19 as well as the time penalty is minimum. To reduce the time penalty, the lesser energy consuming resources are used at as maximum frequency as possible.

The CPF-Scheduler : The flow of the proposed algorithm is outlined in Fig. 6.1. In step 1, the switching activities at the inputs of each node are determined using behavioral simulation of the DFG. For this purpose, different sets of application specific input vectors (having different

```
CPF-Scheduler-Heuristic
{
(01) Initialize CurrentSchedule as modified ASAPSchedule ;
(02) while( all mobile vertices are not time stamped ) do
(03) {
(04)    for the CurrentSchedule
(05)    {
(06)       if ( v_i is a multiplication ) then
           Find the lowest available voltage for multipliers;
(07)       if ( v_i is add/sub/comparison ) then
           Find the highest available operating voltage for ALUs;
(08)    } /* end for (04) */
(09)    Find CPF for CurrentSchedule and denote is as CurrentCPF ;
(10)    Find R_T for CurrentSchedule and denote is as CurrentR_T;
(11)    Maximum = −∞ ;
(12)    for each mobile vertex v_i
(13)    {
(14)       c1 = CurrentSchedule[v_i]; c2 = ALAPSchedule[v_i];
(15)       for c = c1 to c2 in steps of 1
(16)       {
(17)          Find a TempSchedule by adjusting CurrentSchedule in which v_i
                        is scheduled in step c ;
(18)          Find next higher operating voltage for multiplication vertex for the TempSchedule
                        (next lower for ALU operation) ;
(19)          Find CPF for TempSchedule, denoted by TempCPF ;
(20)          Find R_T for TempSchedule, denoted TempR_T;
(21)          Difference = (CurrentCPF + CurrentR_T) − (TempCPF + TempR_T) ;
(22)          if ( Difference > Maximum ) then
(23)          {
(24)             Maximum = Difference ;
(25)             CurrentVertex = v_i ;
(26)             CurrentCycle = c ;
(27)             CurrentVoltage = Operating voltage of v_i
(28)          } /* end if (22) */
(29)       } /* end for (15) */
(30)    } /* end for (12) */
(31)    Adjust CurrentSchedule to accomodate CurrentVertex in Currentcycle operating
                        at voltage assigned above ;
(32) } /* end while (02) */
} /* End CPF-Scheduler-Heuristic */
```

Figure 6.2. The CPF-Scheduler Algorithm Heuristic

correlations) are given at the primary inputs of the DFG and the average swtiching activity at each node is calculated [167, 168, 169]. In step 2, the scheduler constructs a look-up table with effective switching capacitance and the average switching activity pair as described in Eqn. 6.17. The size of the look-up table impact the accuracy of the results. If the look-up table is large enough to contain the switching capacitance for all estimated average swtiching activities in step 1, then the power model accuracy is the highest. The scheduler uses interpolation techniques to find the switching capacitance for a pair of input average swtiching activity that does not exist in the look-up table. The algorithm determines the as-soon-as-possible (ASAP) and the as-late-as-possible (ALAP) schedules for the UDFG in step 3. The ASAP schedule is unconstrained and the ALAP schedule uses the number of clock steps found in the ASAP schedule as the latency constraint. In step 4, the number of resources of each type and voltage levels is determined. For example, if the resource constraint is 1 multiplier at $2.4V$, 2 multipliers at $3.3V$, 2 ALUs at $2.4V$ and 3 ALUs at $3.3V$, then the relaxed voltage initial resource constraint is found out to be 3 multipliers and 5 ALUs. In step 5, the scheduler uses the above relaxed voltage resource constraints and modifies the ASAP and ALAP schedules to take into account the resource constraints. This helps in restricting the mobility of vertices to a great extent and reducing the solution search space for the heuristic. Due to the resource constraints the number of control steps of modified ASAP and modified ALAP may be different from that of the ASAP and ALAP schedule in step 3. In step 6, the scheduler fixes the total number of control steps of the schedule which is the maximum of the control steps of the modified ASAP or modified ALAP in step 5. In step 7, the vertices are marked as having zero mobility or non-zero mobility. The zero mobility vertices are those having same modified ASAP time stamp and modified ALAP time stamp, and non-zero mobility vertices are those having different modified ASAP and modified ALAP time stamp. On determining the vertices having zero mobility and vertices having non-zero mobility, proper time stamp and operating voltage for mobile vertices, and operating voltages for non-mobile vertices are found out. Further, operating clock frequencies are established such that the CPF as well as the time penalty is minimum. The CPF-Scheduler uses an heuristic algorithm for the same. In step 9, the scheduler determines the base frequency (f_{base}) and cycle frequency index (cfi_c) using Eqn. 6.21. In step 10, the scheduler

calculates the peak power, average power, peak power differential, energy estimates of the scheuled DFG and also the critical path delay.

The CPF-Scheduler Heuristic : Fig. 6.2 shows the heuristic algorithm used by the CPF-Scheduler. The inputs to the CPF-Scheduler heuristic are modified ASAP time stamp of each vertex (S_i), the modified ALAP time stamp of each vertex (E_i), the resource constraints, the number of allowable voltage levels (L_V), the number of allowable frequencies (L_f). Delay of each functional unit (d_{FU}), multiplexor (d_{Mux}), register (d_{Reg}) at different voltage levels are also given as inputs. Delays of level converters (d_{Conv}) is represented in the form of a matrix. The heuristic has to find time stamp c (in the range $[S_i, E_i]$) and operating voltage $V_{i,c}$ for each vertex v_i with operation o_i. The aim of the heuristic is to minimize CPF as described in Eqn. 6.18 and 6.19 while keeping time penalty at a minimum. The heuristic minimized time ratio R_T alongwith CPF to minimize the time penalty. The time ratio (R_T) is defined as the ratio between the critical path delay when the vertices of the DFG are operating at multiple voltage (T_D) and when each of the vertices of the DFG is operated at the highest voltage. Expressing mathematically, $R_T = \frac{T_D}{T_S}$. These two objectives, minimization of CPF (minimization of energy and power) and minimization of time penalty are mutually conflicting. This is due to the fact that if operating voltage is reduced to minimize energy / power consumption this results in increase of critical path delay and hence increase of time penalty. The heuristic operates the energy hungry functional units at the highest possible voltage (frequency) and the less energy consuming functional units at lowest voltage (frequency) to achieve the simultaneous minimization of the mutually conflicting objectives. The heuristic fixes operating voltages of the non-mobile vertices as per this order depending on the types of resource they need. The heuristic attempts to find suitable time stamp and operating voltage for the mobiles vertices using exhaustive search. The mobiles-vertices are attempted to be placed in each of the time stamps within their mobile range ($[S_i, E_i]$), when each placement and voltage assignment is done, the CPF and R_T value is calculated. The predecessor and successor time stamps are adjusted accordingly to maintain the precedence. For this purpose the heuristic maintains a matrix of dimension ($N * |k|V_{L_V}$) having number of resources of different types (k) as entries rowwise over all control steps. The $|k|$ is the type of resources available, for example, if only multiplier and

ALUs are the available resources then the $|k| = 2$. If a voltage is assigned for a vertex, then the matrix entry of the corresponding type and operating voltage is decremented. A particular vertex is placed in a cycle for which the sum of CPF and R_T is minimum. The heuristic, initially assumes the modified ASAP schedule (with relaxed voltage resource constrained) as the current schedule (line 01). In case a vertex is a multiplication operation, then the initial voltage assignment is the minimum available operating depending on the number of multipliers, whereas, for ALU operations vertex, it is the maximum available operating voltage (line 04-08). Then the CPF and R_T value for the current schedule is calculated (line 09 and line 10). The heuristic finds CPF (and R_T) values for each allowable control step of each mobile vertices and for each available operating voltages denoted as TempCPF (and TempR_T) (line 17-20). The statement in line 17 adjusts the current schedule by adjusting the time stamps of successor vertices while maintaining the resource constraint (using the matrix) and guaranting that the precedence is satisfied. In line 12, the vertices are visited in ASAP manner. Another possible way of visiting the mobile vertices is to prioritise them in some manner, say vertex with lower mobility is visited first. The heuristic fixes the time step and operating voltage for a vertex and hence cycle frequency for which $CPF + R_T$ is minimum (line 22-26). For CPF computation the heuristics uses $\frac{1}{d_c}$ as a temporary measure for f_c. The above steps are repeated until all mobile vertices are time stamped.

Time complexity of CPF-Scheduler Heuristic : Let there be $|V|$ number of vertices in the DFG, out of which $|V_m|$ number of vertices have mobility and the maximum mobility of any mobile vertex is t_m. It should be noted that the total number of vertices in the DFG is total number of operations in DFG and the total number of NO-OPs. The running time of finding an operating voltage from the matrix for particular type of operation is $O(L_V)$. The statements from line 04-08 have running time of $\Theta(|V|L_V)$. The worst case running time of the statement in line 17 (or line 31) that adjusts the current schedule is $O(|V_m|)$. The running time of the code segment between line 17-28 is $O(|V_m|) + O(L_V) + \Theta(|V|) + \Theta(|V|)$, which is $\Theta(|V|)$, since it is always true that $|V_m|, L_V < |V|$. So, the running time of the code segment from line 15-29 is $\Theta(t_m|V|)$. Thus, the running time of the code segment line 12-30 is $\Theta(t_m|V_m||V|)$. The other statements of the pseudocode have constant running time. So, the running time or time complexity of the code

segment in line 03-29 is $\Theta(|V||L_V|) + \Theta(t_m|V_m||V|) + O(|V_m|)$. This can be simplified to an weak upper bound on worst case running of the code segment (line 03-31) under the assumption that $|V_m| \approx |V|$, but in practice $|V_m| << |V|$. Under the above assumption we conclude that the worst case upper bound on the running time of the code segement in line 03-31 is $\Theta(t_m|V|^2)$. Considering the *while* loop in line 02 the overall running time of the algorithm can be written as $\Theta(t_m|V|^2|V_m|)$. Again under the assumption that $|V_m| \approx |V|$, we conclude that the worst case upper bound on the running time of the algorithm is $\Theta(t_m|V|^3)$. In other words, *the heuristic runs in time cubic to the number of vertices in the DFG. It can be noted that the time complexity of the algorithm is independent of the number of operating voltage levels.*

6.3 Experimental Results

The CPF-Scheduler algorithm was implemented in C and tested with selected benchmark circuits. The benchmarks used are given below.

- Auto-Regressive filter (ARF) (total 28 nodes, 16*, 12+, 40 edges).
- Band-Pass filter (BPF) (total 29 nodes, 10*, 10+, 9-, 40 edges).
- DCT filter (total 42 nodes, 13*, 29+, 68 edges).
- Elliptic-Wave filter (EWF) (total 34 nodes, 8*, 26+, 53 edges).
- FIR filter (total 23 nodes, 8*, 15+, 32 edges).
- HAL differential equation solver (total 11 nodes, 6*, 2+, 2-, 1<, 16 edges).

Our algorithm can handle large DFGs and find solutions in reasonable time. The parameters used to express our experimental results are shown in Table 6.2.

We use a look-up table method as discussed in Section 6.1 for average switching capacitance calculation. The look-up table construction consists of two phases, such as input pattern generation and cell characterization. We generate the primary input signals of different correlations, using the autoregressive moving average (ARMA) model [169]. We perform the characterization of the

Table 6.2. Notations used to Express the Results

E_S	: total energy consumption assuming single frequency and single supply voltage
E_D	: total energy consumption for dynamic clocking and multiple supply voltage
P_{pS}	: peak power consumption for single frequency and single supply voltage
P_{pD}	: peak power consumption for dynamic clocking and multiple supply voltage
P_{mS}	: minimum power consumption for single frequency and single supply voltage
P_{mD}	: minimum power consumption for dynamic clocking and multiple supply voltage
T_S	: execution time assuming single frequency
T_D	: execution time assuming dynamic frequency
ΔE	: total energy reduction $= \frac{E_S - E_D}{E_S}$
ΔP	: average power reduction $= \frac{(E_S/T_S)-(E_D/T_D)}{(E_S/T_S)}$
ΔP_p	: peak power reduction $= \frac{P_{pS}-P_{pD}}{P_{pS}}$
ΔDP	: differential power reduction $= \frac{(P_{pS}-P_{mS})-(P_{pD}-P_{mD})}{(P_{pS}-P_{mS})}$
R_T	: time ratio $= \frac{T_D}{T_S}$

physical implementations of the library modules available in [55] by applying the input patterns generated as above for the values of $({\alpha_i}^1, {\alpha_i}^2)$ pairs in the table. We used interpolation to find the average switching capacitance for any of $({\alpha_i}^1, {\alpha_i}^2)$ pairs that do not exist in the look-up table. It should be noted that larger the size of look-up table, better is the accuracy. Our look-up table has 100 pairs of entries for $({\alpha_i}^1, {\alpha_i}^2)$. The signals are propagated through different operators in the DFG and the average switching activities are calculated as described in [169] for each node.

Our first set of experiments were carried out for the CPF model 1 (Eqn. 6.18) in which the cycle difference power is based on the absolute deviation. We tested the scheduling algorithm using the following sets of resource constraints (RC1, RC2, RC3, RC4).

Number of multipliers : 1 at $2.4V$ and Number of ALUs : 1 at $3.3V$
Number of multipliers : 2 at $2.4V$ and Number of ALUs : 1 at $3.3V$
Number of multipliers : 2 at $2.4V$ and Number of ALUs : 1 at $2.4V$ and 1 at $3.3V$
Number of multipliers : 1 at $2.4V$ and 1 at $3.3V$; Number of ALUs : 1 at $2.4V$ and 1 at $3.3V$

The sets of resource constraints was chosen so as to cover resources at different operating voltages. The number of allowable voltage levels was assumed to be two $(2.4V, 3.3V)$ and maximum number of allowable frequencies are three. The CPF-scheduler determines the frequencies, in this case they are $(4.5MHz, 9.0MHz, 18.0MHz)$. The experimental results for different benchmarks are

Table 6.3. Power Estimates for Different Benchmarks (using Model 1)

CKT	Power reduction details, Energy savings, Number of clock cycles and Time penalty										
	RC	P_{pS} (mW)	P_{pD} (mW)	ΔP_p (%)	P_{mS} (mW)	P_{mD} (mW)	ΔDP (%)	ΔP (%)	ΔE (%)	N	r_T
1	2	3	4	5	6	7	8	9	10	11	12
ARF	1	9.30	2.83	69.60	0.26	0.52	74.50	71.40	47.57	18	1.6
	2	18.33	4.77	73.96	0.26	0.52	76.47	68.30	47.57	13	1.4
	3	18.59	4.84	73.96	0.26	0.52	76.44	71.72	49.87	11	1.5
	4	18.59	7.26	60.96	0.26	0.52	63.25	59.10	29.49	11	1.5
	Average values			69.62			72.67	67.63	43.62		1.5
BPF	1	9.30	2.45	73.62	0.26	0.52	78.64	65.80	46.69	17	1.3
	2	18.33	4.20	77.10	0.26	1.67	86.03	58.81	46.69	17	1.2
	3	18.59	4.84	73.96	0.52	0.97	78.59	71.09	48.61	9	1.4
	4	18.59	7.33	60.60	0.52	0.97	64.84	64.01	32.02	9	1.4
	Average values			71.32			77.02	64.93	43.50		1.3
DCT	1	9.30	2.83	69.60	0.26	0.52	74.50	50.90	42.44	29	1.1
	2	9.30	2.83	69.60	0.26	0.52	74.50	50.90	42.44	29	1.1
	3	18.59	4.84	73.96	0.26	0.40	75.75	67.70	42.93	15	1.4
	4	18.59	7.61	59.05	0.26	0.40	60.63	65.19	38.49	15	1.4
	Average values			68.05			71.35	58.67	43.58		1.2
EWF	1	9.30	2.45	73.62	0.26	0.52	78.64	41.17	44.43	27	0.9
	2	18.07	4.07	77.49	0.26	0.52	80.09	37.49	44.43	27	0.9
	3	18.07	4.07	77.49	0.26	0.40	79.38	57.89	44.73	16	1.2
	4	18.07	6.55	63.75	0.26	0.40	65.49	53.10	38.45	16	1.2
	Average values			73.09			75.90	47.41	43.01		1.1
FIR	1	9.30	2.74	70.52	0.26	0.52	75.45	58.54	46.11	15	1.3
	2	9.30	2.74	70.52	0.26	0.52	75.45	58.54	46.11	15	1.3
	3	18.59	4.77	74.32	0.26	0.40	76.12	51.21	46.77	11	1.0
	4	18.59	7.04	62.15	0.24	0.40	63.77	40.69	27.21	11	1.2
	Average values			69.38			72.70	52.25	41.55		1.2
HAL	1	9.30	2.45	73.62	0.26	1.67	91.38	72.32	50.58	7	1.6
	2	18.33	4.49	75.53	0.26	1.67	84.44	64.70	50.58	5	1.4
	3	18.33	4.49	75.53	0.52	0.97	80.27	72.48	51.84	4	1.5
	4	18.33	6.97	61.98	0.52	0.97	66.32	57.14	25.00	4	1.5
	Average values			71.67			80.60	66.66	44.50		1.5
Average values				70.52			75.04	59.59	43.29		1.3

shown in Table 6.3 for different resource constraints. The average results is shown in Fig. 6.7 for visual inspection. The results take into account the power or energy consumptions in overheads, such as level converters and dynamic clocking unit. This indicates that the scheduling scheme could achieve significant reductions in peak power, peak power differential, average power and total energy with reasonable time penalties. The time penalty for the benchmarks circuits (ARF and HAL) were relatively high. *For many cases, CPF-Scheduler could reduce energy and power even without any time penalty or even with gain in time.* This happens when the performance degradation due to multiplications in the critical path are adequately compensated by the number of ALU operations in the critical path. For this to happen, the ALU operations should be larger than or equal to the number of multiplications in the critical path. This is the case for most of the schedules obtained for the EWF and FIR benchmarks indicated by the time ratio (R_T) of less than or equal to one.

For the above experimental set up, we plotted the power consumption per cycle, over all the control steps (clock steps) for different benchmarks in Fig. 6.3,6.4, 6.5 and 6.6 for resource constraints RC1, RC2, RC3 and RC4, respectively. The curves labeled as "S" correspond to the profile when the schedule is operated at a single frequency (which is the maximum frequency of the slowest operator, the multiplier) and single voltage. The profiles labeled as "D" correspond to the case when dynamic clocking and multiple voltage scheme are used. The effectiveness of the proposed scheduling scheme is obvious from the figures. Since the CPF is a complex function consisting of several parameters, it is difficult to accurately quantify the impact of a specific parameter.

We also performed experiments with three voltage levels ($1.5V, 2.4V, 3.3V$) and four frequency levels. The results could improve within the range of $5 - 10\%$ in terms of power or energy reductions. However, the time penalty increased by 15%. It is to be noted that the number of allowable frequency levels should be as close to the number of allowable voltages in order to keep the time penalty within a reasonable limit. We performed the same set of experiments for the CPF model 2 (Eqn. 6.19) in which the cycle difference power is modeled as cycle-to-cycle power gradient. The experimental results for different benchmarks are shown in Table 6.4 for different resource constraints using model 2 and the average data presented in Fig. 6.8. The results take into account

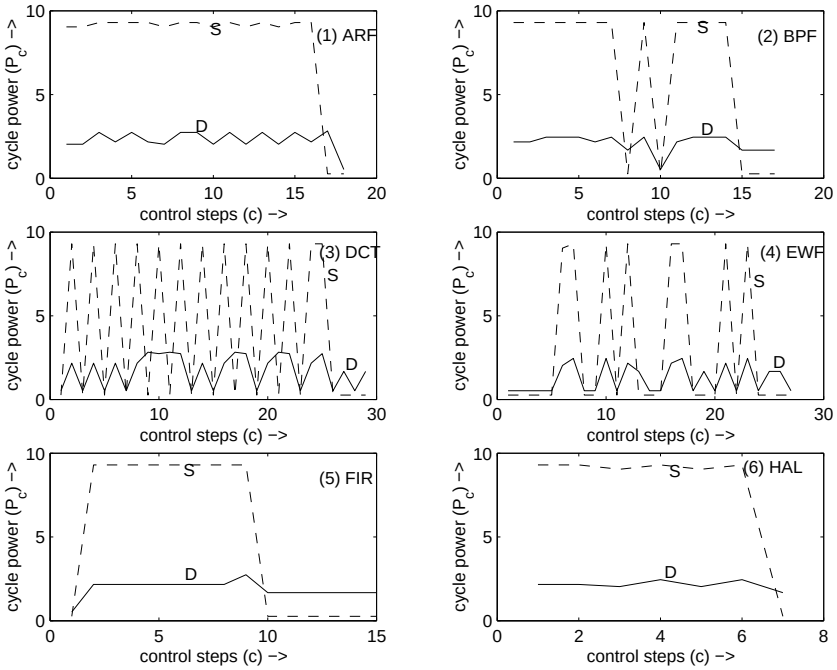


Figure 6.3. Cycle Power Consumptions for Resource Constraint RC1

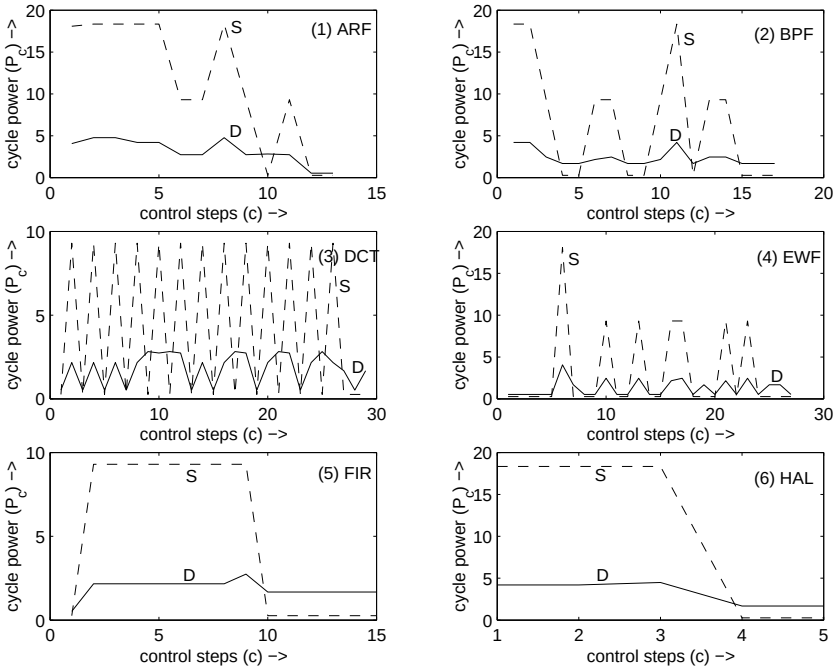


Figure 6.4. Cycle Power Consumptions for Resource Constraint RC2

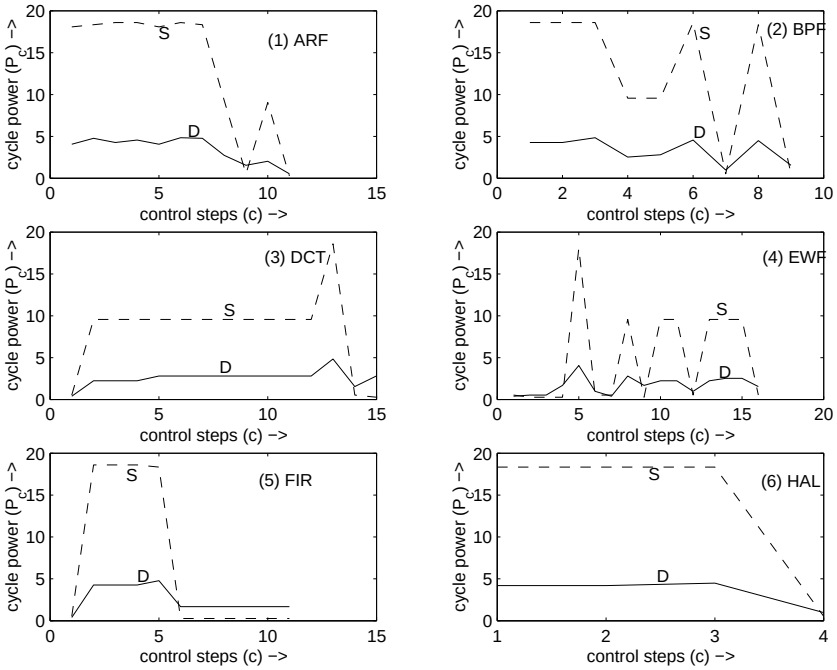


Figure 6.5. Cycle Power Consumptions for Resource Constraint RC3

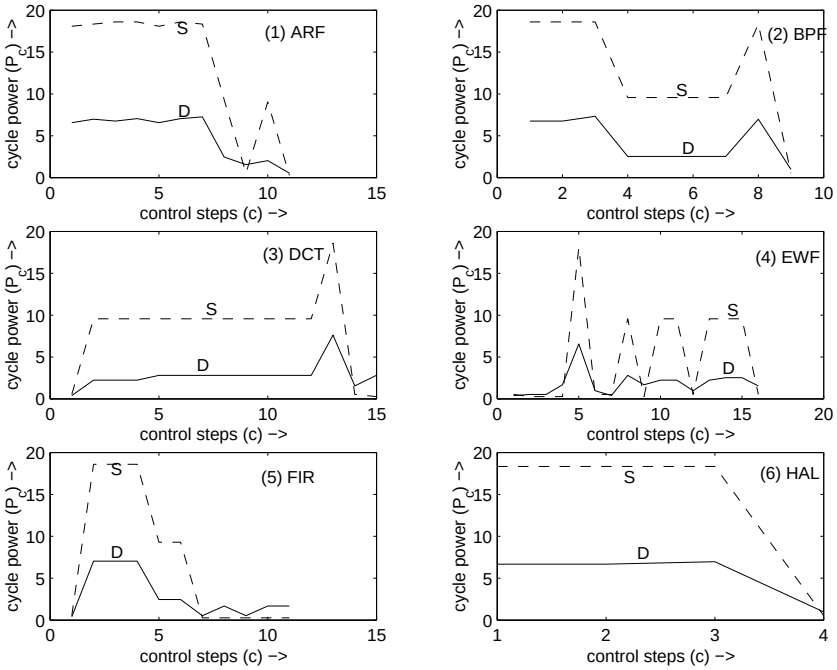


Figure 6.6. Cycle Power Consumptions for Resource Constraint RC4

Table 6.4. Power Estimates for Different Benchmarks (using Model 2)

CKT	Power reduction details, Energy savings, Number of clock cycles and Time penalty										
	RC	P_{pS} (mW)	P_{pD} (mW)	ΔP_p (%)	P_{mS} (mW)	P_{mD} (mW)	ΔDP (%)	ΔP (%)	ΔE (%)	N	r_T
1	2	3	4	5	6	7	8	9	10	11	12
ARF	1	9.30	2.64	71.58	0.26	0.52	76.54	71.99	48.64	18	1.6
	2	18.33	4.68	74.49	0.26	0.52	77.01	68.91	48.64	13	1.4
	3	18.59	4.74	74.49	0.26	0.52	76.47	71.35	49.87	11	1.5
	4	18.59	7.23	61.13	0.26	0.52	63.42	56.77	24.34	11	1.5
	Average values			70.42			73.36	67.25	42.87		1.5
BPF	1	9.30	2.40	74.15	0.26	0.52	79.18	66.48	47.74	17	1.3
	2	18.33	4.44	75.80	0.26	0.52	78.34	56.67	47.74	17	1.2
	3	18.59	4.74	74.99	0.52	1.35	81.23	73.26	49.48	9	1.4
	4	18.59	7.23	61.13	0.52	0.87	64.84	64.38	32.72	9	1.4
	Average values			71.52			78.78	65.20	44.42		1.3
DCT	1	9.30	2.64	71.58	0.26	0.52	76.54	52.25	44.02	29	1.1
	2	9.30	2.64	71.58	0.26	0.52	76.54	52.25	44.02	29	1.1
	3	18.59	4.74	74.49	0.26	0.40	76.29	68.68	44.66	15	1.4
	4	18.59	7.47	59.85	0.26	0.40	61.44	66.21	40.31	15	1.4
	Average values			69.38			72.70	59.85	43.25		1.2
EWF	1	9.30	2.40	74.15	0.26	0.52	79.18	42.22	45.43	27	0.9
	2	18.07	4.07	77.49	0.26	0.52	80.09	34.42	41.70	27	0.9
	3	18.07	4.07	77.49	0.26	0.40	79.38	55.29	41.32	16	1.2
	4	18.07	6.55	63.75	0.26	0.40	65.49	50.50	35.03	16	1.2
	Average values			73.22			76.03	45.60	40.87		1.1
FIR	1	9.30	3.01	67.62	0.26	0.52	72.46	56.30	43.27	15	1.3
	2	9.30	3.91	57.99	0.26	0.52	62.55	56.36	43.27	15	1.3
	4	18.59	5.04	72.87	0.26	0.40	74.64	48.61	48.61	11	1.0
	5	18.59	7.53	59.51	0.24	0.40	61.09	24.70	17.86	11	1.2
	Average values			64.50			69.69	46.49	38.25		1.2
HAL	1	9.30	2.40	74.15	0.26	1.48	89.75	72.62	51.11	7	1.6
	2	18.33	4.44	75.80	0.26	1.48	83.62	65.08	51.11	5	1.4
	4	18.33	4.44	75.80	0.52	0.87	79.99	72.68	52.20	4	1.5
	5	18.33	6.92	62.65	0.52	0.87	66.04	57.34	25.35	4	1.5
	Average values			72.10			79.85	66.93	44.94		1.5
Average values				70.19			75.07	58.55	42.43		1.3

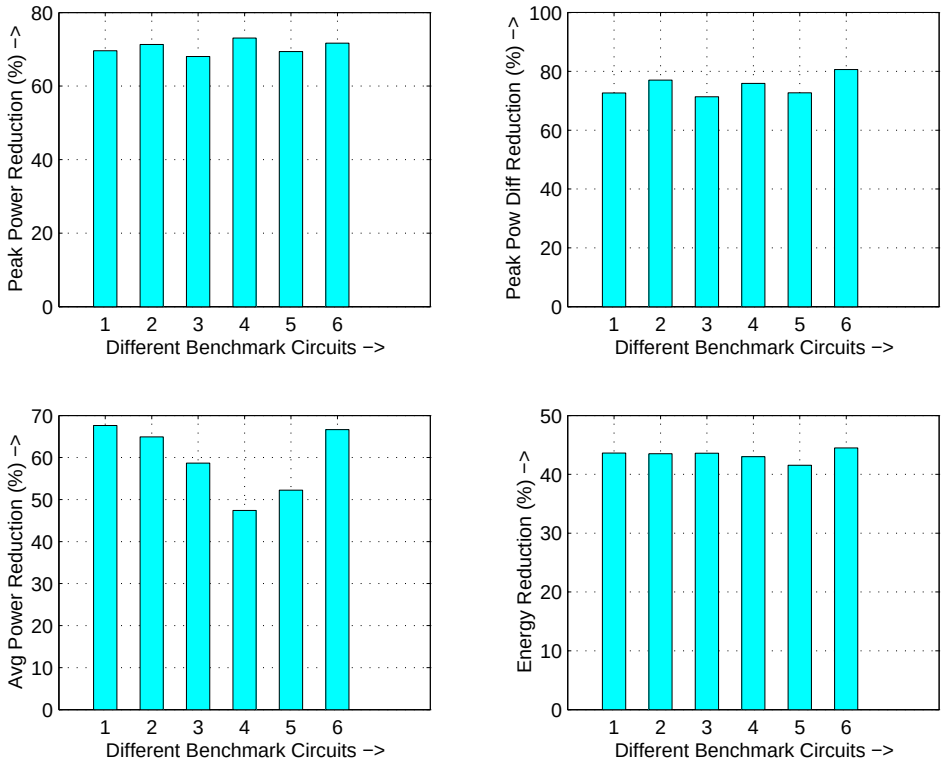


Figure 6.7. Percentage Average Reduction for Benchmarks using Model1

the power or energy consumptions due to the overheads. The results indicate that the energy and power reduction were similar with small differences, but there were no changes in terms of time penalty. We conclude that the minor difference is due to the fact that the quantitative difference between the values of $(\frac{1}{N}\sum_{c=1}^{N}|P - P_c|)$ and $(\frac{1}{N-1}\sum_{c=1}^{N-1}|P_{c+1} - P_c|)$ are not significant. We did not provide the cycle power plot for this model since it was almost the same as that of model 1.

6.4 Conclusions

For deep submicron and nanometer technology designs for low power battery driven systems, simultaneous minimization of total energy and transient power is beneficial. The CPF parameter defined and used in this work essentially facilitates such simultaneous optimization. The datapath scheduling algorithm described in this paper is particularly useful for synthesizing data intensive application specific integrated circuits. The algorithm attempts to optimize energy and power while keeping the time penalty at minimum. The CPF-Scheduler algorithm assumes the number of dif-

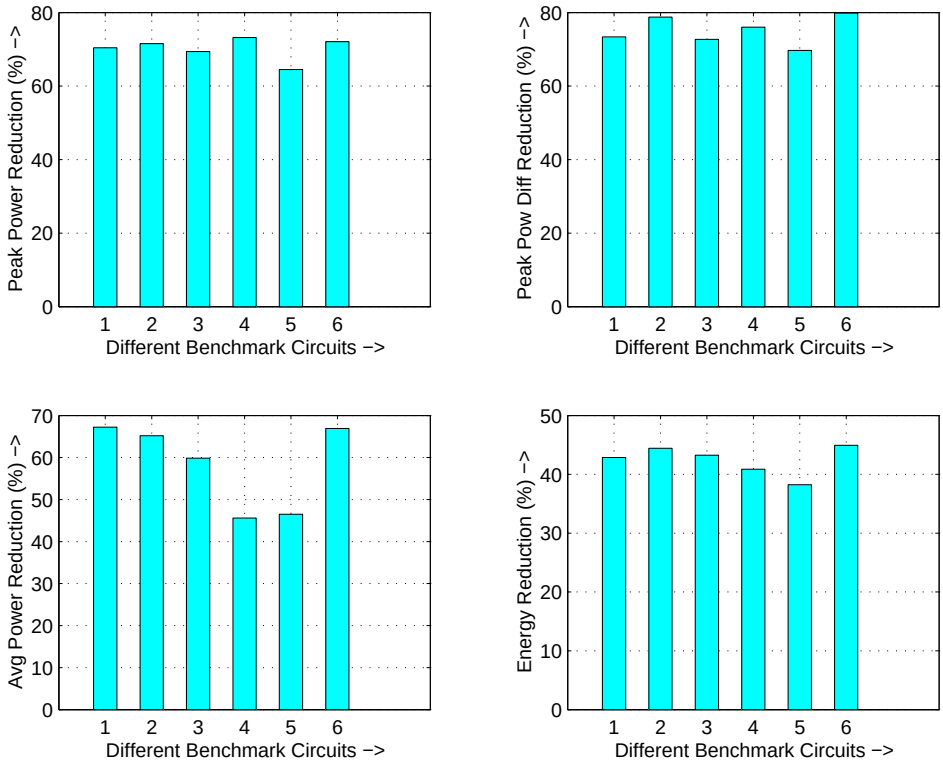


Figure 6.8. Percentage Average Reduction for Benchmarks using Model2

ferent types of resources at each voltage level and the number of allowable frequencies as resource constraints. The work provides a unified framework for simultaneous multicost space metric optimization of different energy and power components in CMOS circuit design. Future work could address leakage reduction and interconnect issues. The effectiveness of the CPF in the context of a pipelined datapath and control intensive applications also needs to be investigated.

CHAPTER 7

TRANSIENT POWER MINIMIZATION

In the previous chapter, we proposed a framework for simultaneous reduction of the four parameters through datapath scheduling. A new parameter called "cycle power function" is defined that captures the four parameters and it is minimized using heuristic based scheduling algorithm. In this chapter, we modify the non-linear CPF (denoted as CPF^*) so that integer linear programming (ILP) can be used for its minimization during datapath scheduling. The model for CPF takes into consideration the effect of switching activity on the power consumption of functional units. The first scheme, CPF-MVDFC combines both multiple supply voltages (MV) and dynamic frequency clocking (DFC) for CPF^* minimization [170], while the second scheme, CPF-MVMC uses multiple supply voltages (MV) and multicycling (MC) [171]. We conducted experiments on selected high-level synthesis benchmark circuits for various resource constraints and estimated power, energy and energy delay product for each of them. Experimental results show that significant reductions in power, energy and energy delay product can be obtained.

The rest of the chapter is organized as follows. We discuss the related works in the next section. We define, the cycle power profile function as the equally weighted sum of normalized mean cycle power and normalized mean cycle differential power followed by the analysis of the functions (CPF and CPF^*). Since, the CPF^* function is non-linear and we aim at using linear programming for its minimization, we discuss the procedures to handle standard nonlinearities using linear programming. The ILP formulations for CPF^* minimization using multiple supply voltages and dynamic frequency clocking is discussed, followed by the ILP formulations for CPF^* minimization using multiple supply voltages and multicycling. Then, we describe the ILP-based scheduling algorithm followed by the experimental results and conclusions.

7.1 Modified Cycle Power Function

In this section, we redefine the parameter called cycle power function (CPF) which captures the peak power, the peak power differential and the average power of the datapath circuit. It should be noted that CPF captures the transient power characteristics of the circuit and the minimization of CPF using multiple voltages could lead to reduction of energy. In this section, we define CPF, study its nonlinear behavior and modify it so that we can use integer linear programming (ILP) for its minimization. The datapath is represented as a sequencing data flow graph (DFG). The definitions and notations used in this chapter are the same as that of the previous chapter (Table 6.1.

Following the same steps as in the previous chapter, the cycle power function CPF is modeled as an equally weighted sum of the normalized mean cycle power (P_{norm}) and the normalized mean cycle difference power (DP_{norm}) as given below.

$$CPF(P_{norm}, DP_{norm}) \quad = \quad P_{norm} + DP_{norm} \tag{7.1}$$

The CPF has a value in the range [0,2]. In terms of peak cycle power (P_{peak}) and peak cycle difference power (DP_{peak}), CPF can be expressed as :

$$CPF \quad = \quad \frac{P}{P_{peak}} + \frac{DP}{DP_{peak}} \quad = \quad \frac{\frac{1}{N}\sum_{c=1}^{N} P_c}{P_{peak}} + \frac{\frac{1}{N}\sum_{c=1}^{N} |P - P_c|}{DP_{peak}} \tag{7.2}$$

Thus, the cycle power function (CPF) can be written as follows.

$$\begin{aligned} CPF \quad = \quad & \frac{\frac{1}{N}\sum_{c=1}^{N}\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c}{max\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right)_{\forall c}} \\ + \quad & \frac{\frac{1}{N}\sum_{c=1}^{N}\left(\left|\frac{1}{N}\sum_{c=1}^{N}\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right) - \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right|\right)}{max\left(\left|\frac{1}{N}\sum_{c=1}^{N}\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right) - \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right|\right)_{\forall c}} \end{aligned} \tag{7.3}$$

The above function (Eqn. 7.3) can serve as the objective function for low power datapath scheduling. *The minimization of this objective function using multiple supply voltages, dynamic frequency clocking and multicycling can reduce both power and energy*. From the Eqns. 7.2, and

7.3, we make following observations about the cycle power function (CPF). The CPF is a *non-linear* function. It is a function of four parameters, such as, average power (P), peak power (P_{peak}), average difference power (DP) and peak difference power (DP_{peak}). The absolute function (abs or $|\ \ |$) in the numerator (of Eqn. 7.3) contributes to the nonlinearity. The complex behavior of the function is also contributed by the denominator parameters, P_{peak} and DP_{peak}.

We need to develop scheduling algorithms that accept, an unscheduled DFG, the resource/time constraints, switching activity information, load capacitance, voltage levels and the number of allowable frequency levels as input parameters. For optimum minimization of the function, such an algorithm has to be based on non-linear optimization techniques, which are of large time and space complexity. In this work, we aim at developing integer linear programming (ILP) based model for minimization of the CPF. We alter the CPF in order to simplify the ILP-based model. It is known that the denominator parameters, P_{peak} equals to $max\left(P_c\right)_{\forall c}$ and the DP_{peak} equals to $max\left(|P - P_c|\right)_{\forall c}$. It is evident that $|P - P_c|$ is upper bounded P_c for all control steps c, since $|P - P_c|$ is a measure of mean difference error of P_c. Thus, we conclude that DP_{peak} is upper bounded by P_{peak}. We modify the CPF by substituting DP_{peak} with P_{peak} and define modified CPF (denoted as CPF^*) as follows.

$$\begin{aligned} CPF^* &= \frac{P}{P_{peak}} + \frac{DP}{P_{peak}} \\ &= \frac{P+DP}{P_{peak}} \\ &= \frac{\frac{1}{N}\sum_{c=1}^{N} P_c + \frac{1}{N}\sum_{c=1}^{N} |P-P_c|}{P_{peak}} \\ &= \frac{\frac{1}{N}\sum_{c=1}^{N}\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c}{max\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right)_{\forall c}} \\ &+ \frac{\frac{1}{N}\sum_{c=1}^{N}\left(\left|\frac{1}{N}\sum_{c=1}^{N}\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right) - \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right|\right)}{max\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right)_{\forall c}} \end{aligned} \tag{7.4}$$

Unlike CPF, the CPF^* is dependent on three factors, P, P_{peak} and DP. The absence of DP_{peak}, in the denominator helps in reducing the complexity of the ILP formulations (which will be discussed in next section) to a great extent. We minimize the "modified cycle power function" (CPF^*) instead of CPF using the ILP-based model.

The power models developed in Eqn. 7.3 for the CPF use parameters, such as $\alpha_{i,c}$, $C_{i,c}$, $V_{i,c}$ and f_c. The model can accomodate both the look-up table based energy (power) models and energy (power) macro-models. The generic model can also help in easy integration of a CPF model in behavioral synthesis tool that uses both a behavioral power estimator and a datapath scheduler. Using the dynamic energy model proposed in [123], the effective switching capacitance can be expressed as,

$$\alpha_i C_i \quad = \quad C_{swi}({\alpha_i}^1, {\alpha_i}^2) \tag{7.5}$$

Here, α_i and C_i are the parameters corresponding to the functional unit FU_i as defined before. C_{swi} is a measure of the effective switching capacitance of the functional unit FU_i, which is a function of ${\alpha_i}^1$ and ${\alpha_i}^2$; ${\alpha_i}^1$ and ${\alpha_i}^2$ are the average switching activities on the first and second input operands of resource FU_i. It should be noted that in the above switching model, (in Eqn. 7.5) the input pattern dependencies can be handled. Moreover, the generic CPF model can be easily tuned to handle any of the four modes of datapath circuit operation, such as, (i) single supply voltage and single frequency, (ii) multiple supply voltages and single frequency, (iii) multiple supply voltages and dynamic frequency and (iv) multiple supply voltage and multicycling. For the single supply voltage and single frequency scheme, $V_{i,c}$ and f_c is the same for all c, while for multiple supply voltage and multicycling f_c is same for all c. Using Eqn. 7.5, we rewrite Eqn. 7.4 as,

$$\begin{aligned} CPF^* \quad = \quad & \frac{\frac{1}{N}\sum_{c=1}^{N}\sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c}{max\left(\sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c\right)_{\forall c}} \\ + \quad & \frac{\frac{1}{N}\sum_{c=1}^{N}\left(\left|\frac{1}{N}\sum_{c=1}^{N}\left(\sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c\right) - \sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c\right|\right)}{max\left(\sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c\right)_{\forall c}} \end{aligned} \tag{7.6}$$

The notation $C_{swi,c}$ represents C_{swi} for the functional unit FU_i active in control step c. We use the above equation (Eqn. 7.6) as the objective function for our scheduling algorithm. ${\alpha_i}^1$ and ${\alpha_i}^2$ are estimated using behavioral simulation of a DFG with a set of input vectors [167, 168, 169]. A look-up table is constructed that stores the C_{sw} values for (α^1 and α^2) combinations for different types of functional units, such as multipliers and ALUs. We use interpolation to find the C_{sw} values for the (α^1 and α^2) combinations that are not available in the look-up table.

7.2 Modeling of Non-linearities

The "modified cycle power function" (CPF^*) discussed in the previous section, is a *non-linear function*. The nonlinearity is because of the *absolute function* (abs or $|\ |$) and also because of the *fractional form* of the function itself. The ILP formulations need to handle these two forms of non-linearity. We first address the transformations required to derive linear models of the nonlinear functions. Let us represent the general linear programming model as follows [172] :

$$\begin{aligned} \text{Minimize :} \quad & \textstyle\sum_j c_j * x_j \\ \text{Subject to :} \quad & \textstyle\sum_j a_{ij} * x_j \leq b_i, \quad \forall i \\ & x_j \geq 0, \quad \forall j \end{aligned} \tag{7.7}$$

where, c_j, a_{ij}, b_i are known constants and x_j are the decision variables.

7.2.1 LP Formulation Involving Sum of Absolute Deviations

The general form of this type of programming can be represented as given below [173, 174].

$$\begin{aligned} \text{Minimize :} \quad & \textstyle\sum_i |y_i| \\ \text{Subject to :} \quad & y_i + \textstyle\sum_j a_{ij} * x_j \leq b_i, \quad \forall i \\ & x_j \geq 0, \quad \forall j \end{aligned}$$

where, y_i, is the deviation between the prediction and observation. The $|y_i|$ is non-linear because of absolute function. This can be linearized using the following transformation.

Let, y_i be represented as the difference of two non-negative variables,

$$y_i \quad = \quad y_i^1 - y_i^2 \tag{7.9}$$

Using these variables, we can rewrite the LP formulation in Eqn. 7.8 as follows.

$$\begin{aligned} &\text{Minimize :} && \textstyle\sum_i \left|y_i^1 - y_i^2\right| \\ &\text{Subject to :} && y_i^1 - y_i^2 + \textstyle\sum_j a_{ij} * x_j \leq b_i, \quad \forall i \\ & && x_j \geq 0, \quad \forall j \\ & && y_i^1, y_i^2 \geq 0, \quad \forall i \end{aligned} \tag{7.10}$$

If the product of y_i^1 and y_i^2 is zero, then,

$$\begin{aligned} \left|y_i^1 - y_i^2\right| &= \left|y_i^1\right| + \left|y_i^2\right| \\ &= y_i^1 + y_i^2 \end{aligned} \tag{7.11}$$

Using the above, we can write the LP formulation expressed in Eqn. 7.10 as shown below.

$$\begin{aligned} &\text{Minimize :} && \textstyle\sum_i y_i^1 + y_i^2 \\ &\text{Subject to :} && y_i^1 - y_i^2 + \textstyle\sum_j a_{ij} * x_j \leq b_i, \quad \forall i \\ & && x_j \geq 0, \quad \forall j \\ & && y_i^1, y_i^2 \geq 0, \quad \forall i \end{aligned} \tag{7.12}$$

The formulations in Eqn. 7.8 and 7.12 are equivalent and the minimization of Eqn. 7.12 will result in the minimization of Eqn. 7.8.

7.2.2 LP Formulation Involving Fraction

The general expression for the LP formulation involving fractions is considered below [174].

$$\begin{aligned} &\text{Minimize :} && \frac{\sum_j c_j * x_j}{\sum_j d_j * x_j} \\ &\text{Subject to :} && \textstyle\sum_j a_{ij} * x_j \leq b_i, \quad \forall i \\ & && x_j \geq 0, \quad \forall j \end{aligned} \tag{7.13}$$

where, c_j and d_j are known constants and the denominator $\sum_j d_j * x_j$ is strictly positive. Let us assume new variables as follows :

$$\begin{aligned} z_0 &= \left| d_0 + \sum_j d_j * x_j \right|^{-1} \\ x_j &= \frac{z_j}{z_0} \end{aligned} \tag{7.14}$$

Using the above transformation, the original formulation in Eqn. 7.13 can be modified to the following.

$$\begin{aligned} \text{Minimize :} \quad & c_0 * z_0 + \sum_j c_j * z_j \\ \text{Subject to :} \quad & \sum_j a_{ij} * z_j - b_i * z_0 \leq b_i, \quad \forall i \\ & \sum_j d_j * z_j + d_0 * z_0 = 1 \\ & z_0, z_j \geq 0, \quad \forall j \end{aligned} \tag{7.15}$$

The problems defined in Eqn. 7.13 and 7.15 are equivalent. On solving the problem in Eqn. 7.15, we substitute, $z_j = x_j * z_0$ to get the results for x_j.

Although the ILP formulations get complicated as the objective function described in Eqn. 7.4 consists of both of the above non-linearities, it is much simpler than the ILP-formulation of the Eqn. 7.3. We observe that the cycle power fluctuation (DP_c) corresponds to $|y_i|$ in Eqn 7.8. DP_c is a measure of the absolute deviation of cycle power from average power and DP is a measure of mean deviation of the cycle power.

7.3 ILP Formulations to Minimize Cycle Power Function

In this section, we discuss the ILP models for minimization of the "modified cycle power function" (CPF^*). We describe the ILP models for two different scenario of ASIC design. The first one targets design with multiple supply voltages and dynamic frequency clocking (MVDFC). The other one targets multiple supply voltages and multicycling (MVMC) based designs. The ILP models formulated ensure that the dependency constraints and the resource constraints are satisfied. In order to formulate an ILP based model for Eqn. 7.6 and the scheduling schemes for the DFG, we use the following notations (Table 7.3).

Table 7.1. List of Variables used in ILP Formulations

$M_{k,v}$	: maximum number of functional units of type k operating at voltage level v ($FU_{k,v}$)
S_i	: as soon as possible (ASAP) time stamp for the operation o_i
E_i	: as late as possible (ALAP) time stamp for the operation o_i
$P(C_{swi}, v, f)$	: power consumption of functional unit FU_i at voltage level v and operating frequency f used by o_i for its execution
$x_{i,c,v,f}$	: decision variable which takes the value of 1 if operation o_i is scheduled in control step c using the functional unit $F_{k,v}$ and c has frequency f_c
$y_{i,v,l,m}$	: decision variable which takes the value of 1 if operation o_i is using the functional unit $F_{k,v}$ and scheduled in control steps $l \rightarrow m$
$L_{i,v}$	: latency for operation o_i using functional unit operating at voltage v (in terms of number of clock cycles)

7.3.1 Multiple Voltages and Dynamic Frequency Clocking (MVDFC)

In this subsection, we describe the ILP formulation for minimization of CPF^* using multiple supply voltages and dynamic frequency clocking. In dynamic frequency clocking [63, 59, 62], the clock frequency is varied on-the-fly based on the functional units active in that cycle. In this clocking scheme, all the units are clocked by a single clock line which switches at run-time. The frequency reduction creates an opportunity to operate the different functional units at different voltages, which in turn, helps in further reduction of power.

Objective Function : The objective is to minimize the modified cycle power function described in Eqn. 7.4 of the whole DFG over all control steps.

$$\text{Minimize :} \quad CPF^* \tag{7.16}$$

Using Eqn. 7.4, this can be restated as :

$$\text{Minimize :} \quad \frac{\frac{1}{N}\sum_{c=1}^{N} P_c + \frac{1}{N}\sum_{c=1}^{N} |P - P_c|}{P_{peak}} \tag{7.17}$$

This objective function has the two types of non-linearities mentioned in the previous section. We first remove the non-linearity introduced because of the fraction by putting the denominator as a constraint. Then, the problem in Eqn. 7.17 transformed to the one given below.

$$\begin{aligned} &\text{Minimize :} && \tfrac{1}{N}\sum_{c=1}^{N} P_c + \tfrac{1}{N}\sum_{c=1}^{N} |P - P_c| \\ &\text{Subject to :} && \text{Peak power constraints} \end{aligned} \tag{7.18}$$

However, this transformed problem still has the non-linearity in it because of the absolute function. This can be converted to an equivalent problem using the transformation suggested in the previous section.

$$\begin{aligned} &\text{Minimize :} && \tfrac{1}{N}\sum_{c=1}^{N} P_c + \tfrac{1}{N}\sum_{c=1}^{N} (P + P_c) \\ &\text{Subject to :} && \text{Modified peak power constraints} \end{aligned} \tag{7.19}$$

The "peak power constraint" in Eqn. 7.18 and the "modified peak power constraint" in Eqn. 7.19 will be discussed in later part of the subsection. The problem expressed in Eqn. 7.19 is simplified to :

$$\begin{aligned} &\text{Minimize :} && \left(\tfrac{3}{N}\right)\sum_{c=1}^{N} P_c \\ &\text{Subject to :} && \text{Modified peak power constraints} \end{aligned} \tag{7.20}$$

Using the decision variables, the objective function is formulated as,

$$\begin{aligned} &\text{Minimize :} && \sum_c \sum_{i \in F_{k,v}} \sum_v \sum_f x_{i,c,v,f} * \left(\tfrac{3}{N}\right) * P(C_{swi}, v, f) \\ &\text{Subject to :} && \text{Modified peak power constraints} \end{aligned} \tag{7.21}$$

$$\begin{aligned} &\text{Minimize :} && \sum_c \sum_{i \in F_{k,v}} \sum_v \sum_f x_{i,c,v,f} * P^*(C_{swi}, v, f) \\ &\text{Subject to :} && \text{Modified peak power constraints} \end{aligned} \tag{7.22}$$

where, $P^*(C_{swi}, v, f)$ is given by $P(C_{swi}, v, f) * \left(\tfrac{3}{N}\right)$.

Uniqueness Constraints : These constraints ensure that every operation o_i is scheduled to one unique control step within the mobility range (S_i, E_i) with a particular supply voltage and operat-

ing frequency. We represent them as, $\forall i$, $1 \leq i \leq O$,

$$\sum_c \sum_v \sum_f x_{i,c,v,f} \quad = \quad 1 \tag{7.23}$$

Precedence Constraints : These constraints guarantee that for an operation o_i, all its predecessors are scheduled in an earlier control step and its successors are scheduled in a later control step. These are modeled as, $\forall i, j, o_i \in Pred_{o_j}$,

$$\sum_v \sum_f \sum_{d=S_i}^{E_i} d * x_{i,d,v,f} - \sum_v \sum_f \sum_{e=S_j}^{E_j} e * x_{j,e,v,f} \quad \leq \quad -1 \tag{7.24}$$

Resource Constraints : These constraints make sure that no control step contains more than $F_{k,v}$ operations of type k operating at voltage v. These can be enforced as, $\forall c$, $1 \leq c \leq N$ and $\forall v$,

$$\sum_{i \in F_{k,v}} \sum_f x_{i,c,v,f} \quad \leq \quad M_{k,v} \tag{7.25}$$

Frequency Constraints : This set ensures that if a functional unit is operating at a higher voltage level then it can be scheduled in a lower frequency control step, whereas, a functional unit is operating at lower voltage level then it can not be scheduled in a higher frequency control step. We write these constraints as, $\forall i$, $1 \leq i \leq O$, $\forall c$, $1 \leq c \leq N$, if $f < v$, then $x_{i,c,v,f} = 0$.

Peak Power Constraints : As discussed before, with reference to the Eqn. 7.17 and 7.18, these constraints are introduced to eliminate the fractional non-linearity of the objective function. These constraints ensure that the maximum power consumption of the DFG does not exceed P_{peak} for any control step. We enforce these constraints as follows, $\forall c$, $1 \leq c \leq N$,

$$\sum_{i \in F_{k,v}} \sum_v \sum_f x_{i,c,v,f} * P(C_{swi}, v, f) \quad \leq \quad P_{peak} \tag{7.26}$$

Modified Peak Power Constraints : To eliminate the non-linearity introduced due to the absolute function, we modify the above constraints, as outlined in Eqn. 7.18 and 7.19. The peak power constraints in Eqn. 7.26 is modified as, $\forall c, 1 \leq c \leq N$,

$$\begin{aligned} &\tfrac{1}{N} \sum_c \sum_{i \in F_{k,v}} \sum_v \sum_f x_{i,c,v,f} * P(C_{swi}, v, f) \\ &- \sum_{i \in F_{k,v}} \sum_v \sum_f x_{i,c,v,f} * P(C_{swi}, v, f) \leq P^*_{peak} \end{aligned} \tag{7.27}$$

The P^*_{peak} is a modified peak constraint which is added to the objective function and minimized alongwith it.

7.3.2 Multiple Voltages and Multicycling (MVMC)

In this subsection, we describe the ILP formulations based on the modified cycle power function (CPF^*) using multiple supply voltages and multicycling. In this scheme, the functional units are operated at multiple supply voltages. The functional units operating at lower voltages may need to be active in more than one consecutive control steps to complete execution.

Objective Function : The objective is to minimize the CPF^* for the entire DFG. Using Eqn. 7.4, this can be represented as :

$$\begin{aligned} \text{Minimize :} \quad & CPF^* \\ = \quad & \frac{\frac{1}{N}\sum_{c=1}^{N} P_c + \frac{1}{N}\sum_{c=1}^{N} |P - P_c|}{P_{peak}} \end{aligned} \tag{7.28}$$

As discussed in the previous subsection, this objective function has two types of non-linearities, which are because of the absolute function and the fractional form. The fractional non-linearity is removed by introducing the denominators as a constraint. The corresponding constraints are known as "peak power constraints". We remove the absolute function non-linearity by modifying the peak power constraints which give rises to "modified peak power constraints". Thus, the problem in

Eqn. 7.28 is transformed to the following.

$$\begin{aligned} &\text{Minimize :} && \tfrac{1}{N}\sum_{c=1}^{N} P_c + \tfrac{1}{N}\sum_{c=1}^{N}(P + P_c) \\ &\text{Subject to :} && \text{Modified peak power constraints} \end{aligned} \tag{7.29}$$

The "peak power constraint" and the "modified peak power constraint" are discussed in the later part of the subsection. The problem in Eqn. 7.29 is simplified to :

$$\begin{aligned} &\text{Minimize :} && \left(\tfrac{3}{N}\right)\sum_{c=1}^{N} P_c \\ &\text{Subject to :} && \text{Modified peak power constraints} \end{aligned} \tag{7.30}$$

Using the decision variables, the above LP objective function is formulated as,

$$\begin{aligned} &\text{Minimize :} && \sum_l \sum_{i \in F_{k,v}} \sum_v y_{i,v,l,(l+L_{i,v}-1)} * \left(\tfrac{3}{N}\right) P(C_{swi}, v, f_{clk}) \\ &\text{Subject to :} && \text{Modified peak power constraints} \end{aligned} \tag{7.31}$$

where, f_{clk} is the operating frequency level of the datapath circuit in multicycling mode.

$$\begin{aligned} &\text{Minimize :} && \sum_l \sum_{i \in F_{k,v}} \sum_v y_{i,v,l,(l+L_{i,v}-1)} * P^*(C_{swi}, v, f_{clk}) \\ &\text{Subject to :} && \text{Modified peak power constraints} \end{aligned} \tag{7.32}$$

where, $P^*(C_{swi}, v, f_{clk}) = \left(\tfrac{3}{N}\right) * P(C_{swi}, v, f_{clk})$, are modified power values.

Uniqueness Constraints : These constraints ensure that every operation o_i is scheduled in appropriate control steps within the mobility range (S_i, E_i) with a particular supply voltage. Depending on the supply voltage it may be operated at more than one clock cycle. We represent them as, $\forall i$, $1 \leq i \leq O$,

$$\sum_v \sum_{l=S_i}^{S_i+E_i+1-L_{i,v}} y_{i,v,l,(l+L_{i,v}-1)} = 1 \tag{7.33}$$

When the operators are computed at the highest voltage, they are scheduled in one unique control step, whereas, when they are to be operated at lower voltages they need more than one clock cycle for completion. Thus, for lower voltage, the mobility is restricted.

Precedence Constraints : These constraints guarantee that for an operation o_i, all its predecessors are scheduled in earlier control step and its successors are scheduled in later control step. These constraints should also take care of the multicycling operations. These are modelled as, $\forall i, j, o_i \in Pred_{o_j}$,

$$\sum_v \sum_{l=S_i}^{E_i} (l + L_{i,v} - 1) * y_{i,v,l,(l+L_{i,v}-1)} - \sum_v \sum_{l=S_j}^{E_j} l * y_{j,v,l,(l+L_{j,v}-1)} \quad \leq \quad -1 \tag{7.34}$$

Resource Constraints : These constraints make sure that no control step contains more than $F_{k,v}$ operations of type k operating at voltage v. These can be enforced as, $\forall v$ and $\forall l$, $1 \leq l \leq N$,

$$\sum_{i \in F_{k,v}} \sum_l y_{i,v,l,(l+L_{i,v}-1)} \quad \leq \quad M_{k,v} \tag{7.35}$$

Peak Power Constraints : As discussed earlier with reference to Eqn. 7.28 and 7.29, these constraints are enforced to eliminate the fractional non-linearity of the objective function. We enforce these constraints as follows, $\forall l$, $1 \leq l \leq N$,

$$\sum_{i \in F_{k,v}} \sum_v y_{i,v,l,(l+L_{i,v}-1)} * P(C_{swi}, v, f_{clk}) \quad \leq \quad P_{peak} \tag{7.36}$$

Modified Peak Power Constraints : These constraints are introduced to eliminate the absolute function non-linearity of the objective function. These constraints can be enforced as, $\forall l$, $1 \leq l \leq N$,

$$\begin{aligned} &\tfrac{1}{N} \sum_l \sum_{i \in F_{k,v}} \sum_v y_{i,v,l,(l+L_{i,v}-1)} * P(C_{swi}, v, f_{clk}) \\ &- \sum_{i \in F_{k,v}} \sum_v y_{i,v,l,(l+L_{i,v}-1)} * P(C_{swi}, v, f_{clk}) \leq P^*_{peak} \end{aligned} \tag{7.37}$$

where, P^*_{peak} is the modified peak power constraint which is also minimized as a part of the objective function.

7.4 ILP-Based Scheduling Algorithm

In this section, we discuss the solutions for the ILP formulations obtained in the previous section and develop scheduling algorithms for both MVDFC and MVMC schemes. The target architecture model assumed for the scheduling schemes is from [65]. Each functional unit has a register and a multiplexor associated with it. The register and the multiplexor operate at the same voltage level as that of the functional unit. Level converters are used when a low-voltage functional unit drives a high-voltage functional unit [65, 95]. A controller decides which of the functional units are active in each control step and those that are not active are disabled using the multiplexors. For MVDFC scheme, the controller has a storage unit to store cycle frequency index (cfi_c) values obtained from scheduling. This serves as the clock dividing factor for the dynamic clocking unit. The cycle frequency f_c is generated dynamically and a functional unit operating at one of the supply voltages is activated.

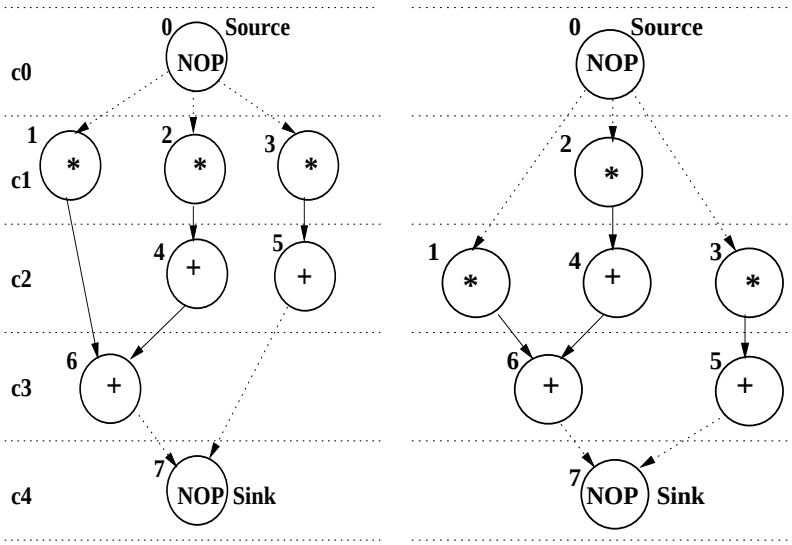
The inputs to the algorithm are an unscheduled data flow graph (UDFG), the resource constraints, the number of allowable voltage levels (L_V), the number of allowable frequencies (L_f), the delay of each resource (d_{FU}), the multiplexor (d_{Mux}), the register (d_{Reg}) at different voltage levels. The delays of level converters (d_{Conv}) is represented in the form of a matrix that shows the delay in converting one at voltage level V_i to another voltage level V_j (where, both $V_i, V_j \in V_{L_V}$). The resource constraint includes the number of ALUs and multipliers at different voltage levels V_i (where, $V_i \in V_{L_V}$). The scheduling algorithm determines the f_{base}, cfi_c time stamp for each operation, and voltage level such that the function CPF^* (Eqn. 7.6) is minimum.

The ILP based scheduler which minimizes the modified cycle power function CPF^* of the DFG is outlined in Fig. 7.1. In step 1, the scheduler constructs a look-up table for effective switching capacitance for known values of the average switching activity pair as described in Eqn. 7.5. In step 2, the scheduler determines the switching activities at the inputs of each node by using behavioral simulation of DFG. For this purpose, a different set of application specific input vectors (having different correlations) are given at the primary inputs of the DFG and average switching activity at each inputs of other nodes are calculated [167, 168, 169]. It should be noted that if the look-up table (in step 1) does not have the switching capacitance for an average switching activity

Input	: UDFG, resource constraints, L_V, L_f, all $V_i \in V_{L_V}$, d_{FU}, d_{Mux}, d_{Reg}, d_{Conv}
Output	: scheduled DFG, f_{base}, N, cfi_c, power, energy and delay estimates
Step 1	: Construct a look up table for effective switching capacitance.
Step 2	: Calculate the switching activities at each node through behavioral simulation.
Step 3	: Find ASAP schedule for the UDFG.
Step 4	: Find ALAP schedule for the UDFG.
Step 5	: Determine the mobility graph of each node.
Step 6	: Modify the mobility graph for MVMC.
Step 7	: Model the ILP formulations of the DFG using AMPL.
Step 8	: Solve the ILP formulations using LP-Solve.
Step 9	: Find the scheduled DFG.
Step 10	: Determine the cycle frequencies (f_c), f_{base} and cfi_c for MVDFC scheme.
Step 11	: Estimate the power and energy consumptions of the scheduled DFG.

Figure 7.1. Scheduling for CPF^* Minimization

value (in step 2), then the scheduler uses interpolation techniques to find the same. The third step is to determine the as soon as possible (ASAP) time stamp of each operation. The fourth step is the determination of the as late as possible (ALAP) time stamp of each vertex for the DFG. The ASAP time stamp is the start time and the ALAP time stamp is the finish time of each operation. These two time stamps provide the mobility of an operation and the operation must be scheduled within this mobility range. This mobility graph needs to be modified for the MVMC scheme. The ILP formulations constructed based on the models described in section 7.3. The scheduler uses the modeling language AMPL to model the ILP formulations [166]. At this step, we calculate the power consumption of the functional units as follows. The operational delay of a functional unit is assumed as ($d_{FU} + d_{Mux} + d_{Reg} + d_{Conv}$). For the MVMC scheme the operating frequency is the frequency corresponding to the operational delay at the highest operating voltage of multiplier unit. On the other hand, for MVDFC scheme, the operating frequency of a functional unit is calculated based on these operational delay using the formulas given in [48]. It is assumed to be the inverse of operational delay of a functional unit at corresponding supply voltage. We get the switching capacitance from step 1 and step 2, and the power values are calculated whenever



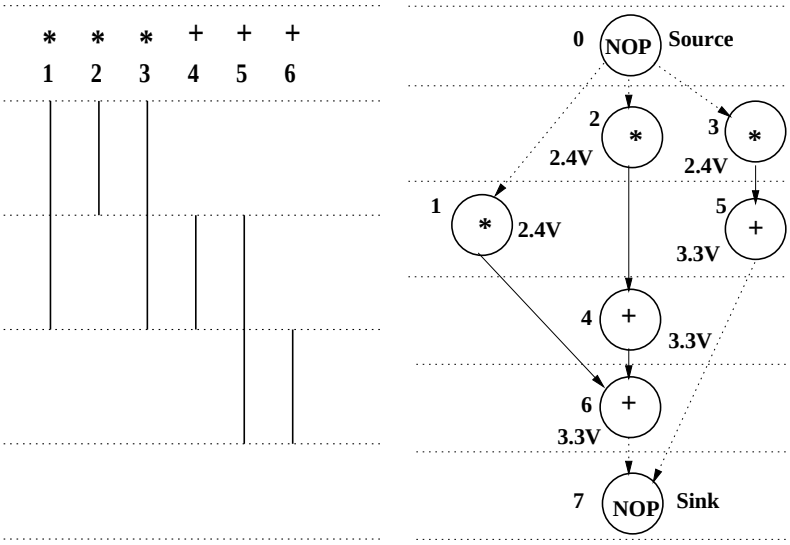
(a) ASAP Schedule for EXP DFG (b) ALAP Schedule for EXP DFG

Figure 7.2. ASAP and ALAP Schedule for Example DFG (used to find Mobility Graph)

necessary for different operating voltages and frequencies. The scheduled DFG is obtained after the ILP formulation is solved using LP-Solve. Then, the scheduler determines the f_{base}, cfi_c and cycle frequency (f_c) using the methods proposed in [48] based on the delay of each cycle. Finally, the power consumption, energy consumption and the energy delay product of the scheduled DFG are calculated.

7.4.1 CPF-MVDFC Scheduling Scheme

We illustrate the solution for the ILP formulation in the MVDFC case, with the help of the DFG shown in Fig. 7.2. The ASAP schedule is shown in Fig. 7.2(a) and the ALAP schedule is shown in Fig. 7.2(b). From the ASAP and ALAP schedules, we obtained the mobility graph which is Fig. 7.3(a). We get the ILP formulations using this mobility graph. We solved the formulation using LP-solve and based on the results, we obtained the scheduled DFG shown in Fig. 7.3(b) for the resource constraint (RC5), two multipliers at $2.4V$ and one ALU operating at $3.3V$. Similarly, other schedules can be obtained for different resource constraints.

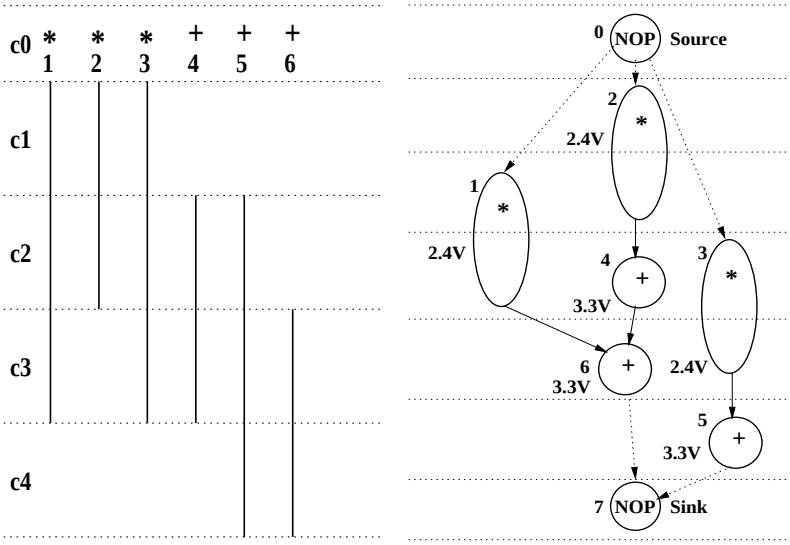


(a) Mobility Graph (b) Final Schedule

Figure 7.3. Mobility Graph and Final Schedule for Example DFG for RC5 using MVDFC

7.4.2 CPF-MVMC Scheduling Scheme

We illustrate the solution for the ILP formulations of the MVMC case, using the DFG shown in Fig. 7.2. The ASAP schedule is shown in Fig. 7.2(a) and the ALAP schedule is shown in Fig. 7.2(b). From the ASAP schedule (Fig. 7.2(a)) and the ALAP schedule (Fig. 7.2(b)), we obtained the mobility graph shown in Fig. 7.4(a). This mobility graph is different from that shown in Fig. 7.3(a). In the MVMC case, the mobility graph considers the multicycle operations. In this illustration, we assume that we have two operating voltage levels, and when the multipliers are operated at the lower voltage, they take two clock cycles. It should be noted that the mobility graph will depend on the number of operating voltages and the assumed operating frequency. We solved the ILP formulation using LP-solve and based on the results we obtained the scheduled DFG shown is Fig. 7.4(b) for the resource constraint (RC5), two multipliers at $2.4V$ and one ALUs operating at $3.3V$.



(a) Mobility Graph (b) Final Schedule

Figure 7.4. Mobility Graph and Final Schedule for Example DFG for RC5 using MVMC

7.5 Experimental Results

The ILP based CPF-MVDFC and CPF-MVMC schedulers were tested with five benchmark circuits :

- Example circuit (EXP) (8 nodes, 3*, 3+, 9 edges)
- FIR filter (11 nodes, 5*, 4+, 19 edges)
- HAL differential equation solver (13 nodes, 6+, 2+, 2-, 1 <, 16 edges)
- IIR filter (11 nodes, 5*, 4+, 19 edges)
- Auto-Regressive filter (ARF) (15 nodes, 5*, 8+, 19 edges).

The following notations are used to express results (Table 7.5).

We use the look-up table method presented in Section 7.1 for average switching capacitance calculation. The look-up table construction consists of two phases, such as input pattern generation and cell characterization. We generate the primary input signals of different correlations using

Table 7.2. List of Variables used to Express the Results

P_{pS}	: peak power consumption (in mW) for single supply voltage and single frequency scheme
P_{pD}	: peak power consumption (in mW) for multiple supply voltages and dynamic frequency operation
P_{pM}	: the peak power consumption (in mW) for multiple supply voltages and multicycle operation
P_{mS}	: minimum power consumption (in mW) for any cycle assuming single frequency and single supply voltage
P_{mD}	: minimum power consumption (in mW) for any cycle for dynamic clocking and multiple supply voltage
T_S	: execution time for single frequency
T_D	: execution time for dynamic frequency
T_M	: execution time for multicycling operation
E_S	: total energy consumption (in nano-Joule or nJ) for single supply voltage and single frequency scheme
E_D	: total energy consumption (in nJ) for multiple supply voltages and dynamic frequency operation
E_M	: total energy consumption (in nJ) for multiple supply voltages and multicycle operation
P_S	: average power consumption (in mW) for single supply voltage and single frequency scheme which is calculated as the mean of the cycle power consumptions
P_D	: average power consumption (in mW) for multiple supply voltages and dynamic frequency operation, estimated as the mean of the cycle power
P_M	: average power consumption (in mW) for multiple supply voltages and multicycle operation, calculated as the mean of the cycle power consumptions
EDP_S	: energy delay product (in 10^{-15}Joule-sec or fJs) for single supply voltage and single frequency operation ($= E_S * T_S$)
EDP_D	: energy delay product (in fJs) for multiple supply voltage and dynamic frequency clocking operation ($= E_D * T_D$)
EDP_M	: energy delay product (in fJs) for multiple supply voltage and multicycle operation ($= E_M * T_M$)
ΔP_p	: percentage peak power reduction, for MVDFC scheme this is defined as, $\frac{(P_{pS}-P_{pD})}{P_{pS}} * 100$ and for MVMC scheme it is calculated as, $\frac{(P_{pS}-P_{pM})}{P_{pS}} * 100$
ΔDP	: percentage differential power reduction, which is calculated as $\frac{(P_{pS}-P_{mS})-(P_{pD}-P_{mD})}{(P_{pS}-P_{mS})} * 100$ for MVDFC scheme and as $\frac{(P_{pS}-P_{mS})-(P_{pM}-P_{mM})}{(P_{pS}-P_{mS})} * 100$ for MVMC scheme
ΔP	: percentage average power reduction, for MVDFC sheme it is $\frac{P_S-P_D}{P_S} * 100$ and for MVMC scheme it is $\frac{P_S-P_M}{P_S} * 100$
ΔE	: percentage reduction in total energy, is calculated as $\frac{E_S-E_D}{E_S} * 100$ for MVDFC scheme and as $\frac{E_S-E_M}{E_S} * 100$ for MVMC scheme
ΔEDP	: percentage EDP reduction, calculated as $\frac{(EDP_S-EDP_D)}{EDP_S} * 100$ for MVDFC scheme and as $\frac{(EDP_S-EDP_M)}{EDP_M} * 100$ for MVMC scheme

the autoregressive moving average (ARMA) model [169]. We perform the characterization of the physical implementations of the library modules available in [55] by applying the input patterns generated above for some values of $({\alpha_i}^1, {\alpha_i}^2)$ pairs. Whenever necessary, we used interpolation to find the average switching capacitance for any other values of $({\alpha_i}^1, {\alpha_i}^2)$ pairs that do not exist in the look-up table. It should be noted that larger the size of look-up table, better is the accuracy. The above generated signals are propagated through different operators in the DFG and the average switching activities are calculated as described in [169].

Both the scheduling algorithms, CPF-MVDFC and CPF-MVMC were tested using five different sets of resource constraints (RC1,RC2,RC3,RC4,RC5) :

(1) multipliers (2 at $2.4V$ and 1 at $3.3V$) and ALUs (1 at $2.4V$ and 1 at $3.3V$),
(2) multipliers (3 at $2.4V$) and ALUs (1 at $2.4V$ and 1 at $3.3V$),
(3) multipliers (2 at $2.4V$) and ALUs (2 at $3.3V$),
(4) multipliers (1 at $2.4V$ and 1 at $3.3V$) and ALUs (1 at $3.3V$), and
(5) multipliers (2 at $2.4V$) and ALUs (1 at $3.3V$).

The reason behind choosing the sets of resource constraints is that it covers a good representive of types of resources at different operating voltages. The number of allowable voltage levels is two $(2.4V, 3.3V)$ and maximum number of allowable frequencies being three. The experimental results for various benchmark circuits are reported in Table 7.3 for CPF-MVDFC scheduling scheme and in Table 7.4 for CPF-MVMC scheduling scheme. The power/energy estimation include the power consumption of the overheads, such as level converters (data taken from [55]). The results are reported for two supply voltages. In case of CPF-MVDFC scheduling the frequencies found out are $(4.5MHz, 9MHz, 18MHz)$. For CPF-MVMC scheduling scheme the operating frequency (f_{clk}) is $9MHz$.

We plotted Fig. 7.5 and 7.6 to get a visual picture of the experimental results. The figures show the average reductions for different benchmarks averaged over all resource constraints. It is obvious from the figure that the reductions are significant. It is also noted that for the reductions for MVDFC scheme is better than the MVMC scheme. The CPF-MVDFC scheme works effectively for all resource constraints and all benchmarks, where as, the CPF-MVMC scheme does not produce good

Table 7.3. Power, Energy and EDP Estimates for Benchmarks using MVDFC

	Power, Energy and Energy-Delay-Product															
	RC	P_{pS}	P_{pD}	ΔP_p	P_{mS}	P_{mD}	ΔDP	P_S	P_D	ΔP	E_S	E_D	ΔE	EDP_S	EDP_D	ΔEDP
		mW	mW	%	mW	mW	%	mW	mW	%	nJ	nJ	%	fJs	fJs	%
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
(1) EXP	1	17.28	4.56	73.61	0.46	0.35	74.97	8.87	2.42	72.72	2.96	1.57	46.8	0.99	0.87	11.34
	2	17.28	4.56	73.61	0.46	0.35	74.97	8.87	2.42	72.72	2.96	1.57	46.8	0.99	0.87	11.34
	3	17.28	4.56	73.61	0.46	0.9	78.24	8.87	2.61	70.57	2.96	1.6	46.0	0.99	0.8	18.98
	4	8.87	2.39	73.05	0.45	0.23	77.55	6.67	1.87	71.96	2.96	1.58	46.4	1.31	1.14	12.89
	5	17.28	4.56	73.61	0.23	0.45	75.89	6.65	1.96	70.53	2.96	1.6	45.9	1.31	0.87	32.49
	Average values			73.50			76.32			71.70			46.38			17.41
(2) FIR	1	17.51	4.62	73.62	0.23	0.12	73.96	8.82	2.35	73.36	4.9	2.6	47.2	2.7	2.3	15.52
	2	25.92	6.84	73.61	0.23	0.12	73.84	8.82	2.36	73.24	4.9	2.6	47.2	2.7	2.0	26.09
	3	17.51	4.67	73.33	0.23	0.45	75.58	8.82	2.5	71.66	4.9	2.6	46.22	2.7	2.0	24.71
	4	17.28	6.6	61.81	0.23	0.45	63.93	8.82	2.84	67.8	4.9	3.1	36.98	2.7	2.9	No
	5	17.51	4.67	73.33	0.23	0.45	75.58	8.82	2.5	71.66	4.9	2.6	46.22	2.7	2.0	24.71
	Average values			71.14			72.60			71.54			44.76			16.21
(3) HAL	1	17.51	4.62	73.62	0.46	0.35	74.96	13.25	3.55	73.21	5.9	3.12	47.0	2.62	2.43	7.25
	2	26.15	6.9	73.61	0.46	0.35	74.5	13.25	3.55	73.21	5.9	3.12	47.0	2.62	2.43	7.25
	3	17.74	4.78	73.05	0.46	0.9	76.97	13.25	3.73	71.85	5.9	3.17	46.2	2.62	2.23	12.55
	4	17.51	6.71	61.68	0.23	0.45	63.77	10.6	3.73	64.8	5.9	4.07	30.8	3.27	3.85	No
	5	17.51	4.67	73.33	0.23	0.45	75.6	10.6	2.98	71.9	5.9	3.17	46.2	3.27	2.46	24.66
	Average values			71.06			73.16			71.0			43.44			10.34
(4) IIR	1	25.92	8.88	65.74	0.23	0.12	65.9	11.03	3.5	68.36	4.9	3.05	37.7	2.18	2.04	6.57
	2	25.92	6.84	73.61	0.23	0.12	73.84	11.03	2.98	72.98	4.9	2.6	47.96	2.18	1.73	20.44
	3	17.51	4.67	73.34	0.23	0.45	75.58	8.82	2.57	70.86	4.9	2.64	46.22	2.72	2.05	24.71
	4	17.51	6.71	61.68	0.23	0.45	63.77	8.82	3.32	62.86	4.9	3.54	27.73	2.72	2.75	No
	5	17.51	4.67	73.33	0.23	0.45	75.58	8.82	2.5	71.66	4.9	2.64	46.22	2.72	2.05	24.71
	Average values			69.54			71.65			69.34			41.17			15.24
(5) ARF	1	8.87	2.34	73.62	0.23	0.12	74.1	4.5	1.22	72.9	5.0	2.64	47.2	5.56	4.4	20.83
	2	8.87	2.34	73.62	0.23	0.12	74.1	4.5	1.22	72.9	5.0	2.64	47.2	5.56	4.4	20.83
	3	8.87	2.39	73.05	0.23	0.45	77.6	4.5	1.4	68.9	5.0	2.74	45.3	5.56	3.8	31.63
	4	8.87	2.39	73.05	0.23	0.45	77.6	4.5	1.4	68.9	5.0	2.74	45.3	5.56	3.8	31.63
	5	8.87	2.39	73.05	0.23	0.45	77.6	4.5	1.4	68.9	5.0	2.74	45.3	5.56	3.8	31.63
	Average values			73.28			76.20			70.5			46.06			27.31
Overall average				71.70			74.0			70.82			44.36			17.31

Table 7.4. Power, energy and EDP Estimates for Benchmarks using MVMC

		Power, Energy and Energy-Delay-Product														
	R	P_{pS}	P_{pM}	ΔP_p	P_{mS}	P_{mM}	ΔDP	P_S	P_M	ΔP	E_S	E_M	ΔE	EDP_S	EDP_M	ΔEDP
	C	mW	mW	%	mW	mW	%	mW	mW	%	nJ	nJ	%	fJs	fJs	%
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
(1) EXP	1	17.28	13.2	23.61	0.46	0.35	23.6	8.87	6.84	22.9	3.0	2.03	31.47	0.99	0.9	8.63
	2	17.28	13.7	20.83	0.46	0.35	20.8	8.87	6.96	21.53	3.0	1.57	46.8	0.99	0.7	29.07
	3	17.28	9.12	47.22	0.46	0.46	48.51	8.87	5.61	36.75	3.0	1.57	46.0	0.99	0.89	9.98
	4	8.87	13.43	NA	0.23	0.23	NA	6.67	6.77	NA	3.0	2.5	16.46	1.31	1.11	15.33
	5	17.28	9.35	45.9	0.23	0.23	46.51	6.65	5.61	15.64	3.0	1.6	46.0	1.31	0.89	32.5
	Average values			27.51			27.88			19.36			37.35			19.10
(2) FIR	1	17.51	17.76	NA	0.23	0.23	NA	8.87	7.67	13.04	4.9	3.09	37.0	2.72	2.06	24.38
	2	25.92	13.68	47.22	0.23	0.12	47.21	8.82	7.66	13.15	4.9	2.59	47.2	2.72	1.72	36.64
	3	17.51	9.35	46.6	0.23	0.23	47.22	8.82	7.75	12.13	4.9	2.64	46.22	2.72	2.05	24.71
	4	17.28	13.43	22.28	0.23	0.23	22.58	8.82	7.51	14.85	4.9	4.0	18.5	2.72	2.66	2.19
	5	17.51	9.35	46.6	0.23	0.23	47.22	8.82	6.65	24.6	4.9	2.64	46.22	2.72	2.05	24.71
	Average values			32.54			32.85			15.55			39.03			22.53
(3) HAL	1	17.51	17.76	NA	0.46	0.35	NA	13.25	9.08	31.47	5.9	4.0	31.6	2.62	2.68	NA
	2	26.15	13.8	47.23	0.46	0.35	47.64	13.25	9.24	30.26	5.9	3.2	47.0	2.62	2.08	20.61
	3	17.74	9.58	46.0	0.46	0.46	47.22	13.25	7.98	39.77	5.9	3.2	46.19	2.62	2.46	6.11
	4	17.51	13.43	23.3	0.23	0.23	23.61	10.6	9.0	15.2	5.9	5.0	15.4	3.27	3.32	NA
	5	17.51	9.35	46.6	0.23	0.23	47.22	10.6	6.41	39.53	5.9	3.17	46.18	3.27	2.82	13.76
	Average values			32.63			33.14			33.14			37.27			8.10
(4) IIR	1	25.92	17.76	31.48	0.23	0.12	31.34	11.03	8.95	18.85	4.9	4.0	19.22	2.18	2.2	NA
	2	25.92	13.8	46.76	0.23	0.12	46.75	11.03	7.68	30.37	4.9	2.6	47.2	2.18	1.72	20.81
	3	17.51	9.12	47.92	0.23	0.23	48.55	8.82	5.82	34.01	4.9	2.6	46.22	2.72	2.34	13.96
	4	17.51	13.43	23.3	0.23	0.23	23.61	8.82	7.51	14.85	4.9	3.54	27.73	2.72	2.36	13.28
	5	17.51	9.12	47.92	0.23	0.23	48.55	8.82	5.82	34.01	4.9	2.64	46.22	2.72	2.34	16.23
	Average values			39.48			39.76			26.42			37.32			12.76
(5) ARF	1	8.87	9.24	NA	0.23	0.12	NA	4.5	3.58	20.44	5.0	2.64	47.22	5.56	3.81	31.4
	2	8.87	9.24	NA	0.23	0.12	NA	4.5	3.58	20.44	5.0	2.64	47.22	5.56	3.81	31.4
	3	8.87	9.35	NA	0.23	0.23	NA	4.5	3.65	18.9	5.0	2.74	45.3	5.56	3.95	28.9
	4	8.87	13.43	NA	0.23	0.23	NA	4.5	3.56	20.9	5.0	3.19	36.24	5.56	4.60	17.11
	5	8.87	9.35	NA	0.23	0.23	NA	4.5	3.65	18.9	5.0	2.74	45.3	5.56	3.95	28.9
	Average values			0			0			19.92			44.26			27.54
Overall average				26.44			26.73			22.51			39.05			17.99

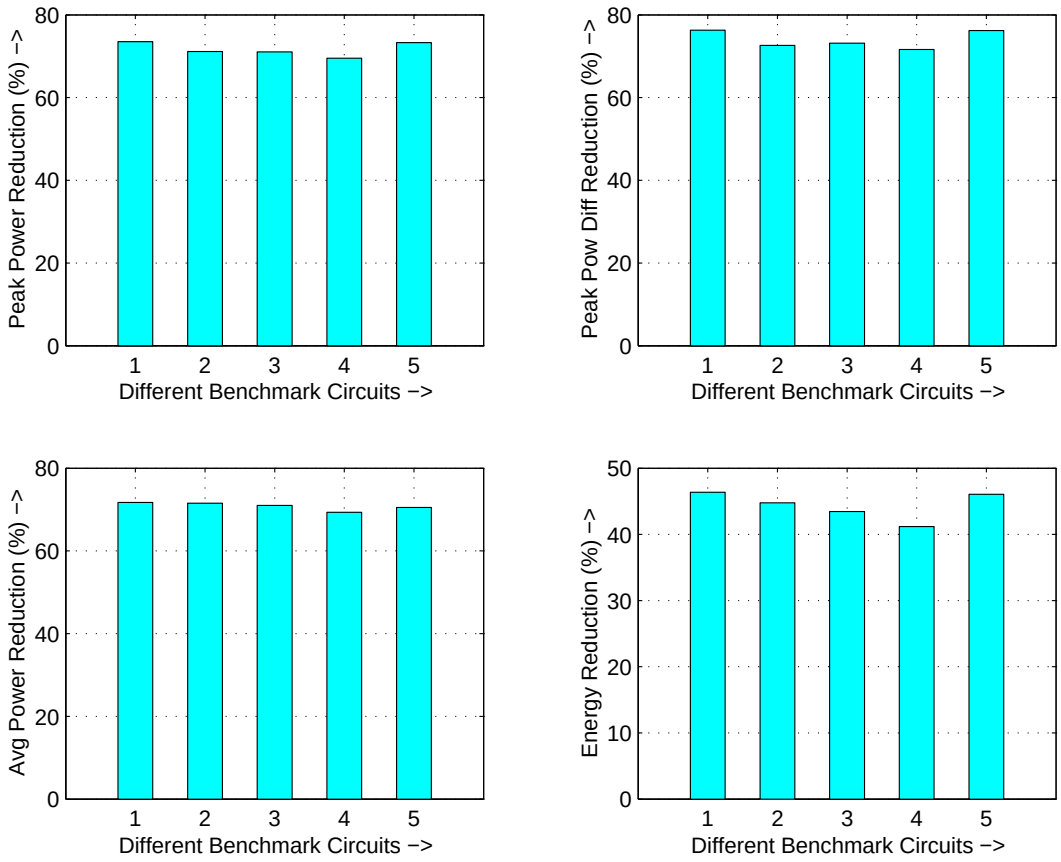


Figure 7.5. Average Reductions in Power or Energy for Benchmarks using CPF-MVDFC

results for ARF benchmark. We did not find any work in the literature that deals with simultaneous reduction of energy and transient power, so we could not provide comparison with any other works.

In order to study the power consumption per cycle, we plotted the power profile for different benchmarks over all the control steps (clock steps). Fig. 7.7, 7.8, 7.9, 7.10 and 7.11 show power profile for benchmarks for resource constraints RC1, RC2, RC3, RC4 and RC5 respectively. The curves labeled as ”SF” correspond to the profile when the schedule is operated at a single frequency (which is the maximum frequency of slower operator, multiplier) and single voltage. The profiles labeled as ”DFC” correspond to the case when dynamic clocking and multiple voltage scheme is used. Similarly, the profiles labeled as ”MC” is for the MVMC scheme. The effectiveness of the proposed scheduling schemes is obvious from the figures.

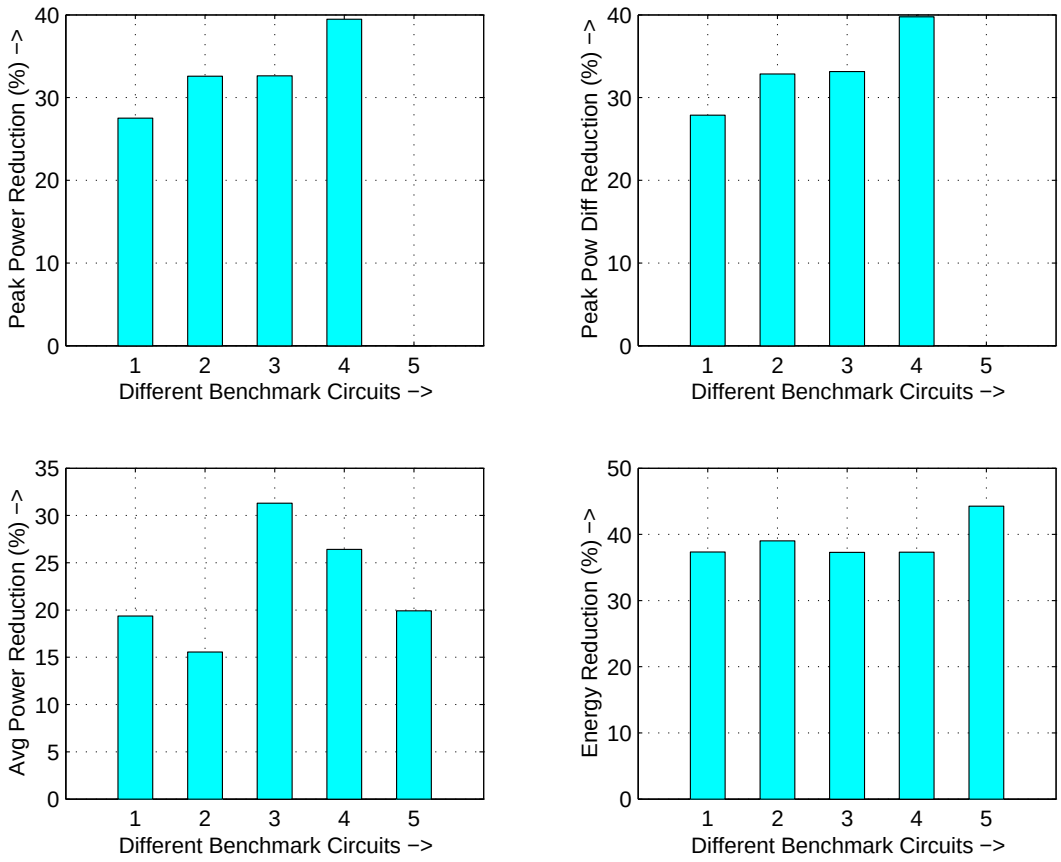


Figure 7.6. Average Reductions for Benchmarks using CPF-MVMC

7.6 Conclusions

In low power deigns for portable applications, the simultaneous minimization of total energy and transient power is essential. The modifed-CPF parameter defined and used in this work essentially facilitates such simultaneous optimization using ILP formulations. The optimization is performed using MVDFC scheme and MVMC scheme. The datapath scheduling algorithm described in this chapter is particularly useful for synthesizing data intensive application specific integrated circuits. The algorithm attempts to optimize energy and power while maintaining performance. The scheduling algorithm assumes number of different types of resources at each voltage levels (both CPF-MVDFC and CPF-MVMC) and the number of allowable frequencies (CPF-MVMC scheme) as resource constraints. The energy delay product for both the CPF-MVDFC and CPF-MVMC scheduling scenario was estimated to keep track of the effect of scheduling algorithms on

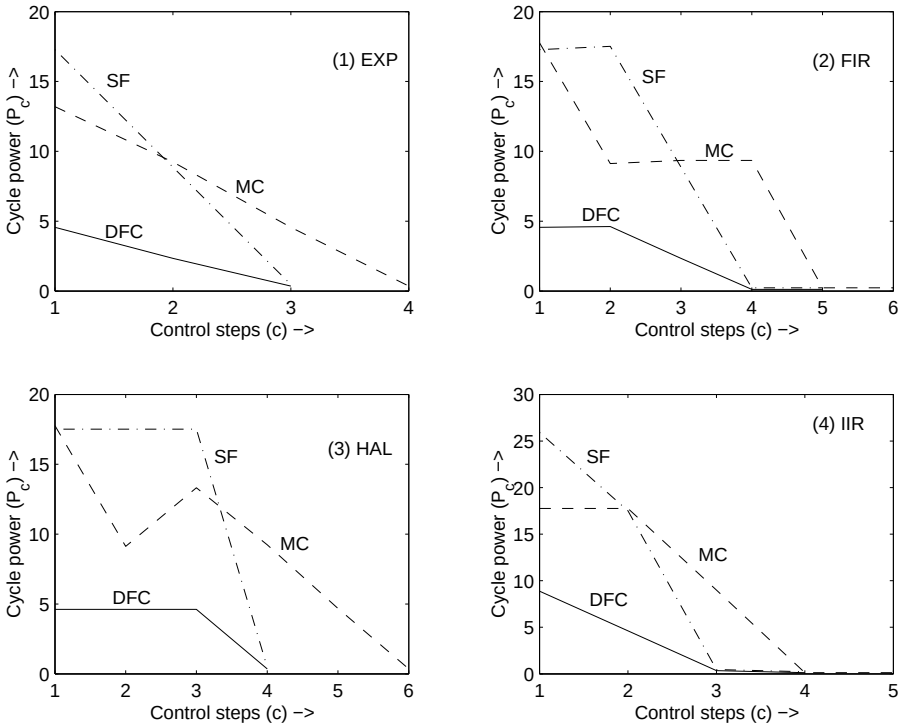


Figure 7.7. Power Profile for Benchmark for Resource Constraint RC1

circuit performance. The CPF-MVDFC scheduling resulted in reduction of EDP for all benchmarks and all resource constraints, which shows its effectiveness. On the other hand, the CPF-MVMC scheme resulted in improvement in EDP in almost all cases, except for a few cases, where there was no improvement. *The results clearly indicate that multiple supply voltage and dynamic frequency clocking scheme yields better power and energy minimization than multiple supply voltage and multicycling scheme*. The effectiveness of the scheduling schemes in the context of pipelined datapath and control intensive applications, needs to be investigated.

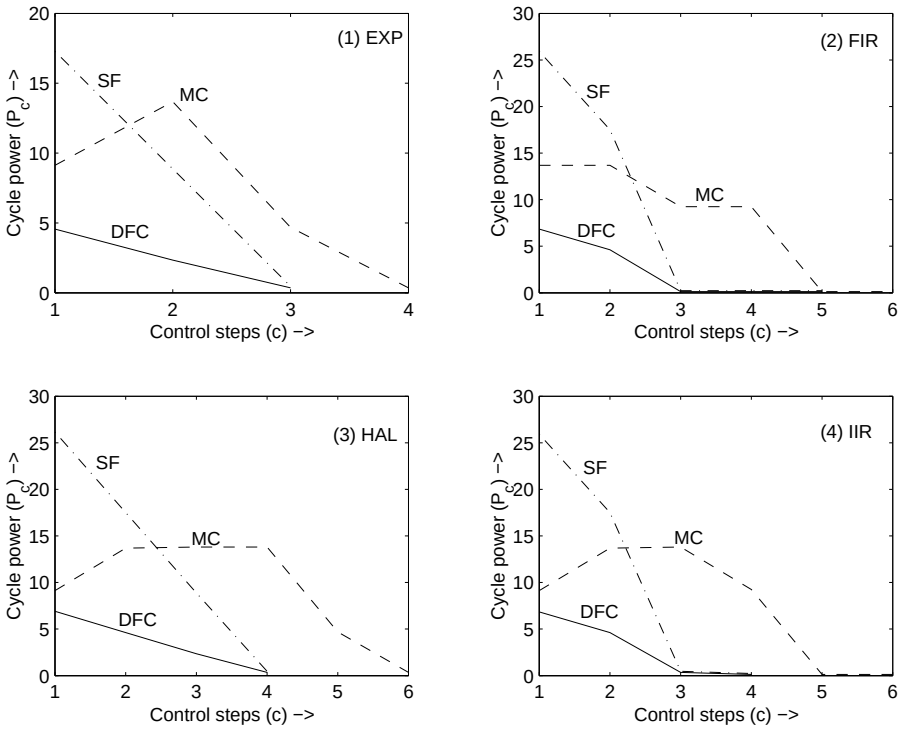


Figure 7.8. Power Profile for Benchmark for Resource Constraint RC2

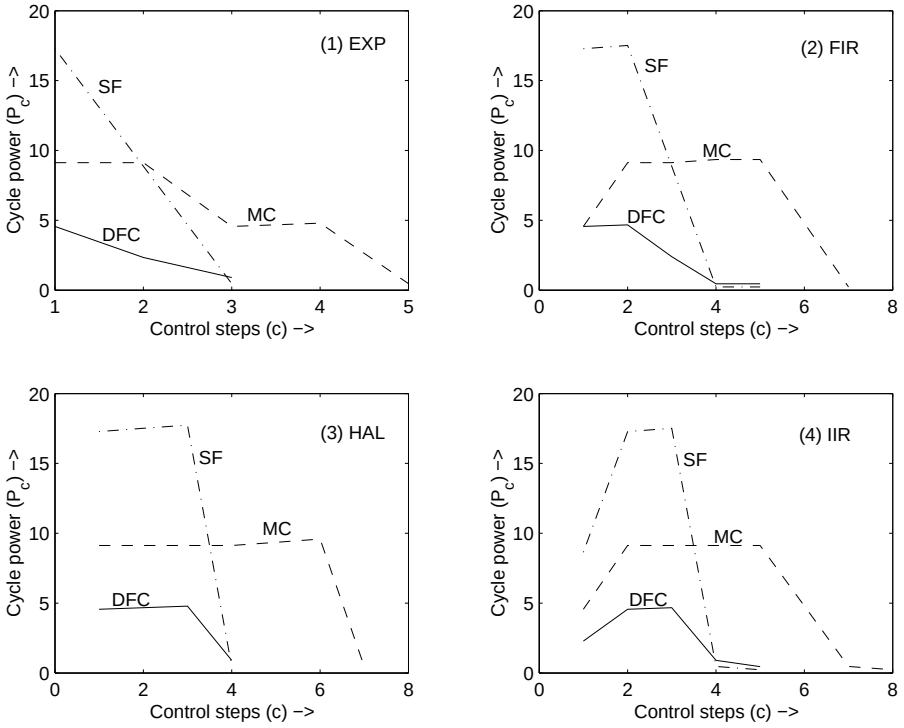


Figure 7.9. Power Profile for Benchmark for Resource Constraint RC3

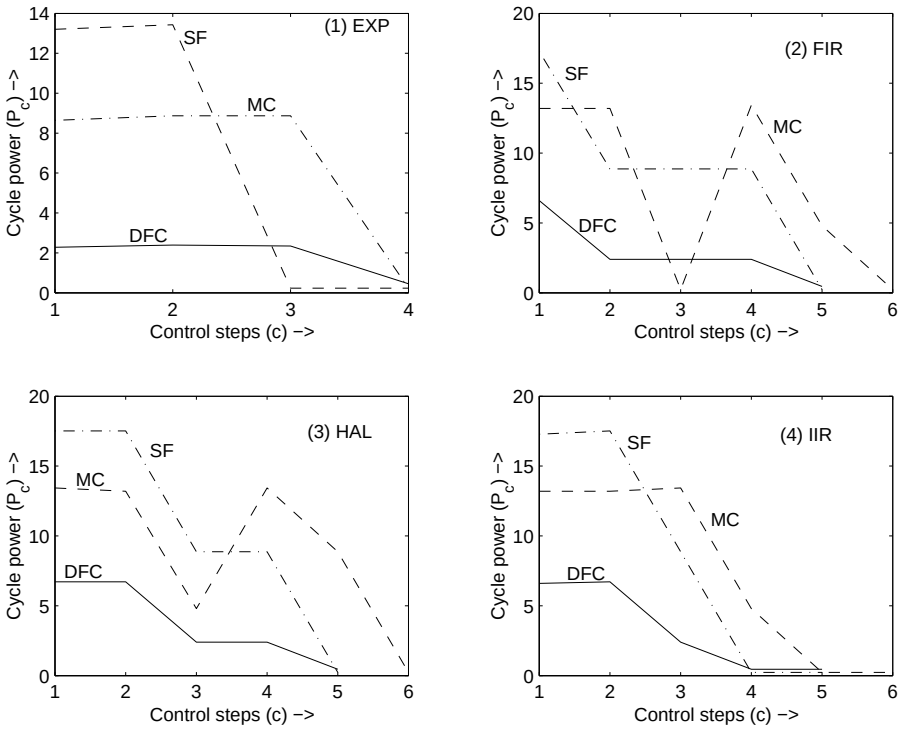


Figure 7.10. Power Profile for Benchmark for Resource Constraint RC4

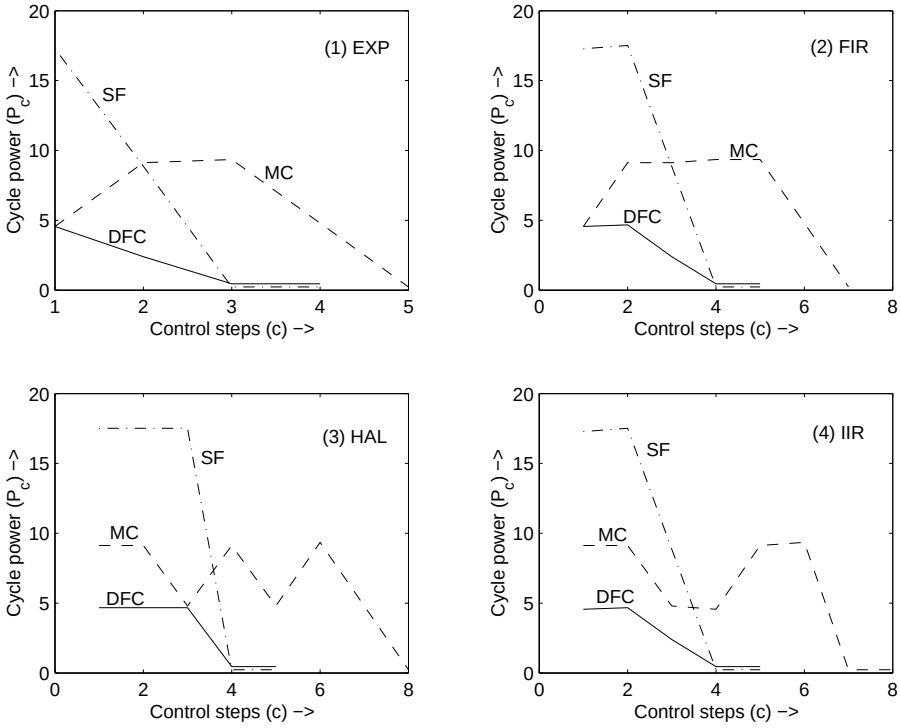


Figure 7.11. Power Profile for Benchmark for Resource Constraint RC5

CHAPTER 8

POWER FLUCTUATION MINIMIZATION

In this chapter, we describe a new datapath scheduling scheme for the reduction of cycle power fluctuation at behavioral level using integer linear programming (ILP) based models [175]. We develop a power model to capture the cycle power fluctuation as cycle-to-cycle power gradient using switching activity, supply voltages and operating frequency. Then, we provide ILP based models for its minimization assuming three modes of circuit operation, such as (1) single supply voltage and single operating frequency (SVSF), (2) multiple supply voltages and dynamic frequency (MVDFC) and (3) multiple supply voltages and multicycling (MVMC). The effectiveness of our scheduling technique is measured by estimating the mean power gradient, the peak power (P_p) consumption, the average power consumption (P_a) and the power delay product (PDP) of the scheduled data flow graph. We compare the MVDFC and MVMC based scheduling algorithms with the results of SVSF based scheduling algorithm. It may be noted that in the case of multiple supply voltage schemes, the power consumption in the level converters is taken into account. Similarly, in hte case of dynamic frequency clocking, the overhead due to dynamic clocking unit is considered. The dynamic frequency clocking methodology is more effective for data intensive signal processing applications. The proposed scheduling algorithms are resource constrained. For the SVSF scheme the resource constraint is the number of functional units. On the other hand, both the MVDFC and MVMC scheduling schemes use the number and type of functional units at different operating voltages as the resource constraints. In addition, the MVDFC scheme uses a certain number of allowable frequencies as resource constraints.

8.1 Power Fluctuation Modeling

In this section, we discuss different power terminologies with reference to a datapath circuit. Let us assume that the datapath is represented in the form of a sequencing data flow graph. The datapath uses various functional units operating at different supply voltages. The level converters are considered as resources operating in the control step in which it needs to step up signal. The dynamic clocking unit (DCU) that generates dynamic frequency is considered as a resource operating in all the control steps. Our aim is to develop power models using generic terms such as switching activity, supply voltages and operating frequencies. The intention of using such parameters is to make the power model a general one, independent of any specific energy or power models. It can accomodate both the look-up table based energy (power) models and energy (power) macro-models. The generic model can also help in easy integration of the proposed power model in a behavioral synthesis tool that uses both behavioral power estimator and datapath scheduler. Moreover, the generic model can be easily tuned to handle any of the three modes of datapath circuit operation, such as (i) single supply voltage and single frequency (SVSF), (ii) multiple supply voltages and dynamic frequency (MVDFC), and (iii) multiple supply voltage and multicycling (MVMC). For MV scheme the datapath uses functional units operating at different supply voltages. In this mode the level converters are considered as resources operating in the control step in which it needs to step up signal.

Let $x_1, x_2, ..., x_n$ be a set of n observations from a given distribution. The sample mean (which is an unbiased estimator for the population mean, μ) is $m = \frac{1}{n}\sum_{i=1}^{n} x_i$. The observation-to-observation gradient can be defined as, $|x_i - x_{i-1}|$, where $2 \leq i \leq n$. The mean gradient is given by $\frac{1}{n-1}\sum_{i=2}^{n} |x_i - x_{i-1}|$. It may be noted that there are $n-1$ gradients for n observations. The notations used in the description is given in Table 8.1. It may be noted that for single frequency and single supply voltage mode of operation, $V_{i,c}$ and f_c are the same for any clock cycle (c) and resource (i). Similarly, for multicycling operation the f_c are the same for any clock cycle (c).

The power consumption for any control step c is given by Eqn. 8.1. This is the total power consumption of all functional units active in control step c. This also includes the power consumption of the level converters where the level converters are considered as resources operating in a cycle

Table 8.1. Notations used in the Description

N	: total number of control steps in the DFG
O	: total number of operations in the DFG
c	: a control step or a clock cycle in DFG ($1 \leq i \leq N$)
o_i	: any operation i, $1 \leq i \leq O$,
P_c	: the total power consumption of all functional units active in control step c (cycle power consumption)
P_p	: peak power consumption for the DFG equal to $max(P_c)_{\forall c}$
P_a	: mean power consumption of the DFG (average P_c)
PG_c	: power gradient for cycle c (where, $c = 2 \rightarrow N$)
PG_p	: peak power gradient of the DFG which is equal to $max(PG_c)_{\forall c}$
MPG	: mean power gradient of the DFG over $c = 2 \rightarrow N$
$FU_{k,v}$	: any functional unit of type k operating at voltage level v
FU_i	: any $FU_{k,v}$ needed by o_i for its execution ($o_i \in FU_{k,v}$)
$FU_{i,c}$	: any functional unit FU_i active in control step c
R_c	: total number of functional units active in step c (same as the number of operations scheduled in c)
$\alpha_{i,c}$	: switching activity of resource $FU_{i,c}$
$V_{i,c}$	: operating voltage of resource $FU_{i,c}$
$C_{i,c}$	: load capacitance of resource $FU_{i,c}$
f_c	: frequency of control step c

c, if the current resource is driven by a resource operating at lower voltage.

$$P_c = \sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c \tag{8.1}$$

The peak power consumption of the DFG is the maximum power consumption over all the control steps which can be expressed as below.

$$P_p = max\left(P_c\right)_{\forall c=1 \rightarrow N} = max\left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right)_{\forall c=1 \rightarrow N} \tag{8.2}$$

The mean cycle power consumption of the DFG (P_a) is defined as,

$$P_a = \frac{1}{N} \sum_{c=1}^{N} P_c = \frac{1}{N} \sum_{c=1}^{N} \left(\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c\right) \tag{8.3}$$

The mean cycle power P_a is an unbiased estimate of the average power consumption of the DFG. The true average power consumption of the DFG is the total energy consumption of the DFG per clock cycle or per second.

The power gradient PG_c for any control step c is defined as the absolute difference of power consumption from the previous control step, as given below.

$$\begin{aligned} PG_c &= |P_c - P_{c-1}|_{(\forall c=2\rightarrow N)} \\ &= \left|\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c - \sum_{i=1}^{R_{c-1}} \alpha_{i,c-1} C_{i,c-1} V_{i,c-1}^2 f_{c-1}\right|_{(\forall c=2\rightarrow N)} \end{aligned} \tag{8.4}$$

The peak of the power gradients is denoted as (PG_p) :

$$\begin{aligned} PG_p &= max\big(|P_c - P_{c-1}|\big)_{\forall c=2\rightarrow N} \\ &= max\left(\left|\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c - \sum_{i=1}^{R_{c-1}} \alpha_{i,c-1} C_{i,c-1} V_{i,c-1}^2 f_{c-1}\right|\right)_{(\forall cc=2\rightarrow N)} \end{aligned} \tag{8.5}$$

The mean power gradient MPG is calculated as,

$$\begin{aligned} MPG &= \tfrac{1}{N-1}\sum_{c=2}^{N} PG_c \\ &= \tfrac{1}{N-1}\sum_{c=2}^{N} |P_c - P_{c-1}| \\ &= \tfrac{1}{N-1}\sum_{c=2}^{N}\left(\left|\sum_{i=1}^{R_c} \alpha_{i,c} C_{i,c} V_{i,c}^2 f_c - \sum_{i=1}^{R_{c-1}} \alpha_{i,c-1} C_{i,c-1} V_{i,c-1}^2 f_{c-1}\right|\right) \end{aligned} \tag{8.6}$$

The above generic power models are independent of any specific energy or power models. Using the dynamic energy model proposed in [51] we can express the effective switching capacitance of our proposed model as,

$$\alpha_i C_i = C_{swi}({\alpha_i}^1, {\alpha_i}^2) \tag{8.7}$$

Here, the α_i and C_i are the parameters corresponding to the functional unit FU_i as defined before. The C_{swi} is a measure of the effective switching capacitance of functional unit FU_i, which is a function of ${\alpha_i}^1$ and ${\alpha_i}^2$; the ${\alpha_i}^1$ and ${\alpha_i}^2$ are the average switching activities on the first and second input operands of FU_i. Similarly, any other power or energy models can be incorporated. It should be noted that the above switching model (in Eqn. 8.7) handles input pattern dependencies. Using

the above Eqn. 8.7 we can rewrite Eqn. 8.6 as follows.

$$MPG = \frac{1}{N-1}\sum_{c=2}^{N}\left(\left|\sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c - \sum_{i=1}^{R_{c-1}} C_{swi,c-1}V_{i,c-1}^2 f_{c-1}\right|\right) \quad (8.8)$$

We use the above MPG as the objective function for low power datapath scheduling. We make the following observations about the MPG. It is a *non-linear* function because of the absolute function (abs or $|\ \ |$). It is a function of parameters, such as switching activity, capacitance, operating voltage and operating frequency. We will use the ILP formulations to minimize MPG through datapath scheduling for three modes of datapath operation, namely SVSF, MVDFC and MVMC as described before.

The critical path delay of the DFG can be calculated as,

$$T = \sum_{i=1}^{N} \frac{1}{f_c} \quad (8.9)$$

It should be noted that the f_c is the same for single frequency and multicycling operations for all values of c and may be different for dynamic frequency clocking operations. The power delay product of the DFG is defined as the product of the average power consumption and critical path delay as shown below.

$$PDP = P_a * T \quad (8.10)$$

Using Eqn. 8.3, 8.7, and 8.9, we have the following expression for the power delay product.

$$PDP = \frac{1}{N}\sum_{i=1}^{N}\sum_{i=1}^{R_c} C_{swi,c}V_{i,c}^2 f_c * \sum_{i=1}^{N}\frac{1}{f_c} \quad (8.11)$$

To study the impact of the scheduling algorithms on the performance of the datapath we estimate the power delay product of the scheduled DFGs using the above expression.

8.2 Modeling of Non-linearities

It is clear from the Eqn. 8.8 that the MPG is a *non-linear function*. The nonlinearity is because of the presence of *absolute function* (abs or $|\ \ |$). The ILP formulations has to handle this form of

non-linearity. In this section, we address the transformations that help in linear modelling of the nonlinear functions. The general form of linear programming can be represented as [173, 174] :

$$\begin{aligned} \text{Minimize :} \quad & \textstyle\sum_i |y_i| \\ \text{Subject to :} \quad & y_i + \textstyle\sum_j a_{ij} * x_j \leq b_i, \ \forall i \\ & x_j \geq 0, \quad \forall j \end{aligned} \tag{8.12}$$

where, y_i, is the deviation between the prediction and observation. The $|y_i|$ is non-linear because of absolute function. This can be linearized using the following transformation.

Let, y_i be represented as difference of two non-negative variables,

$$y_i = y_i^1 - y_i^2. \tag{8.13}$$

Using these new variables we can reexpress the LP problem in Eqn. 8.12 as follows.

$$\begin{aligned} \text{Minimize :} \quad & \textstyle\sum_i \left|y_i^1 - y_i^2\right| \\ \text{Subject to :} \quad & y_i^1 - y_i^2 + \textstyle\sum_j a_{ij} * x_j \leq b_i, \ \ \forall i \\ & x_j \geq 0, \ \ \forall j \\ & y_i^1, y_i^2 \geq 0, \ \ \forall i \end{aligned} \tag{8.14}$$

If the product of y_i^1 and y_i^2 is zero, then

$$\left|y_i^1 - y_i^2\right| \quad = \quad \left|y_i^1\right| + \left|y_i^2\right| \quad = \quad y_i^1 + y_i^2 \tag{8.15}$$

Using the above, we can write the LP problem in Eqn. 8.14 as shown below.

$$\begin{aligned} \text{Minimize :} \quad & \textstyle\sum_i y_i^1 + y_i^2 \\ \text{Subject to :} \quad & y_i^1 - y_i^2 + \textstyle\sum_j a_{ij} * x_j \leq b_i, \ \ \forall i \\ & x_j \geq 0, \ \ \forall j \\ & y_i^1, y_i^2 \geq 0, \ \ \forall i \end{aligned} \tag{8.16}$$

The problem in Eqn. 8.12 and 8.16 are equivalent and minimization of Eqn. 8.16 will result in minimization of Eqn. 8.12.

8.3 ILP Formulations to Minimize Mean Power Gradient

In this section, we discuss the ILP models for minimization of MPG for various modes of datapath operations, such as SVSF, MVDFC and MVMC. It may be noted that different decision variables are to be used for the three different modes. We first discuss the formulations using MVDFC followed by MVMC. The formulation for SVSF is not presented since it is trivial one. The notations used in ILP formulations is given in Table 8.2.

Table 8.2. Notations used in ILP formulations

$M_{k,v}$	: maximum number of functional units $FU_{k,v}$
S_i	: as soon as possible (ASAP) time stamp for the operation o_i
E_i	: as late as possible (ALAP) time stamp for the operation o_i
$P(C_{swi}, v, f)$	: power consumption of functional unit FU_i at voltage v and frequency f used by o_i for its execution
$x_{i,c,v,f}$	: decision variable which takes the value of 1 if operation o_i is scheduled in control step c using the functional unit $F_{k,v}$ and c has frequency f_c
$y_{i,v,l,m}$	: decision variable which takes the value of 1 if operation o_i is using any $F_{k,v}$ and scheduled in control steps $l \rightarrow m$
$L_{i,v}$	: latency for operation o_i using resource operating at voltage v (in terms of number of clock cycles)

8.3.1 Formulations using Multiple Voltages and Dynamic Frequency

In dynamic frequency clocking [59, 62], the clock frequency is varied on-the-fly based on the functional units active in that cycle. In this clocking scheme, all the units are clocked by a single clock line which switches at run-time. The frequency reduction creates an opportunity to operate the different functional units at different voltages, which in turn, helps in further reduction of power.

Objective Function : The objective is to minimize the mean power gradient MPG described

in Eqn. 8.8 of the whole DFG over all control steps.

$$\text{Minimize :} \quad MPG \tag{8.17}$$

Using Eqn. 8.6, this can be restated as :

$$\text{Minimize :} \quad \frac{1}{N-1}\sum_{c=2}^{N}|P_c - P_{c-1}| \tag{8.18}$$

This problem has the non-linearity in it because of the absolute function. This can be converted to an equivalent problem using the transformation suggested in the previous section.

$$\begin{aligned} &\text{Minimize :} \quad \frac{1}{N-1}\sum_{c=2}^{N}(P_c + P_{c-1}) \\ &\text{Subject to :} \quad \text{Power gradient constraints} \end{aligned} \tag{8.19}$$

The above problem in Eqn. 8.19 is simplified to :

$$\begin{aligned} &\text{Minimize :} \quad \frac{2}{N-1}\sum_{c=2}^{N-1}P_c + P_1 + P_N \\ &\text{Subject to :} \quad \text{Power gradient constraints} \end{aligned} \tag{8.20}$$

Using the decision variables and above LP objective function is formulated as,

$$\begin{aligned} \text{Minimize :} &\left(\frac{2}{N-1}\right)\sum_{c=2}^{N-1}\sum_{i\in F_{k,v}}\sum_{v}\sum_{f}x_{i,c,v,f}P(C_{swi},v,f) + \sum_{i\in F_{k,v}}\sum_{v}\sum_{f}x_{i,1,v,f}P(C_{swi},v,f) \\ &+ \sum_{i\in F_{k,v}}\sum_{v}\sum_{f}x_{i,N,v,f}P(C_{swi},v,f) \\ \text{Subject to :} \quad &\text{Power gradient constraints} \end{aligned} \tag{8.21}$$

Uniqueness Constraints : These constraints ensure that every operation o_i is scheduled to one unique control step within the mobility range (S_i, E_i) with a particular supply voltage and operating frequency. We represent them as, $\forall i$, $1 \leq i \leq O$,

$$\sum_{c}\sum_{v}\sum_{f}x_{i,c,v,f} \quad = \quad 1 \tag{8.22}$$

Precedence Constraints : These constraints guarantee that for an operation o_i, all its predecessors are scheduled in an earlier control step and its successors are scheduled in a later control step. These are modelled as, $\forall i, j, o_i \in Pred_{o_j}$,

$$\sum_v \sum_f \sum_{d=S_i}^{E_i} d * x_{i,d,v,f} - \sum_v \sum_f \sum_{e=S_j}^{E_j} e * x_{j,e,v,f} \quad \leq \quad -1 \tag{8.23}$$

Resource Constraints : These constraints make sure that no control step contains more than $F_{k,v}$ operations of type k operating at voltage v. These can be enforced as, $\forall c$, $1 \leq c \leq N$ and $\forall v$,

$$\sum_{i \in F_{k,v}} \sum_f x_{i,c,v,f} \quad \leq \quad M_{k,v} \tag{8.24}$$

Frequency Constraints : This set ensures that if a functional unit is operating at higher voltage level then it can be scheduled in a lower frequency control step, whereas, a functional unit is operating at lower voltage level then it can not be scheduled in a higher frequency control step. We write these constraints as, $\forall i$, $1 \leq i \leq O$, $\forall c$, $1 \leq c \leq N$, if $f < v$, then $x_{i,c,v,f} = 0$.

Power Gradient Constraints : To eliminate the non-linearity introduced due to the absolute function, we introduce these constraints (as outlined in Eqn. 8.19, 8.20 and 8.21), $\forall c$, $2 \leq c \leq N$,

$$\sum_{i \in F_{k,v}} \sum_v \sum_f x_{i,c,v,f} * P(C_{swi}, v, f) - \sum_{i \in F_{k,v}} \sum_v \sum_f x_{i,c-1,v,f} * P(C_{swi}, v, f) \quad \leq \quad PG_p \tag{8.25}$$

The PG_p is peak power gradient constraint added to the objective function and minimized alongwith it.

8.3.2 Formulations using Multiple Supply Voltages and Multicycling

In this subsection, we describe the ILP formulations for the minimization of MPG using multiple supply voltages and multicycling. In this scheme, the functional units are operated at multiple supply voltages and the lower operating voltage functional units are scheduled in consecutive control steps.

Objective Function : The objective is to minimize the mean power gradient MPG described in Eqn. 8.8 of the whole DFG over all control steps.

$$\text{Minimize :} \quad MPG \tag{8.26}$$

Using Eqn. 8.6, this can be restated as :

$$\text{Minimize :} \quad \frac{1}{N-1}\sum_{c=2}^{N}|P_c - P_{c-1}| \tag{8.27}$$

This problem has the non-linearity in it because of the absolute function. This can be converted to an equivalent problem using the transformation suggested in the previous section.

$$\begin{aligned} &\text{Minimize :} \quad \frac{1}{N-1}\sum_{c=2}^{N}(P_c + P_{c-1}) \\ &\text{Subject to :} \quad \text{Power gradient constraints} \end{aligned} \tag{8.28}$$

The above problem in Eqn. 8.28 is simplified to : Following the similar steps as in the previous section (section 8.3.1) and using the transformations, we redefine the objective function.

$$\begin{aligned} &\text{Minimize :} \quad \frac{2}{N-1}\sum_{c=2}^{N-1} P_c + P_1 + P_N \\ &\text{Subject to :} \quad \text{Power gradient constraints} \end{aligned} \tag{8.29}$$

Then, using the decision variables the objective function is formulated as,

$$\begin{aligned} \text{Minimize :} \quad & \left(\frac{2}{N-1}\right)\sum_{l=2}^{N-1}\sum_{i\in F_{k,v}}\sum_{v} y_{i,v,l,(l+L_{i,v}-1)}P(C_{swi},v,f) \\ & + \sum_{i\in F_{k,v}}\sum_{v}\sum_{f} y_{i,v,1,1}P(C_{swi},v,f) \\ & + \sum_{i\in F_{k,v}}\sum_{v}\sum_{f} y_{i,v,N,N}P(C_{swi},v,f) \\ \text{Subject to :} \quad & \text{Power gradient constraints} \end{aligned}$$

Uniqueness Constraints : These constraints ensure that every operation o_i is scheduled in appropriate control steps within the mobility range (S_i, E_i) with a particular supply voltage. Depending

on the supply voltage it may be operated at more than one clock cycle. We represent them as, $\forall i$, $1 \leq i \leq O$,

$$\sum_v \sum_{l=S_i}^{S_i+E_i+1-L_{i,v}} y_{i,v,l,(l+L_{i,v}-1)} \quad = \quad 1 \tag{8.31}$$

When the operators are operating at highest voltage, they are scheduled in one unique control step, whereas, when they are to be operated at lower voltages they need more than one clock cycle for completion. Thus, for lower voltage the mobility is restricted.

Precedence Constraints : These constraints guarantee that for an operation o_i, all its predecessors are scheduled in an earlier control step and its successors are scheduled in a later control step. These constraints should also take care of the multicycling operations. These are modelled as, $\forall i, j, o_i \in Pred_{o_j}$,

$$\sum_v \sum_{l=S_i}^{E_i} (l + L_{i,v} - 1) * y_{i,v,l,(l+L_{i,v}-1)} - \sum_v \sum_{l=S_j}^{E_j} l * y_{j,v,l,(l+L_{j,v}-1)} \quad \leq \quad -1 \tag{8.32}$$

Resource Constraints : These constraints make sure that no control step contains more than $F_{k,v}$ operations of type k operating at voltage v. These can be enforced as, $\forall v$ and $\forall l$, $1 \leq l \leq N$,

$$\sum_{i \in F_{k,v}} \sum_l y_{i,v,l,(l+L_{i,v}-1)} \quad \leq \quad M_{k,v} \tag{8.33}$$

Power Gradient Constraints : These constraints are introduced to eliminate the absolute function non-linearity of the objective function. These constraints can be enforced as, $\forall l$, $2 \leq l \leq N$,

$$\begin{aligned} &\sum_{i \in F_{k,v}} \sum_v y_{i,v,l,(l+L_{i,v}-1)} * P(C_{swi}, v, f_{clk}) \\ &- \sum_{i \in F_{k,v}} \sum_v y_{i,v,(l-1),(l+L_{i,v}-2)} * P(C_{swi}, v, f_{clk}) \quad \leq \quad PG_p \end{aligned} \tag{8.34}$$

Where, PG_p is power gradient constraint which is added to the objective at minimized alongwith it.

8.4 Scheduling Algorithm

In this section, we will discuss the solutions for the ILP formulations obtained in the previous section and develop scheduling algorithms for both MVDFC and MVMC schemes. The target architecture model assumed by the scheduling schemes is same as the one used in [65]. All functional units have a register each and a multiplexor. Each functional unit feeds a single register. The register and the multiplexor operate at the same voltage level as that of the functional units. Level converters are used when a low-voltage functional unit is driving a high-voltage functional unit [65, 95]. A controller decides which of the functional units are active in each control step and those that are not active are disabled using the multiplexors. For MVDFC scheme, the controller has a storage unit to store the parameters, cycle frequency index (cfi_c) obtained from the scheduling, which serves as clock dividing factor for the dynamic clocking unit. The cycle frequency f_c is generated dynamically and a functional unit operating at one of the supply voltages is activated.

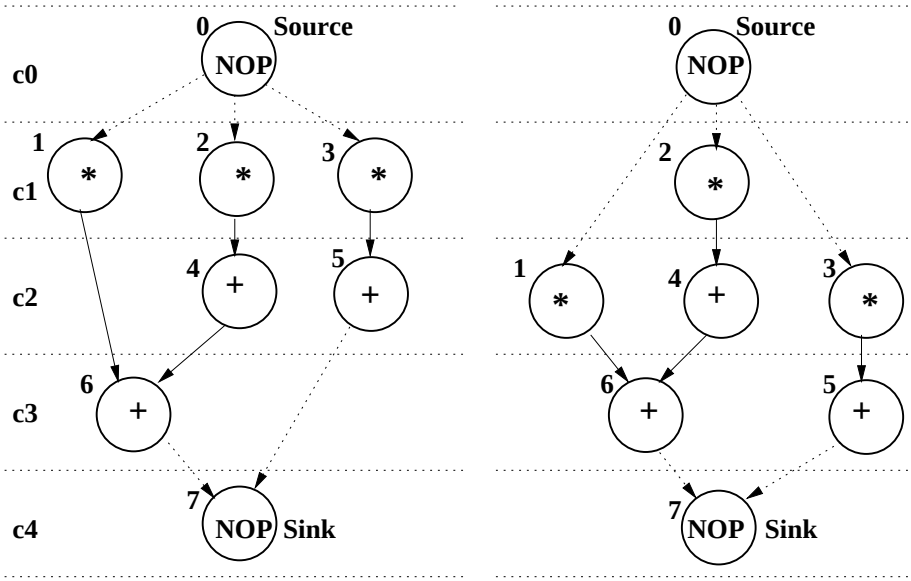
The inputs to the algorithm are an unscheduled data flow graph (UDFG), the resource constraints, the number of allowable voltage levels (L_V), the number of allowable frequencies (L_f), delay of each resource (d_{FU}), multiplexor (d_{Mux}), register (d_{Reg}) at different voltage levels. The delays of level converters (d_{Conv}) is represented in the form of a matrix that shows the delay in converting one at voltage level V_i to another voltage level V_j (where, both $V_i, V_j \in V_{L_V}$). The resource constraint includes the number of ALUs and multipliers at different voltage levels V_i (where, $V_i \in V_{L_V}$). The scheduling algorithm determines the proper time stamp for each operation, f_{base}, cfi_c (using [48]) and voltage level such that the function MPG (Eqn. 8.8) is minimum.

The ILP based scheduler which minimizes modified cycle power profile function of the DFG is outlined in Fig. 8.1. In step 1, the scheduler constructs a look-up table for effective switching capacitance for known values of average switching activity pair as described in Eqn. 8.7. In step 2, the scheduler determines the switching activities at the inputs of each node by using behavioral simulation of DFG. For this purpose, different set of application specific input vectors (having different correlations) are given at the primary inputs of the DFG and average swtiching activity at each inputs of other nodes are calculated [167, 169]. It should be noted that if the look-up table (in step 1) does not have the switching capacitance for a pair of input average swtiching activities

Input	: DFG, Constraints, Voltage and Freq. Levels, Delays
Output	: Scheduled DFG, f_{base}, N, cfi_c, Power estimates
Step 1	: Construct effective switching capacitance look-up table.
Step 2	: Calculate the switching activities for each node.
Step 3	: Find ASAP and ALAP schedule of the UDFG.
Step 4	: Determine the mobility graphs for different schemes.
Step 5	: Calculate operating frequency of FUs using delays.
Step 6	: Model the ILP formulations of DFG using AMPL.
Step 7	: Solve the ILP formulations using LP-Solve.
Step 8	: Obtain the scheduled DFG.
Step 9	: Determine f_c, f_{base} and cfi_c for MVDFC scheme.
Step 10	: Estimate the power and delay of the scheduled DFG.

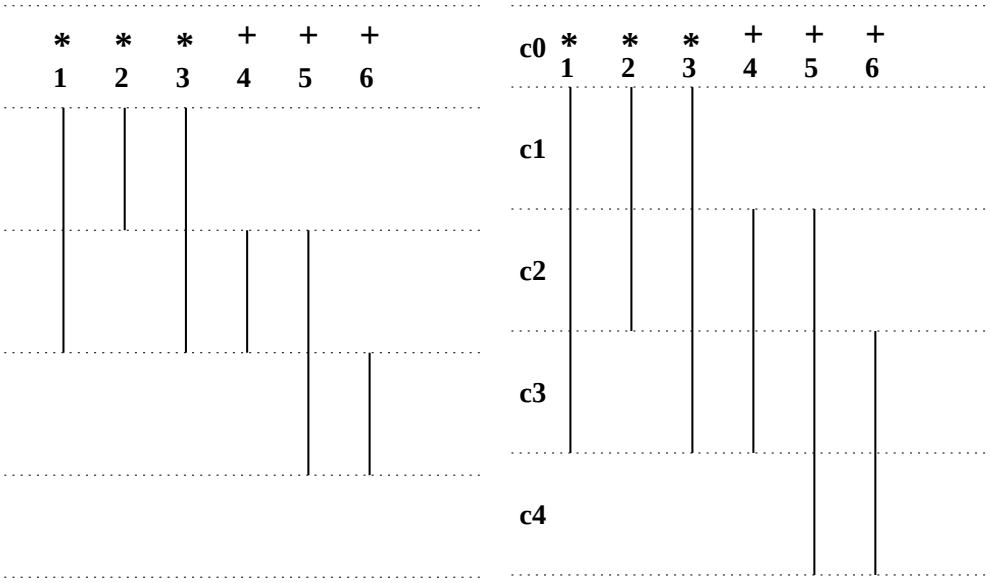
Figure 8.1. Scheduling for MPG Minimization

(in step 2), then the scheduler uses interpolation techniques to find the same. The third step is to determine the as soon as possible (ASAP) time stamp of each operation. The fourth step is the determination of the as late as possible (ALAP) time stamp of each vertex for the DFG. The ASAP time stamp is the start time and ALAP time stamp is the finish time of each operation. These two time stamps provide the mobility of a operation and the operation must be scheduled in this mobile range. This mobility graph needs to be modified for the MVMC scheme. Then the scheduler finds the ILP formulations based on the models described before. The scheduler uses modeling language AMPL to model the ILP formulations [166]. At this step, we calculate the power consumption of the functional units as follows. The operational delay of a functional unit is assumed as ($d_{FU} + d_{Mux} + d_{Reg} + d_{Conv}$). For the MVMC scheme the operating frequency is the frequency corresponding to operational delay at the highest operating voltage of multiplier unit. On the other hand, for MVDFC scheme operating frequency of a functional unit is assumed to be the inverse of operational delay of a functional unit at corresponding supply voltage. We get the switching capacitance from step 1 and step 2, and for different operating voltages and frequencies the power values are calculated whenever necessary. After the ILP formulation is solved using LP-Solve the scheduled DFG is obtained. Then, the scheduler determines the cycle frequencies for MVDFC scheme using the methods proposed in [48]. Finally, power consumptions, energy consumptions and energy delay product of the scheduled DFG is calculated.



(a) ASAP Schedule

(b) ALAP Schedule



(c) Mobility for MVDFC

(d) Mobility for MVMC

Figure 8.2. Example Data Flow Graph (DFG)

We illustrate the solution for the ILP formulations with the help of the DFG shown in Fig. 8.2. The ASAP schedule is shown in Fig. 8.2(a) and the ALAP schedule is shown in Fig. 8.2(b). From the ASAP and ALAP scheduling we obtained the mobility graphs shown in Fig. 8.2(c) and Fig. 8.2(d) for MVDFC and MVMC schemes respectively. Using these mobility graphs, we get the ILP formulations. We solved the formulation using LP-solve and based on the results, we obtained the scheduled DFG. In this MVMC case, the mobility graph considers the multicycle operations. In this illustration, we assume that we have two operating voltage levels, and when the multipliers are operated at lower voltage, they take two clock cycles. It should be noted that the mobility graph will depend on the number of operating voltages and the assumed operating frequency.

8.5 Experimental Results

In this section we discuss the experiments conducted for the scheduling schemes proposed in the previous sections. The ILP based schedulers for all three schemes (SVSF, MVDFC and MVMC) are tested with five benchmark circuits :

- Example circuit (EXP) (8 nodes, 3*, 3+, 9 edges)
- FIR filter (11 nodes, 5*, 4+, 19 edges)
- IIR filter (11 nodes, 5*, 4+, 19 edges)
- HAL differential equation solver (13 nodes, 6*, 2+, 2-, 1 $<$, 16 edges)
- Auto-Regressive filter (ARF) (15 nodes, 5*, 8+, 19 edges).

The following notations are used to express results are given in Table 8.3.

We use the look-up table method for average switching capacitance calculation. The look-up table construction consists of two phases, such as input pattern generation and cell characterization. We generate the primary input signal of different correlations using the autoregressive moving average (ARMA) model [169]. We perform the characterization of the physical implementations of the library modules available in [55] by applying the the input patterns generated above for known values of $({\alpha_i}^1, {\alpha_i}^2)$ pairs. Whenever necessary, we used interpolation method to find the

Table 8.3. Notations used in Describing the Results

MPG_S	: the mean power gradient (in mW) for SVSF operation
MPG_D	: the mean power gradient (in mW) for MVDFC operation
MPG_M	: the mean power gradient (in mW) for MVMC operation
P_{pS}	: the peak power consumption (in mW) for SVSF operation
P_{pD}	: the peak power consumption (in mW) for MVDFC operation
P_{pM}	: the peak power consumption (in mW) for MVMC operation
P_{aS}	: the average power consumption (in mW) for SVSF operation
P_{aD}	: the average power consumption (in mW) for MVDFC operation
P_{aM}	: the average power consumption (in mW) for MVMC operation
T_S	: the critical path delay (in ns) for SVSF operation
T_D	: the critical path delay (in ns) for MVDFC operation
T_M	: the critical path delay (in ns) for MVMC operation
PDP_S	: the power delay product (in nJ) for SVSF operation
PDP_D	: the power delay product (in nJ) for MVDFC operation $(= P_{aD} * T_D)$
PDP_M	: the power delay product (in nJ) for MVMC operation $(= P_{aM} * T_M)$
ΔP_{pD}	: percentage peak power reduction for MVDFC operation $(= \frac{(P_{pS}-P_{pD})}{P_{pS}} * 100)$
ΔP_{pM}	: percentage peak power reduction for MVMC operation $(= \frac{(P_{pS}-P_{pM})}{P_{pS}} * 100)$
ΔPDP_D	: percentage PDP reduction for MVDFC operation $(= \frac{(PDP_S-PDP_D)}{PDP_S} * 100)$
ΔPDP_M	: percentage PDP reduction for MVMC operation $(= \frac{(PDP_S-PDP_M)}{PDP_S} * 100)$

average switching capacitance for any other values of $(\alpha_i{}^1, \alpha_i{}^2)$ pairs that does not exist in the look-up table. It should be noted that larger the size of look-up table, better is the accuracy. Our look-up table has 100 pairs of entries for $(\alpha_i{}^1, \alpha_i{}^2)$. The above generated signals are propagated through different operators in the DFG and the average switching activities are calculated as described in [169].

The schedulers were tested using different sets of resource constraints (RC1,RC2,RC3,RC4,RC5) shown below.

multipliers (2 at $2.4V$ and 1 at $3.3V$) and ALUs (1 at $2.4V$ and 1 at $3.3V$)
multipliers (3 at $2.4V$) and ALUs (1 at $2.4V$ and 1 at $3.3V$)
multipliers (2 at $2.4V$) and ALUs (2 at $3.3V$)
multipliers (1 at $2.4V$ and 1 at $3.3V$) and ALUs (1 at $3.3V$)
multipliers (2 at $2.4V$) and ALUs (1 at $3.3V$)

Table 8.4. Power Estimates for Benchmarks

	MPG Estimates (mW)					Peak Power (%)		Average Power (%)		PDP (%)	
	MPG_S	MPG_D	ΔMPG_D	MPG_M	ΔMPG_M	ΔP_{pD}	ΔP_{pM}	ΔP_{aD}	ΔP_{aM}	ΔPDP_D	ΔPDP_M
1	2	3	4	5	6	7	8	9	10	11	12
e	8.42	2.11	74.94	5.96	29.22	73.61	0	72.80	22.91	54.58	0
x	8.42	2.11	74.94	5.97	29.10	73.61	20.83	72.80	21.56	54.58	0
p	8.42	2.06	75.53	2.17	74.23	73.61	47.22	72.12	36.68	53.56	0
f	4.26	1.11	73.94	3.53	17.14	73.61	0	73.47	15.65	52.24	0
i	6.42	1.72	73.21	4.54	29.28	73.61	47.22	73.47	12.93	52.24	0
r	4.26	1.08	74.65	3.00	29.58	73.61	45.90	72.9	24.72	51.22	0
i	8.56	2.92	65.89	4.41	48.48	65.74	31.48	68.33	18.78	52.24	0
i	8.56	2.24	73.83	2.71	68.34	73.61	47.22	72.96	30.13	59.60	0
r	4.26	1.08	74.65	1.27	70.19	73.61	47.22	72.34	34.13	55.71	0
h	8.49	2.85	66.43	3.53	58.42	65.74	31.48	69.26	32.55	46.09	0
a	8.56	2.19	74.42	4.52	47.20	73.60	47.20	73.18	30.14	53.06	0
l	4.26	1.06	75.12	1.63	61.74	73.33	45.35	72.71	24.64	50.85	0
a	5.66	1.46	74.20	2.92	48.41	73.59	0	74.00	22.00	59.40	0
r	5.66	1.46	74.20	3.00	47.00	73.59	0	74.00	20.44	59.40	0
f	5.66	1.40	75.27	2.97	47.53	73.02	0	71.33	18.89	57.20	0
Average Results			73.42		47.10	72.50	27.41	72.38	24.41	54.13	0

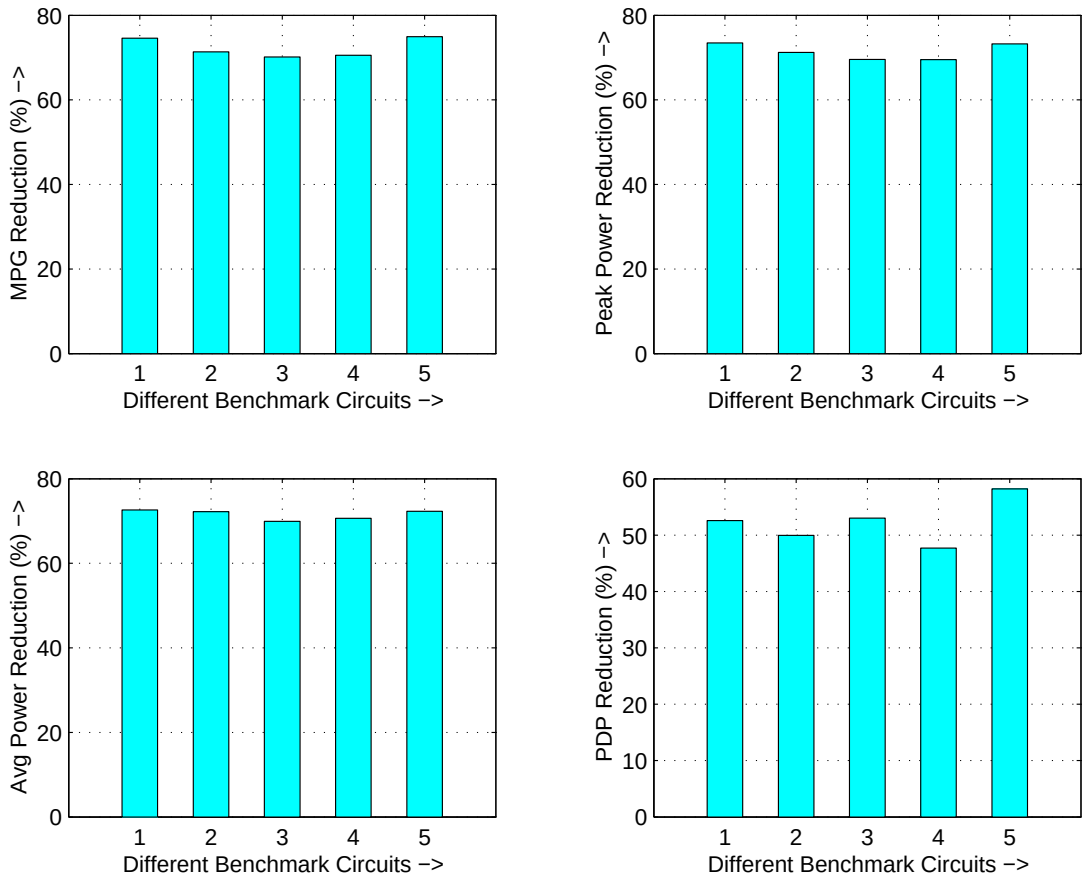


Figure 8.3. Average Reductions using DFC Scheme

The reason behind choosing the sets of resource constraints is that it covers a good representive of types of resources at different operating voltages. The number of allowable voltage levels being two $(2.4V, 3.3V)$ and maximum number of allowable frequencies being three. The experimental results for various benchmark circuits are reported in Table 8.4 for all three schemes for resource constraints RC2, RC3, and RC5. The power estimation step includes the power consumption of the overheads. In case of MVDFC scheduling the frequencies found out are $4.5MHz, 9MHz$ and $18MHz$. For MVMC and SVSF scheduling scheme the operating frequency (f_{clk}) is $9MHz$. The table also reports the average reduction for different benchmarks averaged over all resource constraints. It is obvious from the table that the reductions using MVDFC scheme are appreciable, on the other hand, for the MVMC scheme there is no reduction in PDP. The average results over all five resource constraints are shown in Fig. 8.3 and 8.4.

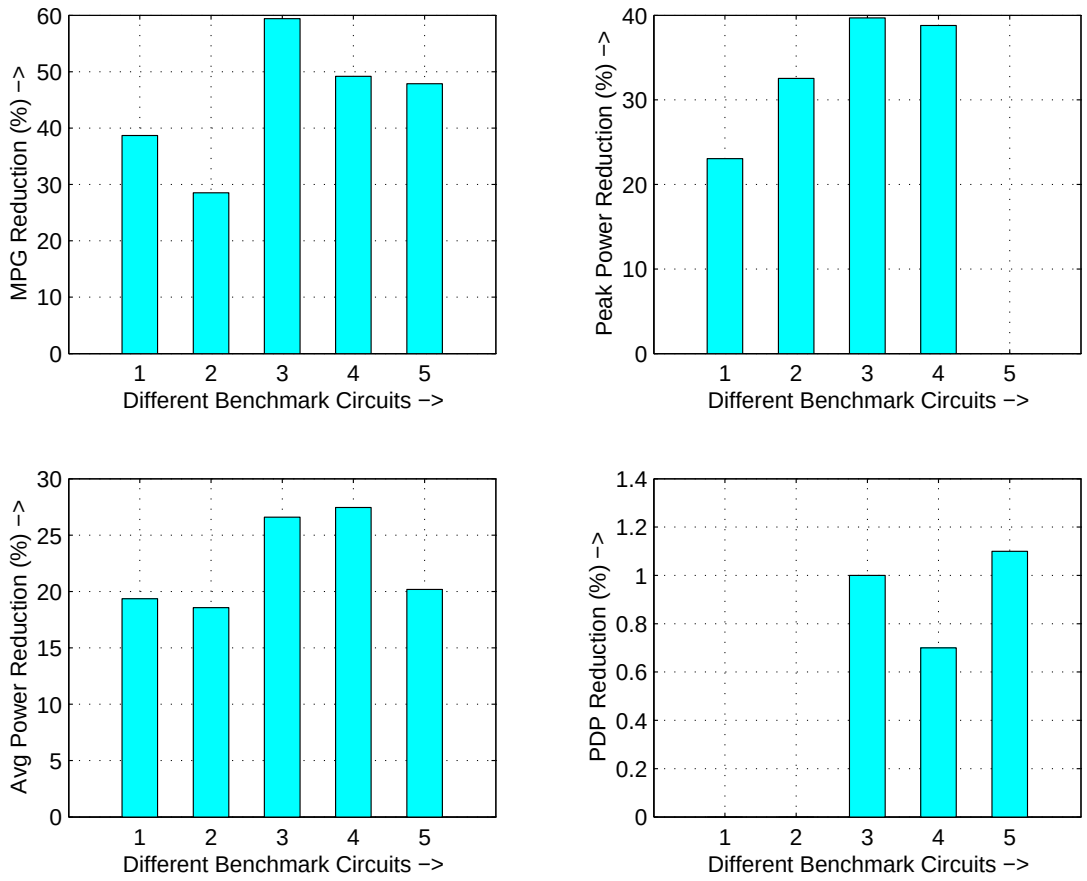


Figure 8.4. Average Reductions using Multicycling Scheme

In order to study the power consumption per cycle, we plotted the power profile for different benchmarks over all the control steps (clock steps). Fig. 8.5, 8.6 and 8.7 show power profile for benchmarks for resource constraints RC2, RC3, and RC5 respectively. The curves labeled as ”SF” correspond to the profile when the schedule is operated at a single frequency (which is the maximum frequency of slower operator, multiplier) and single voltage. The profiles labeled as ”DFC” correspond to the case when dynamic clocking and multiple voltage scheme is used. Similarly, the profiles labeled as ”MC” is for the MVMC scheme. The effectiveness of the proposed scheduling schemes is obvious from the figures.

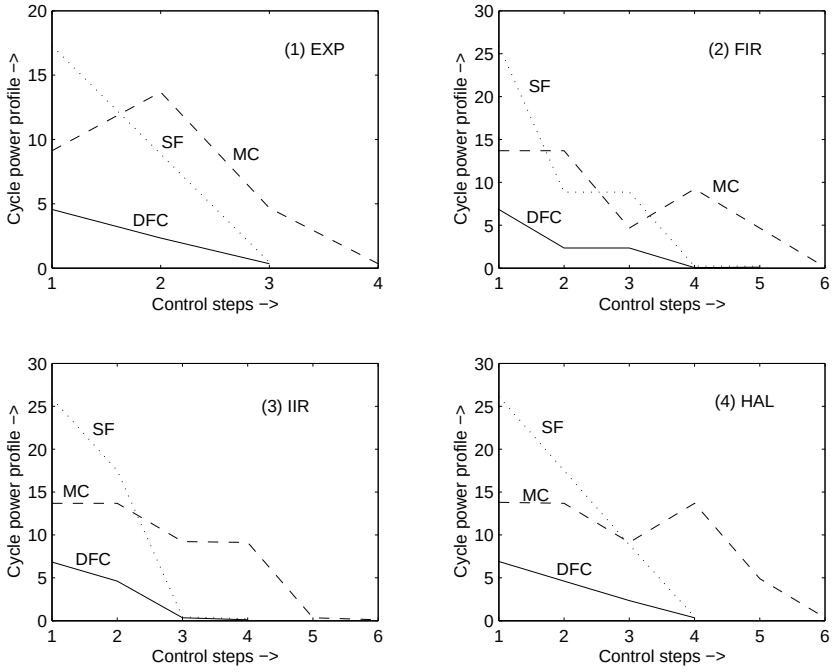


Figure 8.5. Power Profiles for Benchmarks (for RC2)

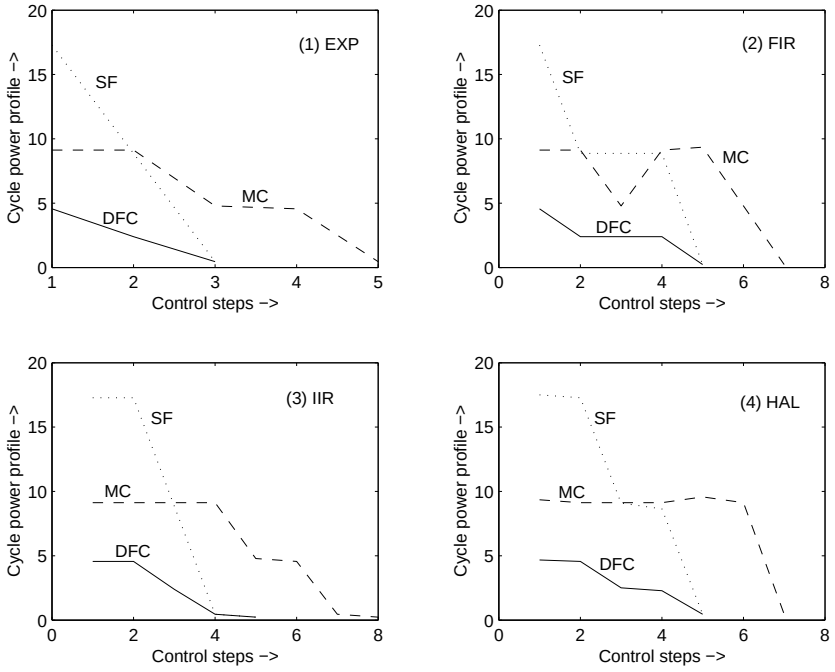


Figure 8.6. Power Profiles for Benchmarks (for RC3)

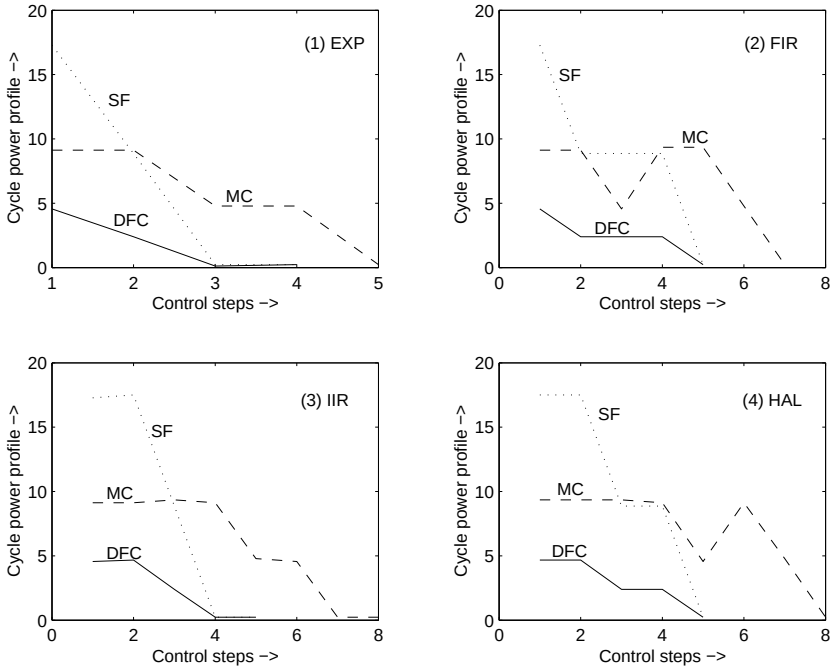


Figure 8.7. Power Profiles for Benchmarks (for RC5)

8.6 Conclusions

The reduction of cycle power fluctuation is important for a CMOS circuit. This paper addresses power fluctuation reduction at the behavioral level using low power datapath scheduling techniques. Three datapath scheduling schemes, (i) using single supply voltages and single frequency (SVSF), (ii) using multiple supply voltage and dynamic clocking (MVDFC) and (iii) using multiple supply voltage and multicycling (MVMC) have been introduced. We used ILP based optimizations for the three modes of datapath operations. The results of MVDFC and MVMC schemes were compared with that of SVSF scheme. In dynamic frequency clocking scheme significant reduction could be achieved in mean power gradient, peak power and average power alongwith reductions in power delay product. *The results clearly indicate that the dynamic frequency clocking is a better scheme than the multicycling approach for power minimization.* The effectiveness of the scheduling schemes in the context of pipelined datapath and control intensive applications need to be investigated.

CHAPTER 9

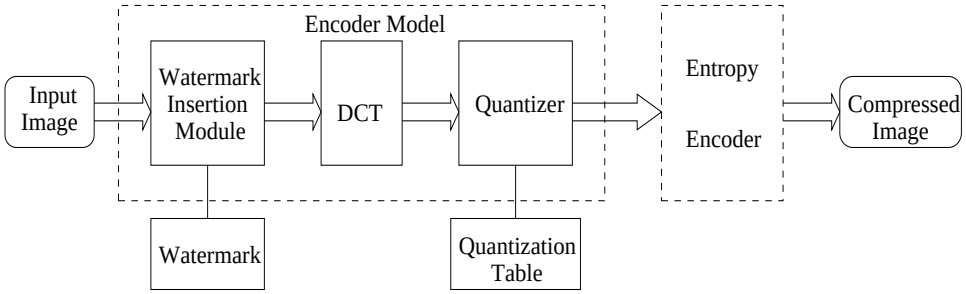
VLSI DESIGN FOR DIGITAL WATERMARKING OF IMAGES

The research in digital watermarking is well matured. Several watermarking algorithms have been proposed for image, video, audio and text in the current literature. Digital Watermarking is the process that embeds data called a watermark into a multimedia object such that watermark can be detected or extracted later to make an assertion about the object. The software implementation of the proposed algorithms are significantly large, whereas the hardware implementation of the algorithms is lacking. The hardware implementation has advantages over the software implementation in terms of low power, high performance, and reliability. In this chapter, we develop hardware system that can insert invisible robust, invisible fragile and visible watermark in the image. The hardware module can be easily incorporated in JPEG encoder to develop a secure JPEG encoder. An outline of such an secure JPEG encoder is provides in Fig. 9.1 [176]. The secure JPEG codec can be a part of a scanner or a digital camera so that the digitized images are wateramarked right at the origin. The proposed watermarking chip can also directed integrated with any existing digital still camera. We provide the schematic view of a still camera having inbuilt watermarking chip in Fig. 9.2, call such an camera as a "secure digital still camera" (S^2DC).

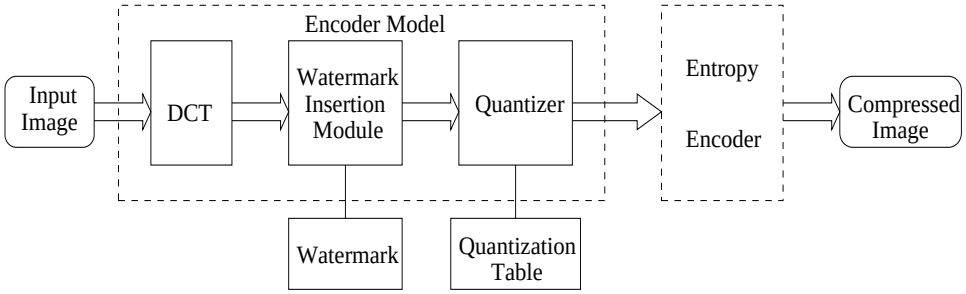
This chapter is organized as follows. We first discuss design and implementation of spatial domain invisible-robust and invisible-fragile watermarking chip. Followed by a design and implementation of a chip that can insert one or two of visible watermarks in an image in spatial domain. Finally, a DCT domain visible and invisible-robust watermarking chip has been discussed.

9.1 Invisible Watermarking in Spatial Domain

In this section, we propose a VLSI architecture [176] that can insert both invisible-robust and invisible-fragile watermarks in spatial domain. Depending on the user's requirement, it can insert



(a) Spatial Domain Watermark



(b) DCT Domain Watermark

Figure 9.1. Secure JPEG Encoder : Block Level View [176]

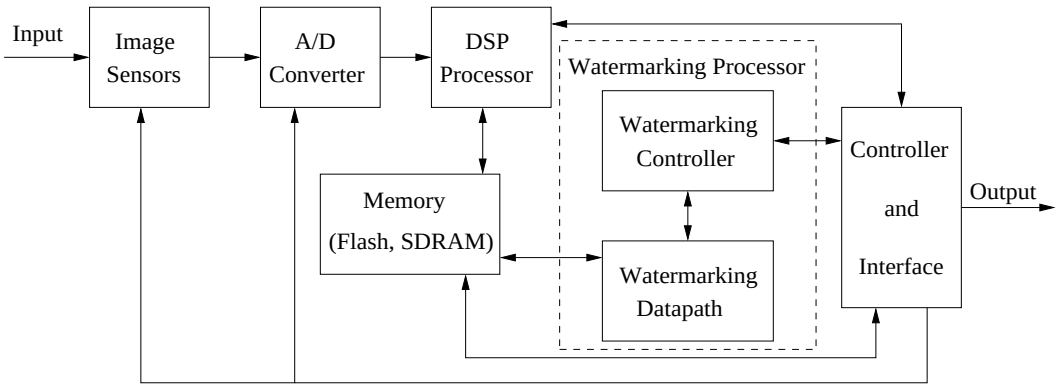


Figure 9.2. Secure Digital Still Camera : Schematic View

either of the watermarks or both. The following watermarking insertion algorithms are implemented : (i) the invisible-robust algorithm from [177, 178] and (ii) the invisible-fragile algorithm proposed by the authors from [83, 72]. Both the algorithms are quite different and are proposed recently.

9.1.1 Spatial Domain Invisible Watermarking Algorithms

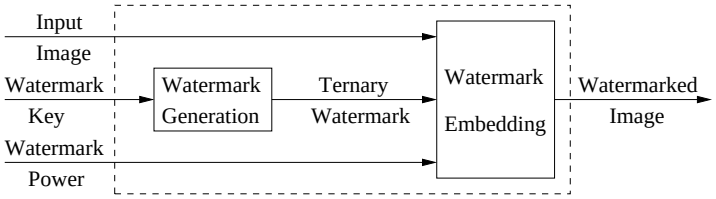
In this section, we describe the algorithms (invisible-robust and invisible-fragile) chosen for VLSI implementation. We outline the insertion and detection methods in brief with the modifications necessary to facilitate the hardware implementation. The notations needed for stating the algorithms are given in Table 9.1.

Table 9.1. Notations used to Explain Spatial Domain Watermarking Algorithms

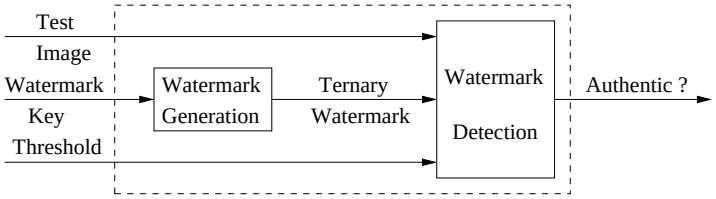
I	: Original image (gray image)
W	: Watermark image (binary or ternary image)
(i, j)	: A pixel location
I_W	: Watermarked image
$N_I \times N_I$	: Image dimension
$N_W \times N_W$	: Watermark dimension
E, E_1, E_2	: Watermark embedding functions
D	: Watermark detection function
r	: Neighborhood radius
I_N	: Neighborhood image (gray image)
K	: Digital (watermark) key
α_1, α_2	: Scaling constants (watermark strength)

9.1.1.1 Invisible Robust Algorithm

A block diagram of the watermark insertion scheme is shown in Fig. 9.3(a) [177, 178]. The watermark W is a ternary image having pixel values $\{0,1 \text{ or } 2\}$. These values are generated using the digital key K. The watermark insertion is performed by altering the pixels of original image as



(a) Watermark Insertion



(b) Watermark Detection

Figure 9.3. Invisible Robust Watermarking in Spatial Domain [177, 178]

follows.

$$I_W(i,j) = \begin{cases} I(i,j) & \text{if } W(i,j) = 0 \\ E_1\big(I(i,j), I_N(i,j)\big) & \text{if } W(i,j) = 1 \\ E_2\big(I(i,j), I_N(i,j)\big) & \text{if } W(i,j) = 2 \end{cases} \tag{9.1}$$

The encoding functions E_1 and E_2 are defined as follows, where $\alpha_1 > 0$ and $\alpha_2 > 0$.

$$\begin{aligned} E_1(I, I_N) &= (1-\alpha_1) I_N(i,j) + \alpha_1 I(i,j) \\ E_2(I, I_N) &= (1-\alpha_1) I_N(i,j) - \alpha_2 I(i,j) \end{aligned} \tag{9.2}$$

It may be noted that the above functions are slightly different from the original algorithm, where α_2 is negative and the second encoding function involved addition, instead of subtraction. However, these changes do not affect the overall encoding-decoding scheme, since we make changes in decoding functions accordingly.

The neighborhood image pixel gray value is calculated as the average gray value of the neighboring pixels of the original image for a particular neighborhood radius r. For example, for neigh-

borhood radius $r = 1$, it is calculated as :

$$I_N(i,j) = \frac{\frac{I(i+1,j)+I(i+1,j+1)}{2} + I(i,j+1)}{2} \tag{9.3}$$

The scaling $(1 - \alpha_1)$ is used to scale I_N to ensure that watermarked image gray value I_W never exceeds the maximum gray value for 8-bit image representation corresponding to pure white pixel. The neighborhood radius r determines the upper bound of the watermarked pixels in an image. It may be noted that a simple average could have been $\frac{I(i+1,j)+I(i+1,j+1)+I(i,j+1)}{3}$, but we used the above method of averaging to simplify the hardware implementation, since the division by two can be implemented using a right shift by 1-bit operation.

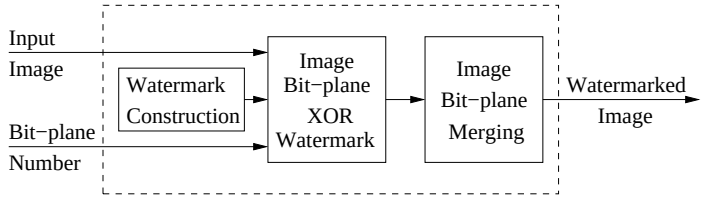
The block diagram for watermark detection is provided in Fig. 9.3(b). The first step detection process is the generation of watermark W using the watermark key K. Next, the watermark is extracted from the test (watermarked) image using the detection function given below.

$$W^*(i,j) = \begin{cases} 1 & \text{if } I_W(i,j) - I_N(i,j) > 0 \\ 2 & \text{if } I_W(i,j) - I_N(i,j) < 0 \end{cases} \tag{9.4}$$

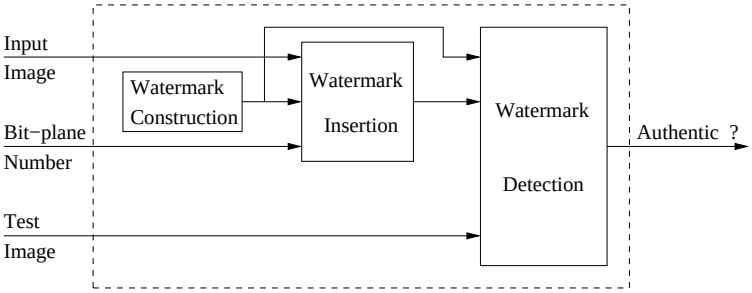
By comparing the original ternary watermark image W and the extracted binary watermark image W^*, the ownership can be established when the detection ratio is larger than a predefined threshold as explained in [177, 178].

9.1.1.2 Invisible Fragile Algorithm

The invisible fragile watermark insertion is carried out as follows (Fig. 9.4(a) [83, 72]). A pseudo-random binary-sequence $\{0,1\}$ of period N is generated using a linear shift register. The period N is equal to the number of pixels ($N_W \times N_W$) of the image. The watermark is generated by arranging the binary sequence into blocks of size 4×4 or 8×8. The size of the watermark is the same as the size of the image. The bit planes of the input image are derived and watermark is inserted in the appropriate bit plane such that $SNR >$ threshold. Assuming that the watermark insertion is to be performed in k^{th} bit plane, the watermark insertion process is given by the following



(a) Watermark Insertion



(b) Watermark Detection

Figure 9.4. Invisible Fragile Watermarking in Spatial Domain [83, 72]

expression.

$$\begin{aligned} I_W[0 \to k-1](i,j) &= I[0 \to k-1](i,j) \\ I_W[k](i,j) &= I[k](i,j) \text{XOR } W(i,j) \\ I_W[k+1 \to 7](i,j) &= I[k+1 \to 7](i,j) \end{aligned} \tag{9.5}$$

The finding of the candidate bit plane for watermark insertion is an iterative process. We have chosen the $2^{nd}(k=2)$ bit plane as the candidate for watermark insertion (for LSB $k=0$). After merging all the bit planes, the watermarked image I_W is obtained.

For image authentication purpose, the testing paradigm provided in [83, 72] is used. To construct the testing paradigm, the cross-correlations of the original image and the watermark image, and the cross-correlations of the watermarked image and the possibly forged test image are calculated. Then, based on the cross-correlations, the test statistics is determined. The test statistics is the basis of the test paradigm.

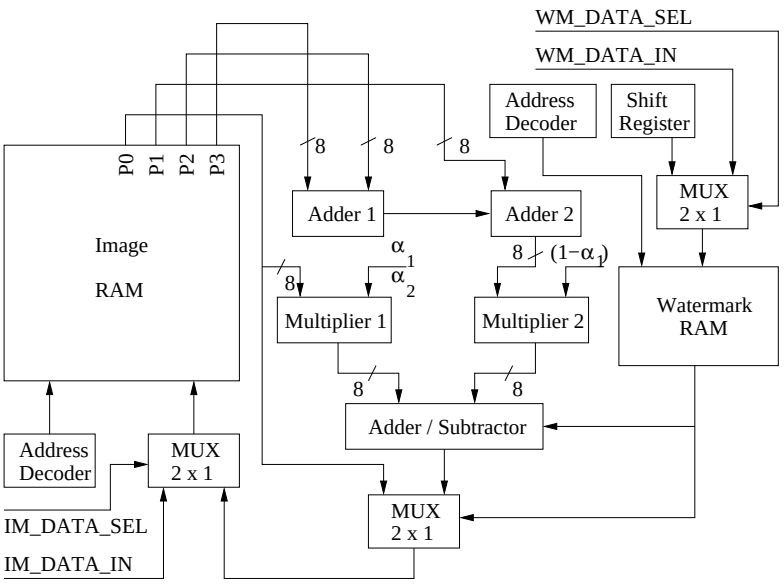


Figure 9.5. Datapath for Robust Watermarking

9.1.2 VLSI Architecture for Invisible Spatial Domain Watermarking

In this section, we discuss the proposed architectures for the algorithms discussed in the previous section.

9.1.2.1 Architecture for Robust Watermarking

The datapath for invisible robust watermarking is shown in Fig. 9.5. The image RAM is used to store the original image, which is to be watermarked. The image data can be written to the image RAM by activating proper control signals. The watermark RAM serves as a storage space for watermark data. The watermark data can either be generated using the shift register or given as an external input by the user. In this hardware design, it is assumed that at any point of time, a 256×256 image can be stored in the image RAM and a 128×128 watermark can be stored in the watermark RAM. It is possible to watermark only a 128×128 region of the original image at a time, whereas the full image can be watermarked if the process is repeated for the other regions (total in four times for the assumed size). The region of the original image to be watermarked is described in terms of five parameters, such as top_left, top_right, center, bottom_left, and bottom_right and address decoders are used to determine the proper locations.

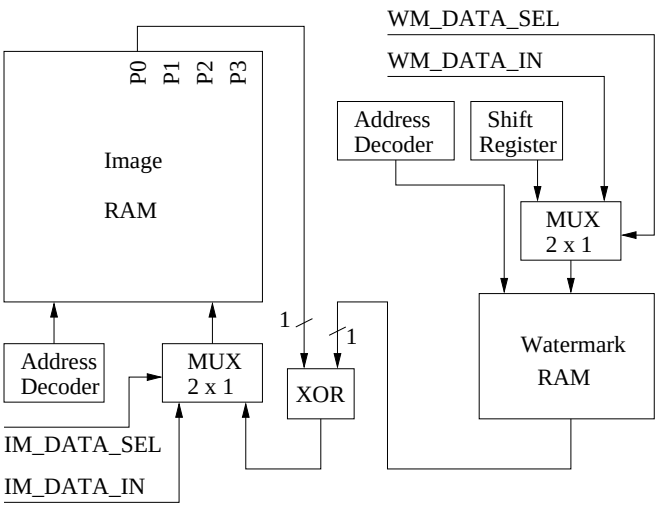


Figure 9.6. Datapath for Fragile Watermarking

The invisible robust watermark insertion scheme involves adding (or subtracting) a constant time the image pixel gray value to (from) a constant time of the neighborhood function. The constants are α_1 and α_2, the values of which determine the strength of the watermark. The four output lines from the image RAM provide the pixels $I(i,j)$, $I(i,j+1)$, $I(i+1,j)$ and $I(i+1,j+1)$ for the row-column address pair (i,j). The neighborhood function specified by Eqn. 9.3 is computed as follows. First, the $I(i,j+1)$ and $I(i+1,j+1)$ are given to the adder1 as input. The resulting sum and carry out from adder 1 are fed to adder 2 alongwith $I(i+1,j)$. The resulting sum of the adder 2 is the neighborhood function value. The division by two is performed by shifting the results bit right by one bit, consequently discarding the rightmost bit (LSB). The scaling of the neighborhood function is achieved by multiplying it with $(1-\alpha_1)$ using the multiplier 2. At the same time, the scaling of the image pixel gray values is performed in multiplier 1 by multiplying $I(i,j)$ with α_2 or α_1. The eight higher order bits of the the multipliers are fed to the adder/subtractor unit to perform watermark insertion as per the Eqn. 9.2. Since, we are concerned only with the integer values of the pixels, the lower eight bits of the multiplier results are discarded, which represent the values after the decimal point. The output of the adder / subtractor unit (watermarked image pixels) and the original image pixel values are multiplexed

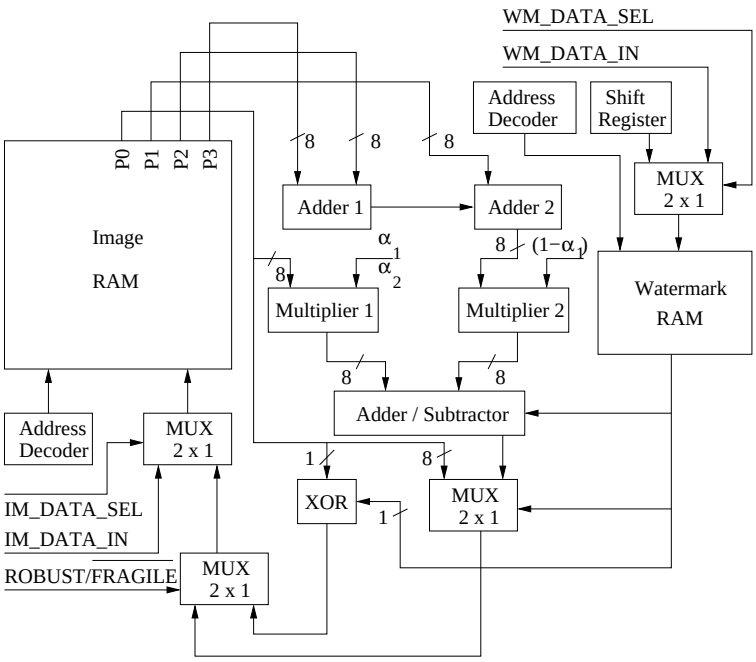


Figure 9.7. Datapath For Combined Spatial Domain Invisible Robust / Fragile Watermarking

based on the watermark values and are written into the image RAM if the watermark value is ”1” or ”2”, as per Eqn. 9.1.

9.1.2.2 Architecture for Fragile Watermarking

The datapath for fragile watermark insertion is shown in Fig. 9.6. The original image is stored in the image RAM and the watermark is created in the same way as in the case of robust watermarking described above and is stored in the watermark RAM. For watermark insertion, the 2^{nd} bit-line of the image pixels is fed as input to an XOR gate alongwith that of the watermark value. The output of the XOR gate is returned to the image RAM and the 2^{nd} bit-line is over-written by selecting appropriate control signals.

9.1.2.3 Overall Chip Architecture

The combined datapath for both robust and fragile watermarking is shown in Fig. 9.7. The datapath is obtained by stitching the two datapaths from (Fig. 9.5 and Fig. 9.6) using multiplexers, which in turn give rise to additional control signals. The controller that drives the datapath is

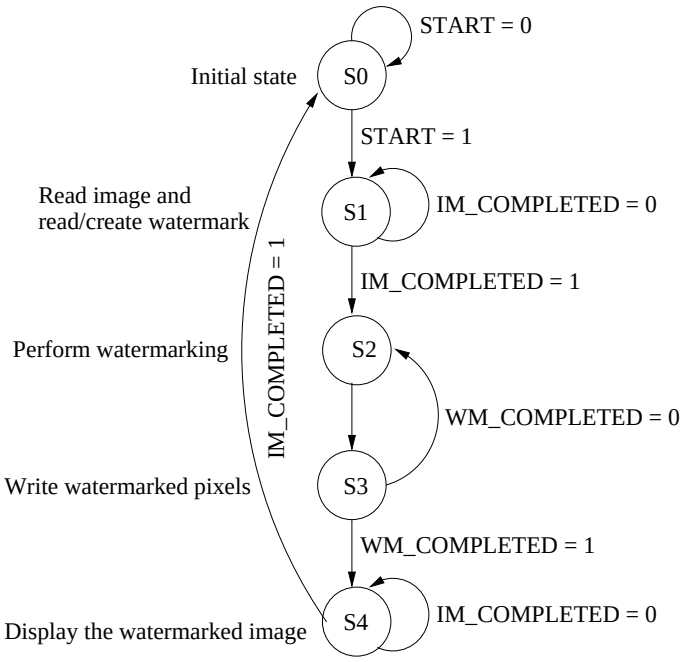


Figure 9.8. Controller For Combined Spatial Domain Invisible Robust / Fragile Watermarking

shown in Fig. 9.8. The controller has five states, such as S0, S1, S2, S3 and S4. The state S0 is the initial sate. In state S1, the image and watermark data are written into the respective RAMs. The image and the watermark pixels are read from the RAMs in state S2 and watermarking insertion is performed. In state S3, watermarked pixels are written back to the image RAM. In state S4, the watermarked image is ready in the RAM. The control signals and their functional descriptions are given in Table 9.2.

9.1.3 Implementation of Spatial Domain Invisible Watermarking Chip

In this section, we discuss the implementation of the integrated architecture which combines the two architectures from the previous section. The implementation of the watermarking datapath and controller was carried out in the physical domain using the Cadence Virtuoso layout tool using bottom-to-top hierarchical design approach. The design involved the construction of three main modules, the memory, the watermarking module (datapath) and the controller unit. Each of the three modules were designed individually through modularization and later interfaced with each other. The layouts of the gates at the lowest level of hierarachy are drawn using the CMOS standard

Table 9.2. Control Signals for Spatial Domain Invisible Watermarking Chip

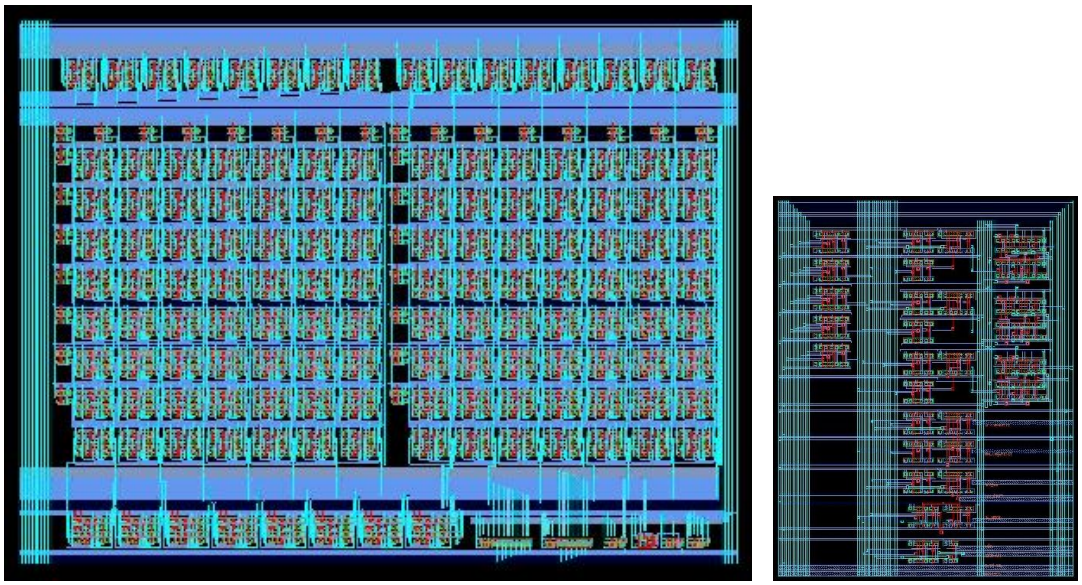
IM_ADDR_COUNT	: increment signal for the counters used to generate address for image
WM_ADDR_COUNT	: incre. signal for the counters used to generate address for watermark
IM_$\overline{\text{READ}}$/WRITE	: image RAM read (1) or write (0)
WM_$\overline{\text{READ}}$/WRITE	: watermark RAM read or write
IM_DATA_SELECT	: select input or watermarked image
WM_DATA_SELECT	: select input or generate watermark
IM_ADDR_SELECT	: select location of image
WM_ADDR_SELECT	: select address of watermark
START	: watermarking begins when set to 1
IM_COMPLETED	: set to 1 when all the pixels of the image are covered
WM_COMPLETED	: set to 1 when all the pixels in watermark are covered
BUSY	: high as long as the watermarking process continues
DATA_READY	: high when watermarked image is ready to be read
ROBUST/$\overline{\text{FRAGILE}}$	: choose between robust or fragile

cell design approach. We designed a standard cell library containing basic gates, such as AND, OR, NOT and 1-bit RAM cell.

The memory module involves two read/write memory structure, one for 256×256 size original/watermarked image and other for 128×128 size watermark. The bit size for the image RAM is $8-$bits and for the watermark RAM, it is $2-$bits. The basic building block for a memory module is a $6-$transistor static RAM cell available in the cell library. We have chosen a SRAM instead of a DRAM due to its shorter read and write cycles. The memories are built as $n \times n$ arrays of SRAM cells and are addressed using row and column address decoders. Each decoder is implemented as a $m-$bit counter with additional AND-logic to address 2^m cells.

The watermarking module (datapath) involves the implementation of two watermarking algorithms as described in Section 9.1.1. The main components of this module are two 8-bit adders, two 8-bit multipliers and a 8-bit adder/ subtractor. Each adder is constructed using 1-bit adders in a ripple-carry manner. The adder/subtractor unit is obtained from the adder using XOR gates. The carry inputs to the adder/ subtractor and one of the inputs to the XOR gate are set to high whenever the watermark pixel value is ”2” so that a subtraction is carried out as required for the robust watermarking encoding function (Eqn. 9.2). An 8-bit parallel array multiplier is built using full-adders and AND gates to implement multiplication operations with reduced delay.

Several multiplexers are used at appropriate places in the design to select one of the incoming lines. Each of such multiplexer is implemented using a combination of transmission gates. Three asynchronously resettable registers are designed to encode the five states of the controller depicted in Fig. 9.8. At anytime, the three registers could be reset by the user to return the controller to its intial state and from there, the watermarking function could be started afresh.



(a) Datapath Layout (b) Controller Layout

Figure 9.9. Layout of the Invisible Spatial Domain Watermarking Datapath and Controller

Table 9.3. Power, Area Details for Individual Units

Modules	Gate Count	Power (mW)	Delay (ns)
Datapath	4547	1.1931	0.9158
Controller	233	0.0045	0.3901
RAM	1183,744	21.8012	2.3891

Each of the above mentioned modules is implemented and tested separately and then connected together to obtain the final chip. The number of gates, power and areas of each module is shown in Table 9.3 for operating voltage of $3.3V$. The statistics are obtained using HSPICE for 0.35μ MOSIS SCN3M SCMOS technology. It is evident from the above statistics that the RAM consumes

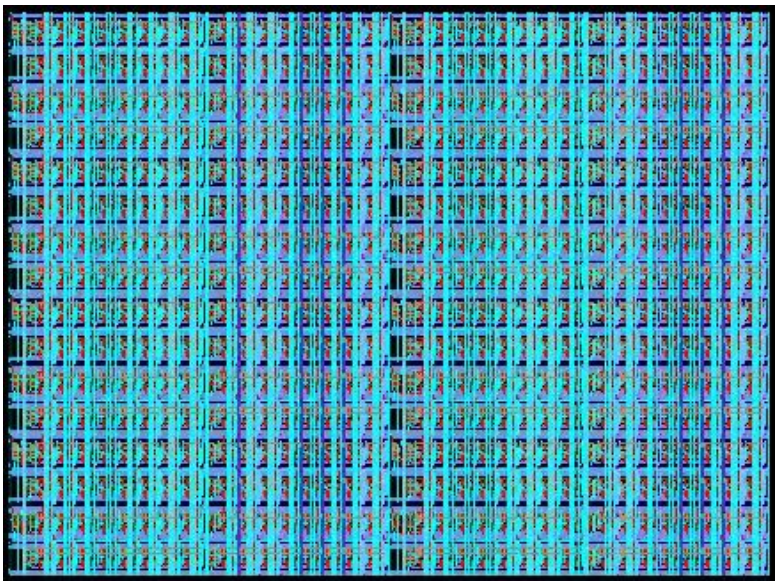


Figure 9.10. Layout of RAM (Zoomed view of a portion is shown)

most amount of power. If we assume that the proposed chip is to be used as a module within a complete JPEG enoder, then the memory module could be avoided in the watermarking datapath circuit. The layout of the datapath is shown in Fig. 9.9(a). and the layout of the controller is shown in Fig. 9.9(b). The layout of RAM is shown in Fig. 9.10. This shows a zoomed view of a small portion of the RAM. The complete layout and the floor plan of the watermarking chip is given in Fig. 9.11. The pin diagram for the chip showing the inputs and the outputs is given in Fig. 9.12. The overall design statistics of the chip are in Table 9.4.

Table 9.4. Overall Chip Statistics

Area (with RAM)	$15.012 \times 14.225 mm^2$
Number of gates (with RAM)	$1188K$
Number of gates (without RAM)	4820
Operating Voltage	$3.3V$
Clock frequency (with RAM)	$151MHz$
Clock frequency (without RAM)	$545MHz$
Number of I/O pins	25
Power (with RAM)	$24mW$
Power (without RAM)	$2.0547mW$

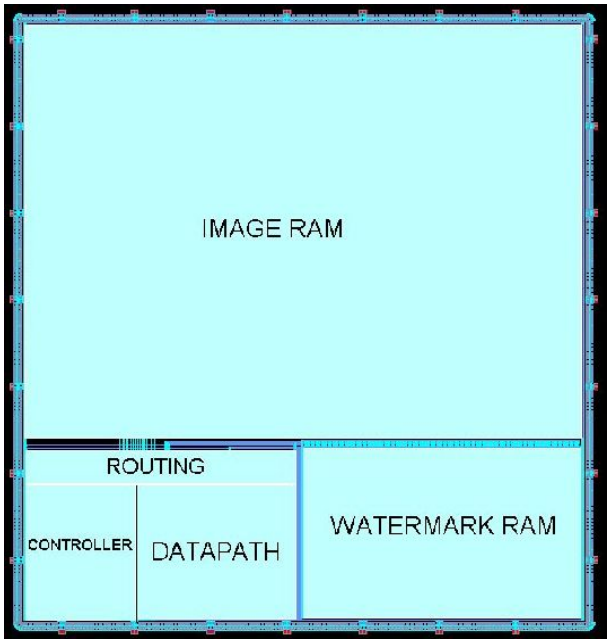


Figure 9.11. Layout of the Proposed Spatial Domain Invisible Watermarking Chip

9.1.4 Results and Conclusions

The verification of the chip implementation was performed by watermarking on several test images, examples of which are shown in Fig. 9.13 and Fig. 9.14. The visual inspection of the images illustrate the quality of the watermarking. As a quantitative measure of the perceptibility of the watermark, we used the expression for signal-to-noise ratio given in Eqn. 9.6 as suggested by

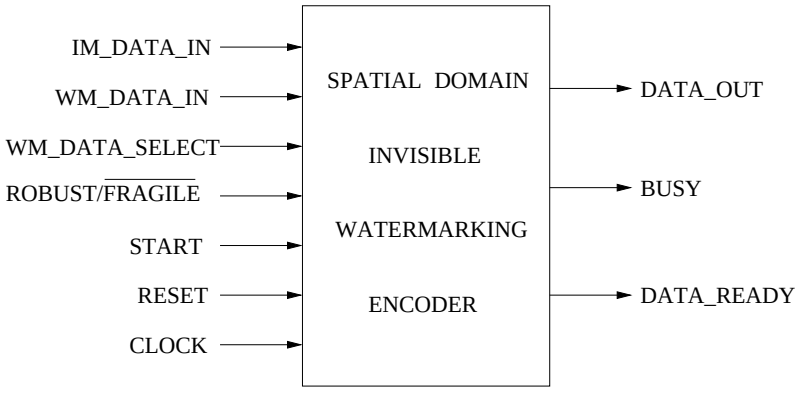


Figure 9.12. Pin Diagram for the Proposed Spatial Domain Invisible Watermarking Chip

(a) Original Shuttle (b) Robust Watermarked (c) Fragile Watermarked

Figure 9.13. Spatial Domain Invisible Watermarked Shuttle

(a) Original Bird (b) Robust Watermarked (c) Fragile Watermarked

Figure 9.14. Spatial Domain Invisible Watermarked Bird

[159, 83, 72].

$$SNR = 10\ log\left(\frac{\text{Var}_I}{\text{Var}_{I_E}}\right) \tag{9.6}$$

The Var_I is the variance of the original input image and the Var_{I_E} is the variance of the error image (difference between original input image and watermarked image). We calculated the SNR using the original and the watermarked image with the help of a software simulator. The SNR for various watermarked images were in the range of $23dB - 25dB$.

In this work, we presented a watermarking encoder that can perform invisible robust, invisible fragile watermarking and the combination of both in spatial domain. To our knowledge, this is the

first watermarking architecture having both functionalities. The chip can be easily integrated in any existing JPEG encoder to watermark the images right at the source end. The disadvantage of the watermarking algorithms implemented is that the processing needs to be done pixel by pixel. In future, we are aiming to investigate block by block processing. The implementation of a low power high performance watermarking decoder which will be a part of JPEG decoder is currently under implementation.

9.2 Visible Watermarking in Spatial Domain

In this section, we present a new VLSI architecture for two visible watermarking schemes presented in the literature. We implement the VLSI architecture using 0.35μ CMOS technology. The proposed watermarking chip is designed aiming at easy integration with any existing digital camera framework [179]. To our knowledge, this is the first watermarking chip implementing visible watermarking schemes.

9.2.1 Watermarking Algorithms

In this section, we discuss the image watermarking algorithms whose VLSI architecture is proposed. We outline the schemes in brief with the modifications necessary to facilitate the hardware implementations. The following notations are needed for description of the algorithms.

9.2.1.1 Visible Watermarking Algorithm 1 :

In this subsection, we discuss the visible watermarking algorithm proposed in [73]. The watermark has three goals, such as, (i) the visible watermark should identify the ownership, (ii) the visual quality of the host image should be preserved, (iii) the watermark should be difficult to remove from the host image. To satisfy these three conflicting criteria, schemes have been proposed for adding watermark with the orginal image. The watermarked image is obtained by adding a scaled gray value of the watermark image to the host image. The amount of scaling is done in such a way that the alternation of each original image pixel occurs to a perceptual equal degree. The

Table 9.5. List of Variables used in Algorithm Explanation

I	: Original (or host) image (a grayscale image)
W	: Watermark image (a grayscale image)
(m, n)	: A pixel location
I_W	: Watermarked image
$N_I \times N_I$	: Original image dimension
$N_W \times N_W$	: Watermark image dimension
i_k	: The k^{th} block of the original image I
w_k	: The k^{th} block of the watermark image W
$i_{W\,n}$	: The k^{th} block of the watermarked image I_W
α_k	: Scaling factor for k^{th} block (used for host image scaling)
β_k	: Embedding factor for k^{th} block (used for watermark image scaling)
μ_I	: Mean gray value of the original image I
$\mu_{I\,k}$	: Mean gray value of the original image block i_k
$\sigma_{I\,k}$	: Variance of the original image block i_k
α_{max}	: The maximum value of α_k
α_{min}	: The minimum value of α_k
β_{max}	: The maximum value of β_k
β_{min}	: The minimum value of β_k
I_{white}	: Gray value corresponding to pure white pixel
α_I	: A global scaling factor
C_1, C_2, C_3, C_4	: Linear regression co-efficients

original formulas have been simplifed to the following [75].

$$I_W(m,n) = \begin{cases} I(m,n) + W(m,n)\ \left(\frac{I_{white}}{38.667}\right)\ \left(\frac{I(m,n)}{I_{white}}\right)^{\frac{2}{3}} \alpha_I & \text{for } \frac{I(m,n)}{I_{white}} > 0.008856 \\ I(m,n) + W(m,n)\ \left(\frac{I(m,n)}{903.3}\right)\ \alpha_I & \text{for } \frac{I(m,n)}{I_{white}} \leq 0.008856 \end{cases} \tag{9.7}$$

The scaling factor α_I determines the strength of watermark.

Our aim is to implement the watermarking algorithms in a hardware. The above equation is simplified so that the hardware implementation becomes easier. At the same time, care is taken to make sure that the hardware is as accurate as the software implementations. We assume $I_{white} = 255$ and simplify the above equations to the following.

$$I_W(m,n) = \begin{cases} I(m,n) + \left(\frac{\alpha_I}{6.0976}\right)\ W(m,n)\ (I(m,n))^{\frac{2}{3}} & \text{for } I(m,n) > 2.2583 \\ I(m,n) + \left(\frac{\alpha_I}{903.3}\right)\ W(m,n)\ I(m,n) & \text{for } I(m,n) \leq 2.2583 \end{cases} \tag{9.8}$$

The above expression involves cubic root calculation, which could complicate the hardware implementation. So, we further simplify the above expressions and remove the cubic root function with a piecewise linear model. We divide the gray values range $[0,\, I_{white}]$ to four ranges, such as $\left[0,\, \frac{I_{white}}{4}\right]$, $\left[\frac{I_{white}}{4},\, \frac{I_{white}}{2}\right]$, $\left[\frac{I_{white}}{2},\, \frac{3I_{white}}{4}\right]$, and $\left[\frac{3I_{white}}{4},\, I_{white}\right]$. We fit four linear regression co-efficients that best approximates the cubic root in each of these ranges. Moreover, we roundup the fraction involved in the comparison operation and the final simplified expression that is implemented using hardware is as follows.

$$I_W(m,n) = \begin{cases} I(m,n) + \left(\frac{\alpha_I}{903.3}\right)\, W(m,n)\, I(m,n) & \text{for } I(m,n) \leq 2 \\ I(m,n) + \left(\frac{\alpha_I C_1}{6.0976}\right)\, W(m,n)\, I(m,n) & \text{for } 2 < I(m,n) \leq 64 \\ I(m,n) + \left(\frac{\alpha_I C_2}{6.0976}\right)\, W(m,n)\, I(m,n) & \text{for } 64 < I(m,n) \leq 128 \\ I(m,n) + \left(\frac{\alpha_I C_3}{6.0976}\right)\, W(m,n)\, I(m,n) & \text{for } 128 < I(m,n) \leq 192 \\ I(m,n) + \left(\frac{\alpha_I C_4}{6.0976}\right)\, W(m,n)\, I(m,n) & \text{for } 192 < I(m,n) < 256 \end{cases} \tag{9.9}$$

9.2.1.2 Visible Watermarking Algorithm 2 :

In this subsection, we discuss the visible watermarking algorithm proposed in [83]. The pixel gray values are modified based on local and global statistics. The watermaking insertion process consists of the following steps.

- Both host image (one to be watermarked) I and the watermark (image) W are divided into blocks of equal sizes (the two images may be of unequal size).
- Let i_k denote the k^{th} block of the original image I and w_k denote the k^{th} block of the watermark W. For each block (i_k), the local statistics; mean μ_{Ik} and variance σ_{Ik} are computed. The image mean gray value μ_I is also found out.
- The watermarked image block is obtained by modifying i_k as follows. Assuming that α_k and β_k are scaling and embedding factors respectively, depending on μ_{Ik} and σ_{Ik} of each host image block.

$$i_{W\,k} = \alpha_k\, i_k \,+\, \beta_k\, w_k \qquad k = 1, 2... \tag{9.10}$$

The choice of α_k and β_k are governed by certain characteristics of human visual system (HVS) and mathematical models are proposed so that the perceptual quality of the image are not degraded due to watermark addition. The α_k and β_k are obtained as follows.

- The α_k and β_k for edge blocks are taken to be α_{max} and β_{min} respectively.
- The α_k and β_k are found out using the following equations.

$$\begin{aligned} \alpha_k &= \frac{1}{\hat{\sigma_{I}}_k} exp\left(-(\hat{\mu_{I}}_k - \hat{\mu}_I)^2\right) \\ \beta_k &= \hat{\sigma_{I}}_k \left(1 - exp\left(-(\hat{\mu_{I}}_k - \hat{\mu}_I)^2\right)\right) \end{aligned} \tag{9.11}$$

  Where, $\hat{\mu_{I}}_k$ and $\hat{\mu}_I$ are normalised values of μ_{kI} and μ_I, and $\hat{\sigma_{I}}_k$ are normalised logarithm values of σ_{I}_k.

- The α_k and β_k are scaled to the ranges (α_{min},α_{max}) and (β_{min}, β_{max}) respectively, where α_{min} and α_{max} are minimum and maximum values of scaling factor, and β_{min} and β_{max} are minimum and maximum values of embedding factor. These parameters determine the extent of watermark insertion. A linear tranformation is used to scale current α_k and β_k values to the ranges (α_{min}, α_{max}) and (β_{min}, β_{max}), respectively. Let current values of α_k be written as α_k^c, and α_{min}^c and α_{max}^c, respectively denote the current minimum and maximum values. Similarly, let current values of β_k be written as β_k^c, and β_{min}^c and β_{max}^c, respectively denote the current minimum and maximum values. The α_k and β_k values are scaled as follows.

$$\begin{aligned} \alpha_k &= \left(\frac{\alpha_{max}-\alpha_{min}}{\alpha_{max}^c-\alpha_{min}^c}\right) \alpha_k^c + \left(\alpha_{max} - \left(\frac{\alpha_{max}-\alpha_{min}}{\alpha_{max}^c-\alpha_{min}^c}\right) \alpha_{max}^c\right) \\ \beta_k &= \left(\frac{\beta_{max}-\beta_{min}}{\beta_{max}^c-\beta_{min}^c}\right) \beta_k^c + \left(\beta_{max} - \left(\frac{\beta_{max}-\beta_{min}}{\beta_{max}^c-\beta_{min}^c}\right) \beta_{max}^c\right) \end{aligned} \tag{9.12}$$

We used first-order derivatives for edge detection. For horizontal edge detection, we compute the horizontal gradient as :

$$G_h(m,n) = I(m,n) - I(m+1,n) \tag{9.13}$$

The vertical gradient is computed as follows for vertical edge detection.

$$G_v(m,n) = I(m,n) - I(m,n+1) \tag{9.14}$$

The amplitude of an edge is calculated as,

$$G(m,n) = |G_h(m,n)| + |G_v(m,n)| \tag{9.15}$$

The mean amplitude for a block is computed as,

$$G_\mu = \frac{1}{N_B \times N_B} \sum_m \sum_n G(m,n) \tag{9.16}$$

When the mean amplitude for a block exceeds a predefined threshold, we declare it as an edge block. The values of m and n correspond to the pixel locations of individual blocks with reference to the original image pixel location.

The mean gray value of a block is calculated as the average of gray values of all pixels in the image block. The mean gray values are normalized with pure white pixel gray value. Thus, we have normalized mean gray values of a block as,

$$\hat{\mu_{I}}_k = \frac{1}{N_B \times N_B} \left(\frac{1}{I_{white}} \right) \sum_m \sum_n I(m,n) \tag{9.17}$$

Where, m and n are the pixel locations of the k^{th} image block; same as their locations in the original image. The normalized standard deviation of gray values for the k^{th} block is calculated as follows.

$$\hat{\sigma_{I}}_k = \frac{1}{N_B \times N_B} \left(\frac{2}{I_{white}} \right) \sum_m \sum_n \left| I(m,n) - \frac{I_{white}}{2} \right| \tag{9.18}$$

The exponential term in the Eqn. 9.11 is approximated as a power series. For $0 \leq x \leq 1$, we have the following Taylor series approximation which was used upto the square term in our implementation.

$$e^x = \sum_i \frac{x^i}{i!} = 1 + x + \frac{1}{2}x^2 + ... \tag{9.19}$$

In the step three of the insertion algorithm, scaling needs to be done using a linear transformation. The transformation needs to find the current minimum and maximum values for both α_k and β_k over all the blocks to perform the transformation. Due to this the hardware performance is going to be severely degraded since it has to wait till all the pixels of the images are covered to find local statistics of all the blocks. So, we modify the above Eqn. 9.11 to ensure that the performance of the hardware is improved with no compromise on the quality. We find α_k and β_k using the following equations.

$$\begin{aligned} \alpha_k &= \alpha_{min} + (\alpha_{max} - \alpha_{min}) \frac{1}{\hat{\sigma_I}_k} exp\left(-(\hat{\mu_I}_k - \hat{\mu}_I)^2\right) \\ \beta_k &= \beta_{min} + (\beta_{max} - \beta_{min}) \hat{\sigma_I}_k \left(1 - exp\left(-(\hat{\mu_I}_k - \hat{\mu}_I)^2\right)\right) \end{aligned} \tag{9.20}$$

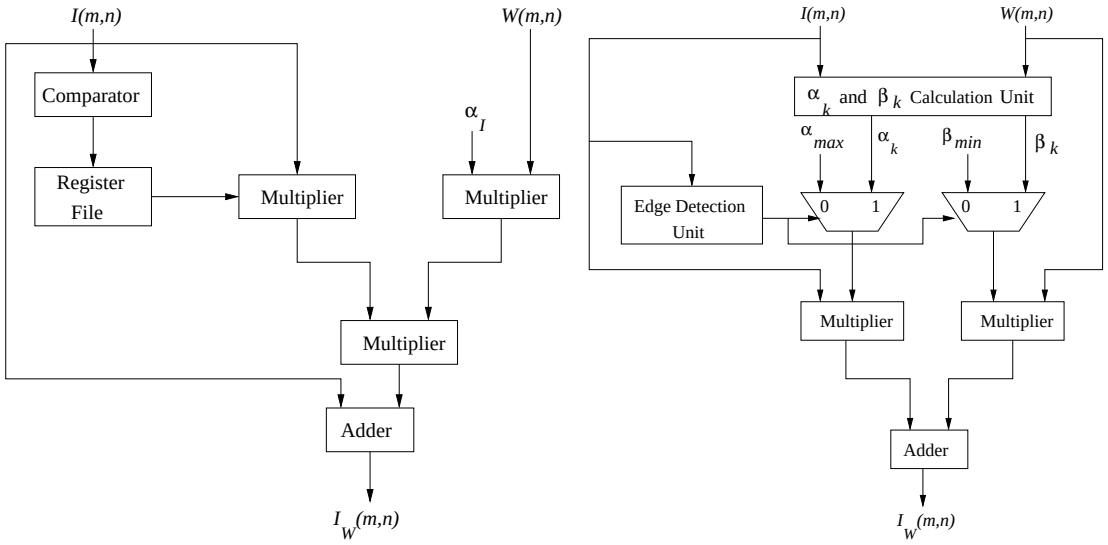
Extensive simulations for various images show that the α_k and β_k obtained using Eqn. 9.12 and Eqn. 9.20 are comparable (maximum difference is 5% [72]). Thus, we use Eqn. 9.20 for the α_k and β_k calculations.

9.2.2 VLSI Architecture

In this section, we discuss the architectures proposed for the hardware implementations of the algorithms described in Section 9.2.1. We discuss the implementation of the first algorithm and the architecture of the second algorithm in the first subsection and the second subsection respectively. The above two architectures are stitched to develop the proposed watermarking datapath. The FSM based design of a controller that drives the datapath is outlined. We assume that both the original host image and the watermark image are stored in some memory in the digital camera framework and are available for processing. The images may be in some compression format or may be available in raw ascii data. We need to have a corresponding decoder to decode the image and get the uncompressed data in case it is in compressed format. The decoder implementation is not a part of this research.

9.2.2.1 Architecture for Algorithm 1 :

The insertion operation for the first watermarking algorithm is described in Eqn. 9.7. This insertion function is simplified to Eqn. 9.9 using a piecewise linear model such that we have a



(a) For Algorithm 1

(b) For Algorithm 2

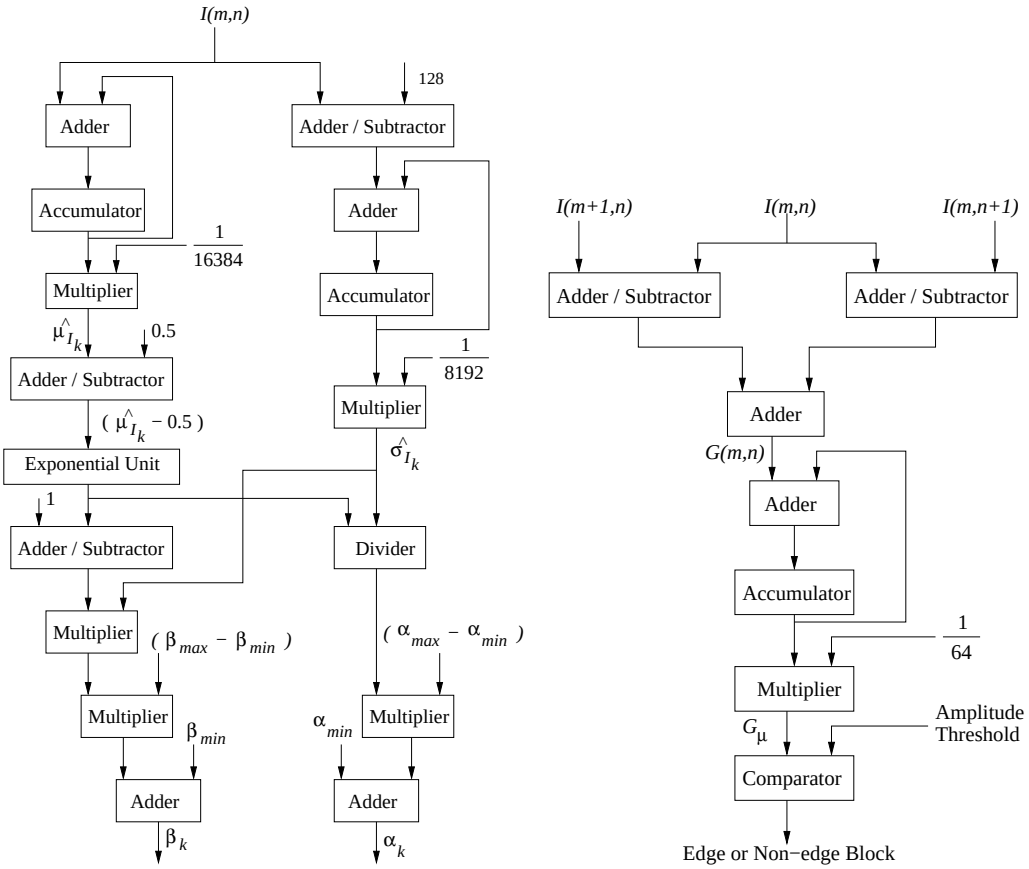
Figure 9.15. Datapath Architectures for the Visible Watermarking Algorithms

compact and efficient hardware design, as described in the previous section. Fig. 9.15(a) shows the architecture proposed for the first algorithm. The watermarking in this scheme is performed pixel-by-pixel as evident from the insertion function. A register file is used to store the constants needed to scale the image-watermark product in Eqn. 9.9. We store the constants $\frac{1}{903.3}$, $\frac{C_1}{6.0976}$, $\frac{C_2}{6.0976}$, $\frac{C_3}{6.0976}$, and $\frac{C_4}{6.0976}$. The other constant α_I is assumed as a parameter, which can be changed user to vary the watermark strength. The comparator is used to determine the range in which a particular pixel gray value lies, such that an appropriate constant can be picked up from the register file. The left side multiplier calculates appropriate constant times the host image pixel gray values and the right side multiplier is used to find α_I times the watermark image pixel gray value. The results of the above two multiplier is fed to the third multiplier which effectively calculates the product of constants, α_I, host image pixel gray value, and watermark image pixel gray value, respectively. The above product is added to the host image pixel gray values using the adder to obtain watermarked image pixel gray values. The above described process has to be carried out for all the pixels in order to obtain the watermarked image.

9.2.2.2 Architecture for Algorithm 2 :

The proposed architecture for the second algorithm is shown in Fig. 9.15(b). Using the second algorithm the watermarking insertion is performed block-by-block as described in Eqn. 9.10. But, for each block the watermarking insertion has to be carried out pixel-by-pixel. The proposed architecture in Fig. 9.15(b) present the operation at pixel level. The "α_k and β_k calculation unit" computes the α_k and β_k values for the k^{th} non-edge block using expression in Eqn. 9.20. The "edge detection unit" determines if a block is an edge block or non-edge block if the G_μ exceeds a user defined threshold, then it is an edge-block. Larger the threshold more are the blocks declared as edge-blocks. The multiplexors help in selecting the scaling and embedding factors between the edge and non-edge blocks. The left side multiplier calculates the scaling factors times the host image pixel gray value. The right side multiplier multiplies the embedding factor with the watermark image pixel gray value. The products from these two multipliers are added using an adder to find the watermarked image pixel gray value. This process is repeated for all pixels in a block, and subsequently for all the blocks in the image.

α_k and β_k calculation unit : The architectural details of "α_k and β_k calculation unit" is shown in Fig. 9.16(a). This hardware implements Eqn. 9.20 for α_k and β_k calculation for a block at a time. The left side adder-accumulator combination finds the sum of all the image pixel gray values for a block. After the sum is multiplied with $\left(\frac{1}{N_B \times N_B} * \frac{1}{I_{white}}\right)$, we get the normalised mean gray value of k^{th} block denoted by $\hat{\mu_{kI}}$. Since we have assumed block size of 8×8, and I_{white} as 256, this evaluates to $\frac{1}{16384}$. It may be noted that I_{white} is 255, but using 256 makes hardware implementation easier, the latter being representable as a power of two. In the original algorithm $(\hat{\mu_{kI}} - \hat{\mu}_I)$ is the deviation of a mean gray value of a block from the image mean gray value. We are evaluating the deviation of mean block gray value from mid-intensity of $\frac{I_{white}}{2}$ for simplicity, . Thus, $(\hat{\mu_{kI}} - \hat{\mu}_I)$ is computed as $(\hat{\mu_{kI}} - 0.5)$, when normalised with I_{white}. This assumption accelerates the hardware performance to a great extent since the block-by-block watermarking can be performed without waiting for the global image statistics computed over the whole image before the watermark insertion can be performed. The expression $exp\left(-(\hat{\mu_{kI}} - \hat{\mu}_I)^2\right)$ is computed using the "exponential unit".



(a) Architecture of α_k and β_k Calculation Unit (b) Architecture of Edge Detection Unit

Figure 9.16. Individual Datapath Units for Algorithm 2

The adder/subtractor unit finds the image pixel gray value absolute deviation from $\frac{I_{white}}{2}$. The adder-accumulator following this accumulate the $\sum_m \sum_n \left| I(m,n) - \frac{I_{white}}{2} \right|$ for a block. When this sum is multiplied with $\left(\frac{1}{N_B \times N_B}\right) * \left(\frac{2}{I_{white}}\right)$, which is 8192 for our case, we get the normalised standard deviation $\hat{\sigma_{Ik}}$. The right side divider divides exponential value computed before by $\hat{\sigma_{Ik}}$. The quotient is then multiplied with $\alpha_{max} - \alpha_{min}$. The above product is added to α_{min} to evaluate α_k expressed in Eqn. 9.20. The exponential unit result is fed to a adder/subtractor on left side which finds its difference from 1. The result is then multiplied with $\hat{\sigma_{Ik}}$ obtained from the computations performed before. The product obtained is then multiplied with $\beta_{max} - \beta_{min}$. This product is then added to β_{min} which in turn gives the required β_k as per Eqn. 9.20.

Edge detection unit : The architecture used to declare if a block is an edge or non-edge block is shown in Fig. 9.16(b). The left side and right side calculate the absolute value of horizontal gradient $|G_h(m,n)|$and absolute value of vertical gradient $|G_v(m,n)|$, respectively. The amplitude of an edge $G(m,n)$ is calculated using the first adder. Then, the adder-accumulator combination finds the sum of $G(m,n)$ for all pixels of a block. The above sum when multiplied with $\left(\frac{1}{N_B \times N_B}\right)$ $(= 64)$, we get the mean amplitude G_μ for a block. The comparator compares the G_μ values with an user defined threshold and declares the block as a edge or non-edge block.

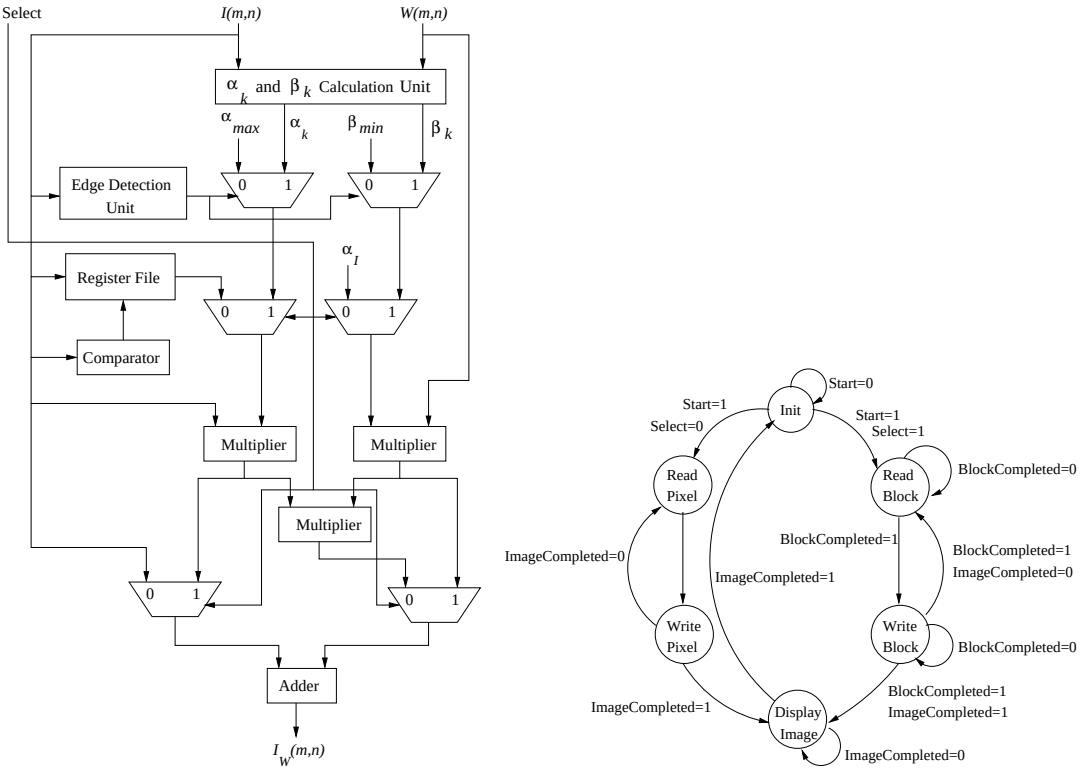
9.2.2.3 Architecture for the Watermarking Processor :

The datapaths for both the algorithms shown in Fig. 9.15(a) and Fig. 9.15(b) are stitched together using multiplexors and a combined datapath shown in Fig. 9.17(a) is obtained. This datapath can perform both the watermarking insertion schemes. Both the datapaths share the same multipliers, as it is evident from Fig. 9.17(a), the multiplexors help in selecting input for the multipliers. The ”Select” signal helps in choosing one of the watermarking scheme. When Select is ”0” first algorithm is used and when select is ”1”, second algorithm is performed.

The controller that drives the datapath is shown in Fig. 9.17(b). The controller has six states, such as Init, ReadBlock, WriteBlock, ReadPixel, WritePixel, and DisplayImage. When the Start signal is ”1” the watermarking process is initiated. Depending on the Select signal, one of the watermarking schemes is chosen and the corresponding datapath needs to be driven to carry out the watermarking process.

When Select is ”0”, first watermarking scheme is chosen. At the ReadPixel state a pixel is read and the watermarked pixel is written at the WritePixel state after watermarking is performed. The process continues as long as ImageCompleted is ”0” so that watermarking can be performed over all the pixels of the image.

The second algorithm is chosen when the Select is ”1”. In the ReadBlock state the pixel gray values are read for a block. The watermarked image block is written in the WriteBlock state once the watermarking is completed for the block. The system loops between the two states as long as all the blocks of the host image are not watermarked. Once, the watermarking is performed over



(a) Merged Datapath for Algorithms 1 and 2

(b) Controller for the Merged Datapath

Figure 9.17. Architecture for the Proposed Watermarking Processor

whole image, the ImageCompleted signal is set to ”1”; thus, completing the watermarking process. State DisplayImage is the state at which the watermark image is ready in the digital camera storage.

9.2.3 Chip Implementation

The implementation of the watermarking datapath and controller was carried out in the physical domain using the Cadence Virtuoso layout tool using bottom-to-top hierarchical design approach. The design involved the construction of four main units, such as the exponential unit, the edge detection unit, the α_k and β_k calculation unit, register file, and the accumulator. All of the above units have multipliers, adders, adder/subtractor, divider, comparator, and so on. These small functional units are laid out individually through modularization and later interfaced with each other to get the four above mentioned units. The datapath and the controller are constructed using the main units

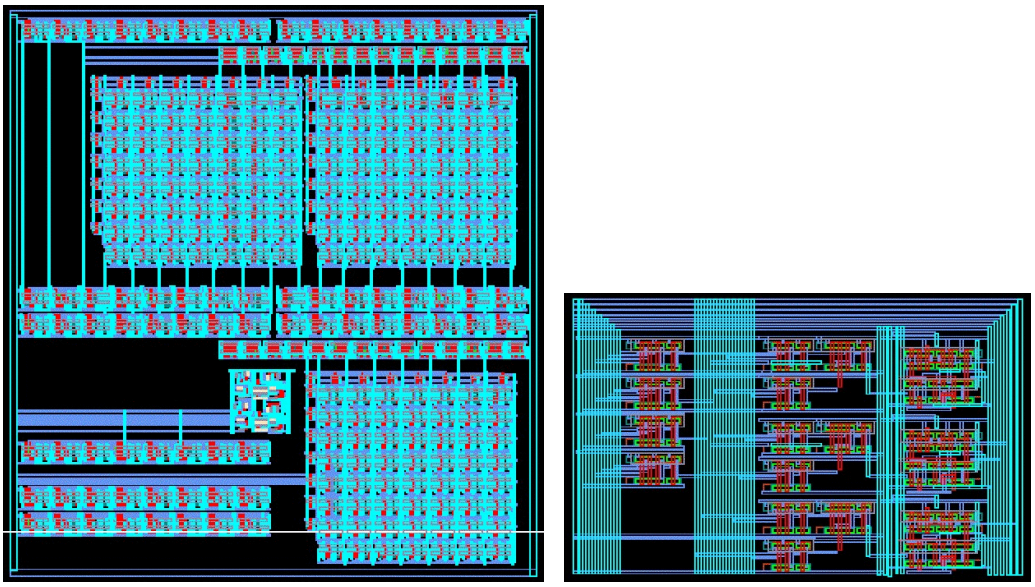
and the functional units. The layouts of the gates at the lowest level of hierarachy is drawn using the CMOS standard cell design approach. We designed our own standard cell library containing basic gates, such as AND, OR, NOT.

The datapath construction involves the implementation of the proposed architecture in the previous section. The fundamental functional units are 8-bit adders, 8-bit multipliers and 8-bit adder/subtractor. Each adder is constructed using 1-bit adders in a ripple-carry manner. The adder/subtractor unit is obtained from the adder using XOR gates [180]. The carry inputs to the adder/ subtractor and one of the inputs to the XOR gate are set to high whenever the select signal for this unit is ”2” so that a subtraction is carried out. The output of the adder/subtracter module gives the absolute value of the difference of two numbers when the difference is positive. When the difference is less than 0 (which is indicated by the carry bit taking a value 0), the absolute value is obtained by taking the 2’s complement of the output of the adder/subtractor module.

An 8-bit parallel array multiplier is obtained from full-adders and AND gates to implement multiplication operations with reduced delay [181]. The divider is implemented using the shift and subtract logic for the division [180]. The number to be divided is initially stored in two registers, A and Q, and with each subtraction, the values in A and Q are shifted left, with the most-significant bit in Q replacing the least-significant bit in A, and a 1 placed in the least-significant bit of Q. If the value in A is less than that of the divisor, the same shift procedure is repeated, except that a 0 is placed in the least-significant bit of Q. Finally, the quotient is available in the register Q, and the remainder in A.

The comparator was designed to compare the values of two 8-bit numbers for greater-than, equal to, or less-than relations. First, a single-bit comparator was designed to compare the values of two single-bit numbers, and later, instances of this module were cascaded to compare two 8-bit numbers, starting from the most-significant bit position and proceeding towards the least-significant bit position.

The accumulator is implemented as a 14-bit register to accommodate a maximum value of 64×256. The maximum value occurs when each pixel in a 8×8 block assumes the value of pure white pixel gray value. The register file is an addressable array of 8-bit registers (words) [181].

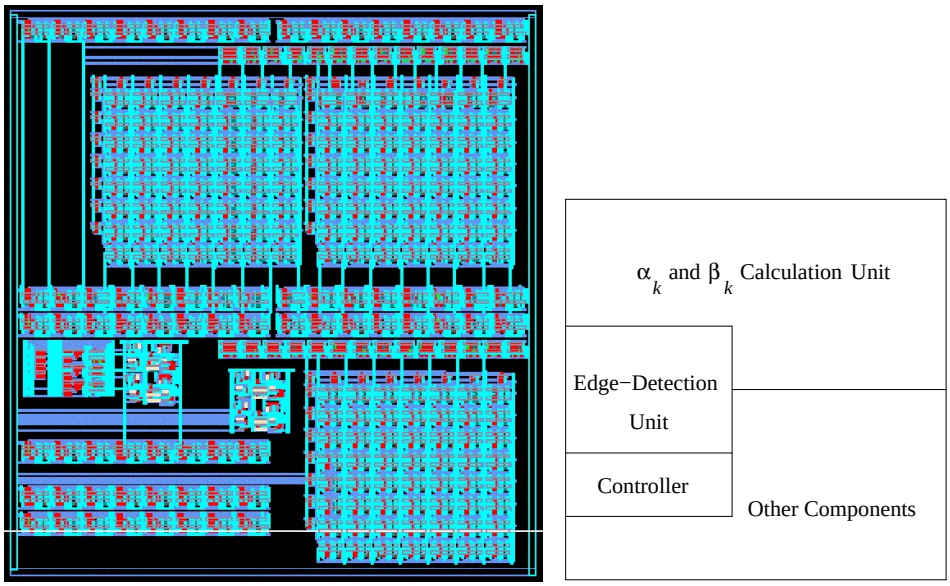


(a) Datapath (b) Controller

Figure 9.18. Layout of Datapath and Controller of the Proposed Chip

Based on the address specified and a Read/Write select line, at any time, a value can be either written to or read from the register file. Here, we used a 5-word register file to store the five different constants, such as $\frac{1}{903.3}$, $\frac{C_1}{6.0976}$, $\frac{C_2}{6.0976}$, $\frac{C_3}{6.0976}$, and $\frac{C_4}{6.0976}$, in Eqn. 9.9. Multiplexors are used at appropriate places in the design to select one of the incoming lines. Each of such multiplexor is implemented using a combination of transmission gates. Three asynchronously resettable registers are designed to encode the five states of the controller depicted in Fig. 9.17(b). The three registers could be reset by the user to return the controller to its intial state at any time and from there, the watermarking function could be started afresh.

Each of the above mentioned modules are implemented and tested separately and then connected together to obtain the final chip. The number of gates, power and areas of each module is shown in Table 9.2.3 for operating voltage of $3.3V$. The statistics are obtained using HSPICE for 0.35μ MOSIS SCN3M SCMOS technology. It is assumed that the proposed chip is to be used as a module in any existing JPEG encoder or a digital camera, and use their memory. The layout of



(a) Chip Layout (b) Chip Floor Plan

Figure 9.19. Layout and Floor Plan of the Proposed Watermarking Chip

the watermarking datapath is shown in Fig. 9.18(a). The layout of the controller is shown in Fig. 9.18(b).

Table 9.6. Power and Area of Different Units

Modules	Gate Count	Power (mW)	Delay (ns)
Exponential unit	2370	1.2314	0.8981
Edge detection unit	3599	1.4137	1.0967
α_k and β_k calculation unit	16279	3.444	2.0241
Controller	163	0.0034	0.3201

The complete layout of the watermarking chip is given in Fig. 9.19(a) and the floor plan of the chip is provided in Fig. 9.19(b). The clock frequency is driven by the critical delay of the watermarking module. Table 9.2.3 shows the overall design details of the chip and the corresponding pin diagram is shown in Fig. 9.20.

Table 9.7. Overall Statistics of the Watermarking Chip

Area	$3.34 \times 2.89 mm^2$
Number of gates	28469
Supply Voltage	$3.3V$
Clock frequency	$292.27 MHz$
Number of I/O pins	72
Power	$6.9286 mW$

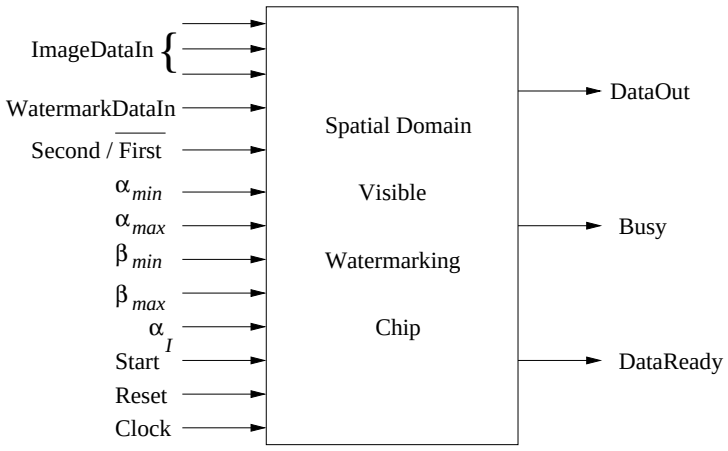


Figure 9.20. Pin Diagram for the Proposed Watermarking Chip

9.2.4 Results and Conclusions

Each of the functional units is simulated individually before being integrated together to develop the whole chip. The functional verification of the whole chip is done by performing watermarking on various test images. Fig. 9.21 shows various test images and the watermark image used, which are borrowed from [83, 74, 77, 72]. The test images as well as the watermark images are of dimension 256×256. The watermarked images obtained using the first algorithm is shown in Fig. 9.22. For this algorithm, the values of α_{min}, α_{max}, β_{min}, and β_{max} are assumed as $0.95, 0.98, 0.02$, and 0.07, respectively. Similarly, Fig. 9.23 shows the watermarked images obtained using the second algorithm, assuming α_I as 0.03. Using simulations, the regression coefficients, such as C_1, C_2, C_3, and C_4, are respectively found to be $0.339644, 0.21988, 0.185746$, and 0.172925.

(a) Lena

(b) Bird

(c) Nuts and Bolts

(d) Watermark

Figure 9.21. Original Host Images (a, b, and c) and Watermark Image (d)

A visual inspection of the watermarked images shows that the watermarking process is able to preserve the quality of the image while explicitly proving the ownership. Of the various quantitative measures available to quantify the quality of the watermarked images, we used signal-to-noise ratio (SNR) given in Eqn. 9.6. Software simulation results show that the SNR for various watermarked images is in the range of $20dB$ to $25dB$.

In this work, we have presented a watermarking chip that can be integrated within a digital camera framework for watermarking images. The watermarking chip can also be integrated in any existing JPEG encoder. The chip has two different types of watermarking capabilities, both in spatial domain. To our knowledge, this is the first watermarking chip having visible watermarking functionalities. Out of the two watermarking schemes implemented, the first one does pixel-by-pixel processing and the second one is a block-by-block processing algorithm. Additional work needs to be done to develop block-by-block operation for the first algorithm so that high perfor-

(a) Lena (b) Bird (c) Nuts and Bolts

Figure 9.22. Watermarked Images for the First Algorithm

(a) Lena (b) Bird (c) Nuts and Bolts

Figure 9.23. Watermarked Images for the Second Algorithm

mance hardware can be designed. However, both the algorithms are comparable from the SNR point of view.

9.3 Invisible and Visible Watermarking in DCT Domain

It is well known that the watermark can prove copyright and provide authenticity of the multimedia object. The watermarking can be performed on the multimedia object either in spatial, DCT or in wavelet domain. In the previous sections we described VLSI implementation of visible and invisible watermarking algorithms. In this era of portable electronic appliances the power consumption is a major issue. Thus, any VLSI chip will be commercialy viable f its power consumption is minimum. VLSI chips operating at multiple supply voltages are widely proposed as a

solution for low power optimization. Recently, the dynamic (or variable) frequency and multiple frequency have been proposed as techniques for low power design. In this work, we propose DCT domain low power wateramarking architectures using both multiple supply voltages and multiple supply frequency. The detailed architecture and the prototype chip implementation using TSMC 0.25μ technology are given in [85]. The prototype chip runs at a frequencies of $200MHz$ and $50MHz$ and voltages of $2.5V$ and $2.00V$.

9.3.1 Watermarking Algorithms

The spread spectrum invisible watermarking algorithm from [182, 183, 80] and the DCT domain visible watermarking algorithm from [74, 77, 72] are chosen for VLSI implementation. We used the following notations in our description.

9.3.1.1 Spread Spectrum Invisible Watermarking Insertion Algorithm

In [182, 183, 80], the watermark is inserted into the spectral components of the image using technique analogous to spread spectrum communication. The watermark is inserted judiciously in the perceptually significant components of a signal to make it robust to common signal distortions, geometric distortion, and malicious attacks, while maintaining perceptual quality of the image.

The insertion of watermark in the host image is as of follows. The DCT co-efficients are computed assuming the entire original image as one block. The 1000 largest of these co-efficients are identified as the perceptually significant for the image. The watermark $X = x_1, x_2, ..., x_{1000}$ is computed where each x_i is chosen according to $N(0,1)$, where $N(0,1)$ denotes a normal distribution with mean 0 and variance 1. The watermark is inserted in the DCT domain of the image by setting the frequency components in the original image using the following.

$$C_{I_W}(m,n) = C_I(m,n) * (1 + \alpha x_i) \tag{9.21}$$

The values of m and n corresponds to the pixels locations for 1000 largest DCT co-efficients, and $\alpha = 0.1$.

Table 9.8. Notations used in the Description of the Algorithm

I	: Original (or host) image (a grayscale image)
C_I	: DCT transformed original image
W	: Watermark image (a grayscale image)
C_W	: DCT transformed watermark image
(m, n)	: A pixel location
I_W	: Watermarked image
C_{I_W}	: DCT transformed watermarked image
$N_I \times N_I$	: Original image dimension (same as watermarked image dimension)
$N_W \times N_W$	: Watermark image dimension
$N_B \times N_B$	: Dimension of a block
$NOIB$	: Number of original image blocks $\left(\frac{N_I \times N_I}{N_B \times N_B}\right)$
$NOWB$	: Number of watermark image blocks $\left(\frac{N_W \times N_W}{N_B \times N_B}\right)$
c_{Ik}	: The k^{th} block of the DCT transformed original image C_I
c_{Wk}	: The k^{th} block of the DCT transformed watermark image C_W
$c_{I_W k}$	: The k^{th} block of the DCT transformed watermarked image C_{I_W}
α_k	: Scaling factor for k^{th} block (used for host image scaling)
β_k	: Embedding factor for k^{th} block (used for watermark image scaling)
$c_{Ik}(0,0)$	: DC-DCT co-efficient of the k^{th} block DCT block c_{Ik}
$c_{I\,max}(0,0)$	: Maximum of the DC-DCT co-efficients $(= Max\,(c_{Ik}(0,0))\ \forall\, k)$
μ_{DCI_k}	: Mean gray value of the original image block i_k, which is same as $c_{Ik}(0,0)$
μ_{DCI}	: Mean gray value of the original image I
$\mu_{DCI_{max}}$	: Maximum of mean gray value of any original image block $= Max\left(\mu_{DCI_k}\right)_{\forall k}$
$\mu_{DCI_{white}}$	: Mean gray value of any original image block with all white pixels
$\mu^*{}_{DCI_k}$	: Normalized μ_{DCI_k}
$\mu^*{}_{DCI}$	: Normalized μ_{DCI}
μ_{ACI_k}	: Mean of the AC-DCT co-efficients of the original image block i_k
σ_{ACI_k}	: Variance of the AC-DCT co-efficients of the original image block i_k
$\sigma_{ACI_{max}}$	: Maximum variance of AC-DCT co-efficients of any block $= Max\left(\sigma_{ACI_k}\right)_{\forall k}$
$\sigma_{ACI_{white}}$	: Variance of AC-DCT co-efficients of original image block with all white pixels
$\mu^*{}_{ACI_k}$	: Normalized μ_{ACI_k}
$\sigma^*{}_{ACI_k}$	: Normalized σ_{ACI_k}
α_{max}	: The maximum value of α_k
α_{min}	: The minimum value of α_k
β_{max}	: The maximum value of β_k
β_{min}	: The minimum value of β_k
α	: A scaling factor used for invisible watermark insertion
I_{white}	: Gray value corresponding to pure white pixel

9.3.1.2 Visible Watermarking Insertion Algorithm

The DCT domain visible watermarking algorithm proposed in [74, 77, 72] incorporates the human visual system (HVS) models to insert watermark adaptively. The insertion algorithm is as follows. The original image I (one to be watermarked) and the watermark image W are divided into blocks of size $N_B \times N_B$. The DCT co-efficient C_I for all the blocks of the original image are found out. For each block of the original image the mean gray value is computed as $\mu_{DCI_k} = c_{Ik}(0,0)$. The normalized mean gray values is calculated using the following equation.

$$\mu^*{}_{DCI_k} = \frac{\mu_{DCI_k}}{\mu_{DCI_{max}}} = \frac{c_{Ik}(0,0)}{c_{Imax}(0,0)} = \frac{c_{Ik}(0,0)}{Max(c_{Ik}(0,0))_{\forall k}} \tag{9.22}$$

Then the normalized mean gray value of the whole image is calculated as follows.

$$\mu^*{}_{DCI} = \frac{1}{NOIB}\sum_{k=0}^{NOIB-1}\mu^*{}_{DCI_k} = \frac{N_I \times N_I}{N_B \times N_B}\sum_{k=0}^{\frac{N_I \times N_I}{N_B \times N_B}-1}\mu^*{}_{DCI_k} \tag{9.23}$$

The mean and variance of AC DCT co-efficients of each block are calculated using the following equations.

$$\begin{aligned} \mu_{ACI_k} &= \frac{1}{N_B \times N_B}\sum_m\sum_n c_{Ik}(m,n) \\ \sigma_{ACI_k} &= \frac{1}{N_B \times N_B}\sum_m\sum_n \left(c_{Ik}(m,n) - \mu_{ACI_k}\right)^2 \end{aligned} \tag{9.24}$$

Where, the values of m and n corresponds to the locations of each pixel for each k^{th} block with reference to the pixel locations of the original image. The normalized variance of AC DCT co-efficients are computed as follows.

$$\sigma^*{}_{ACI_k} = \frac{\sigma_{ACI_k}}{\sigma_{ACI_{max}}} = \frac{\sigma_{ACI_k}}{Max\left(\sigma_{ACI_k}\right)_{\forall k}} \tag{9.25}$$

The scaling and embedding factor for each block are computed as below.

$$\begin{aligned} \alpha_k &= \sigma^*{}_{ACI_k}\ exp\left(-(\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^2\right) \\ \beta_k &= \frac{1}{\sigma^*{}_{ACI_k}}\ \left(1 - exp\left(-(\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^2\right)\right) \end{aligned} \tag{9.26}$$

The α_k and β_k are scaled to the range $(\alpha_{min}, \alpha_{max})$ and $(\alpha_{min}, \alpha_{max})$, respectively. The edge blocks are determined, and the α_k and β_k for edge blocks are taken to be α_{max} and β_{min}, respectively. The DCT co-efficient C_W for all the blocks of the watermark image are found out. The visible watermark is inserted in the host images block-by-block and watermarked image block is obtained. The number of blocks watermarked is $NOWB$, thus $k = 0 \rightarrow NOWB - 1$.

$$c_{I_W k} = \alpha_k \, c_{I k} + \beta_k \, c_{W k} \tag{9.27}$$

9.3.1.3 Algorithm Modification for Hardware Implementations

For invisible watermarking insertion in Eqn. 9.21 the three largest AC DCT co-efficients are considered as the candidates.

$$c_{I_W k}(m,n) = c_{I k}(m,n) + \alpha \, r_k(m,n) \quad (\text{where}, \quad k = 0 \rightarrow NOIB - 1) \tag{9.28}$$

Where, (m, n) corresponds to the three largest AC DCT values for k^{th} block. The random number matrix r_k is constructed from the random number $X = x_1, x_2, ...$

For visible watermarking algorithm the edge detection is an important step. The first step of edge detection involves summation of the absolute values of all AC DCT coefficients of each block as follows.

$$\left|\mu_{AC I_k}\right| = \frac{1}{N_B \times N_B} \sum_m \sum_n |c_{I k}(m,n)| \tag{9.29}$$

The maximum of the above values is $\left|\mu_{AC I_{max}}\right| = Max\left(\left|\mu_{AC I_k}\right|\right)$. A block is declared as an edge block if $\left|\mu_{AC I_k}\right| > \tau \left|\mu_{AC I_{max}}\right|$. The τ is a threshold constant; larger τ means lesser number of blocks declared as edge block.

In Eqn. 9.22 the normalization is performed using the $c_{I max}(0,0)$, the maximum of $c_{I k}(0,0)$. Finding $c_{I max}(0,0)$ out of $\frac{N_I \times N_I}{N_B \times N_B}$ values of $c_{I k}$s can slow down the insertion process. So, to improve the performance of the VLSI chip, we use $c_{I white}(0,0)$ for normalisation; $c_{I white}(0,0)$ is

the DC DCT of a block having all white pixels. Thus, the Eqn. 9.22 is modified to the following:

$$\mu^*{}_{DCI_k} = \frac{\mu_{DCI_k}}{\mu_{DCI_{white}}} = \frac{c_{Ik}(0,0)}{c_{I\,white}(0,0)} \tag{9.30}$$

We aim at improving the performance degradation due to normalization involved in Eqn. 9.25. now we aim at improving the performance degradation due to this step. Using 9.25 in Eqn. 9.26, we have the following equation.

$$\begin{aligned} \alpha_k &= \frac{\sigma_{ACI_k}}{\sigma_{ACI_{max}}}\ exp\left(-(\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^2\right) \\ \beta_k &= \frac{\sigma_{ACI_{max}}}{\sigma_{ACI_k}}\ \left(1 - exp\left(-(\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^2\right)\right) \end{aligned} \tag{9.31}$$

The factor $\sigma_{ACI_{max}}$ in Eqn. 9.31 serves as a constant scaling factor. Hence, we redefine the equations as follows.

$$\begin{aligned} \alpha_k^c &= \sigma_{ACI_k}\ exp\left(-(\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^2\right) \\ \beta_k^c &= \frac{1}{\sigma_{ACI_k}}\ \left(1 - exp\left(-(\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^2\right)\right) \end{aligned} \tag{9.32}$$

Where, the α_k^c and β_k^c values are current values of α_k and β_k, respectively. The above equations contain exponential (exp), which needs to be addressed. Eqn. 9.32 can be rewritten using Taylor series approximation upto the square term as follows.

$$\begin{aligned} \alpha_k^c &= \sigma_{ACI_k} * \left(1 - (\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^2 + (\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^4\right) \\ \beta_k^c &= \frac{1}{\sigma_{ACI_k}} * \left((\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^2 - (\mu^*{}_{DCI_k} - \mu^*{}_{DCI})^4\right) \end{aligned} \tag{9.33}$$

Now, the α_k^c and β_k^c are scaled to the range $(\alpha_{min}, \alpha_{max})$ and $(\alpha_{min}, \alpha_{max})$, respectively. The scaled α_k^c and β_k^c are respectively the α_ks and β_ks we are looking for.

9.3.2 VLSI Architecture

The overall architecture for the proposed DCT domain watermarking chip is shown in Fig. 9.24 which can insert both invisible and visible watermarks. This is a decentralized controller architec-

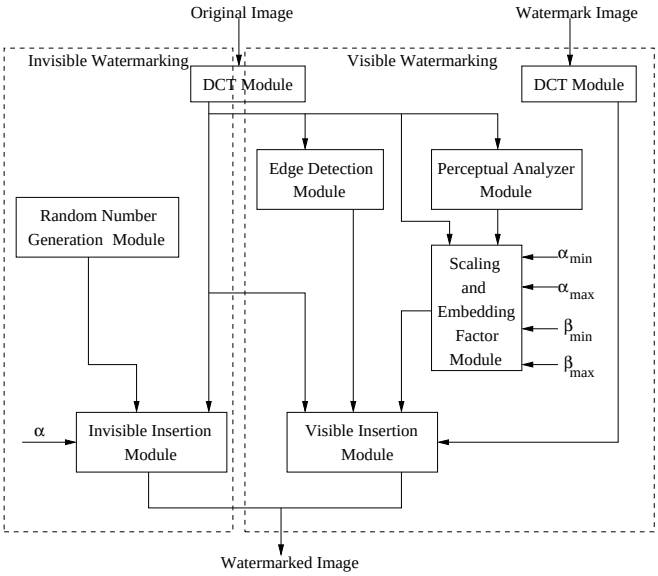
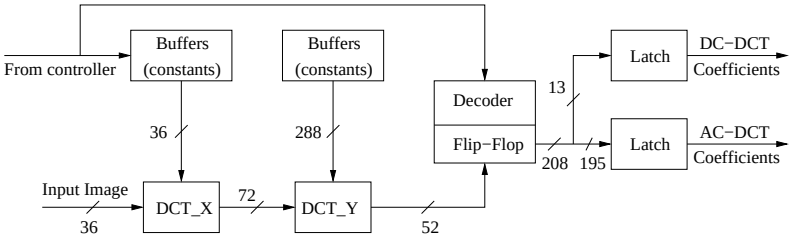


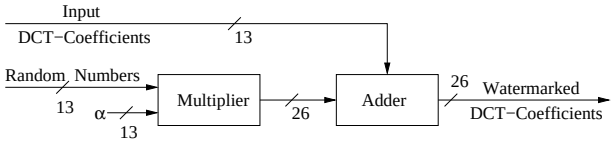
Figure 9.24. Combined Architecture for DCT domain Invisible and Visible Watermarking Chip

ture in which each module has its own controller. Here, we provide the proposed architecture in brief. The detailed architecture and the corresponding VLSI implementation are given in [85].

The modules used for invisible watermark insertion are DCT, random number generator, and invisible insertion (shown in Fig. 9.25). After the DCT co-efficients of the host image is calculated using DCT module, insertion module adds the random numbers to them. The α parameter is also given as input to the insertion module. The three appropriate AC DCT coefficients are chosen for watermark insertion using a counter. The DCT module is shown in Fig. 9.25(a). The DCT module consists of the following three sub-modules: (i) DCT_X, (ii) DCT_Y, and (iii) Controller. Apart from the above, flip-flops and latches are also used to store and forward the appropriate AC-DCT coefficients to the insertion module. The architecture of both the DCT_X and DCT_Y modules are borrowed from [184, 185]. Both DCT_X and DCT_Y use sixteen multipliers and twelve adders. All multipliers and adders pertain to IEEE 754 standard as implemented in IEEE.std_logic_arith package in VHDL [186]. The DCT_controller determines the coefficients to be forwarded, the memory addresses where the coefficients are to be stored, the time to trigger the invisible insertion module, and the random number generation module. The invisible watermark insertion module is shown in Fig. 9.25(b). The insertion module, which consists of a multipler and an adder, has its



(a) DCT Module



(b) Invisible Insertion Module

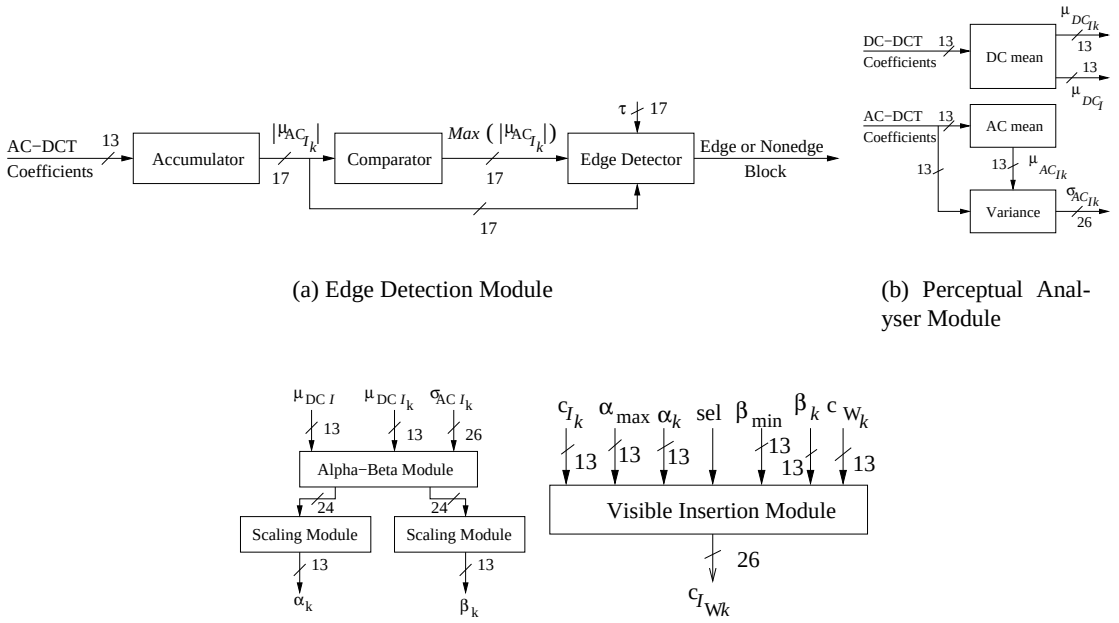
Figure 9.25. Architecture of the Different Units used for Invisible Watermarking

own controller. The insertion module scales the random number generated with α and adds it to the DCT coeffcient. The random number generation module consists of linear feedback shift registers (LFSR) [180].

The five modules involved in visible watermarking are as follows : (i) DCT module, (ii) Edge Detection module (iii) Perceptual Analyzer module, (iv) Scaling and Embedding Factor module, and (v) Visible Watermark Insertion module. Each of the above modules are discussed in detail below. The architecture of the DCT module is same as the one discussed in the previous section (Fig. 9.25(a)). The architecture of the rest are shown in Fig. 9.26.

The edge detection module determines the edge blocks in the original image. The threshold constant τ is given as input to the edge detection module. The three parts of the edge detection module implement a particular function, such as accumulation, comparison and detection needed for edge detection (refer Eqn. 9.29).

The perceptual analyzer module evaluates the Eqns. 9.22 and 9.25. Similar to the edge detection module, the perceptual analyzer module is also divided into three sub modules. The first sub module, namely the mean calculator computes the mean of the AC-DCT coefficients. The result of this sub-module is passed onto the next sub-module called the variance calculator module, which



(a) Edge Detection Module

(b) Perceptual Analyser Module

(c) Scaling and Embedding Factor Module

(d) Visible Insertion Module

Figure 9.26. Architecture of the Different Units used for Visible Watermarking

calculates the variance in the AC-DCT coefficients. The DC-DCT mean calculator is the third submodule of the perceptual analyzer. These submodules are implemented with adders, and feedback flip-flops, etc..

The scaling factor α_k and the embedding factor β_k are computed by the Scaling and Embedding factor module using Eqn. 9.33. This module is divided into two sub modules. The first module calculates the scaling factors and the embedding factors and is called the alpha-beta module. The second sub module scales down the scaling and embedding factors to a particular range depending on the user defined ranges $(\alpha_{min}, \alpha_{max})$ and $(\beta_{min}, \beta_{max})$.

The last module in this chip is the watermark insertion module. It serves the purpose of inserting the watermark into the original image. Using the information provided by the edge detection module and the scaling and embedding factor module, the watermark is inserted into the original image. It consists of two multipliers and an adder for evaluating the Eqn. 9.27 and has similar

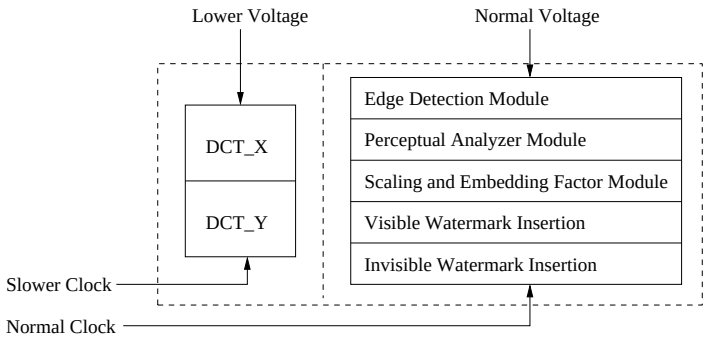


Figure 9.27. Dual Voltage and Dual Frequency Operation of the Datapath

architecture as that of invisible insertion module (in Fig. 9.25(b)). Multiplexors are used to select appropriate values of α_k and β_k for a non-edge blocks and an edge blocks.

The chip is to be operated with dual frequency dual voltage supplies (refer Fig. 9.27). Apart from the dual clock supplies, local clocks are automatically generated to trigger the operation of some modules. These local clocks are generated from the localized controllers embedded within each module. This type of clock generation within the chip helps to indirectly implement the clock gating technique. A low voltage supply is used for the DCT modules. The chip is implemented in such a way that the clock for the non-DCT modules must be an exact multiple of the clock for the DCT module. The DCT block processes 4 image pixels at a time. The other modules in the chip operate on one pixel at at time. Hence the DCT block can be clocked at one fourth the non-DCT clock frequency. The delay of the DCT module is less than its clock period. In this way there is a slack introduced in the DCT module which makes it possible to operate the DCT module at a lower voltage. The combination of low clock frequency and low voltage supply translates to lower power consumption by the DCT module.

A hierarchical design approach was adopted in implementing the chip. Standard cell design methodology was used for generating the layout. The standard cell design library used was obtained from [187], which is designed using TSMC 0.25μ CMOS technology. The standard cell library includes basic gates, flip flops, IO pads and corner cells. The layout for each module was generated and later integrated to obtain the final chip. The detailed implementation of the DCT domain

watermarking chip is discussed in [85]. The layout of the overall chip, floorplan of the chip and chip statistics are given.

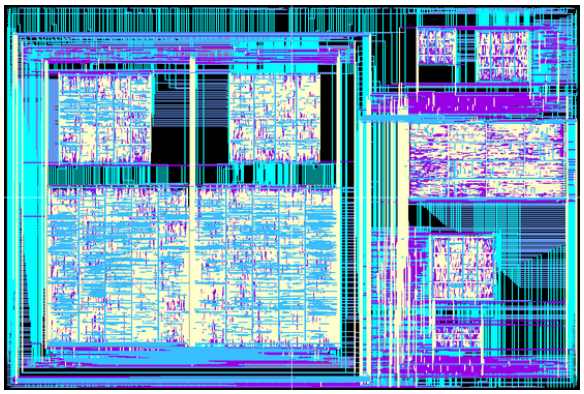


Figure 9.28. Layout of the DCT Domain Invisible and Visible Watermarking Chip [85]

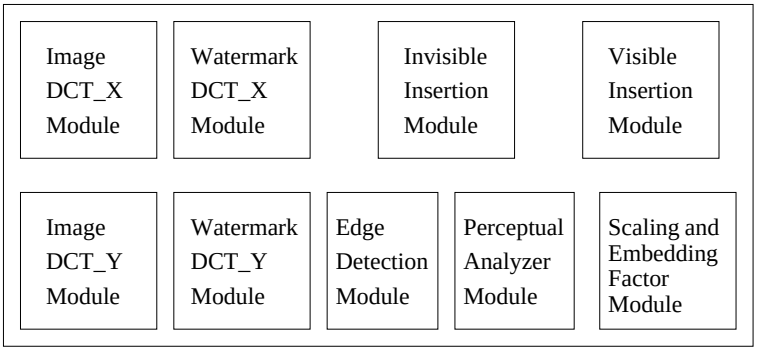


Figure 9.29. Floorplan of the DCT Domain Invisible and Visible Watermarking Chip [85]

Table 9.9. Overall Statistics of the DCT Domain Watermarking Chip [85]

Area	$4.0 \times 4.0mm^2$
Supply Voltages	$2.5V$ and $1.5V$
Operating Frequencies	$285MHz$ and $70MHz$
Power (Dual Voltage and Frequency)	$0.364mW$
Power (Normal Operation)	$1.95mW$

CHAPTER 10

CONCLUSIONS AND FUTURE WORK

The reduction of peak power, peak power differential, average power and energy are equally important. In this dissertation, we propose a framework for the reduction of these parameters through datapath scheduling at behavioral level. Several ILP based and heuristic based scheduling schemes are developed for datapath synthesis to minimize energy, energy delay product, peak power, simultaneous peak power and average power, simultaneous peak power, average power, peak power differential and energy, and power fluctuation. Three modes of circuit design, such as, single supply voltage and single frequency (SVSF), multiple supply voltages and dynamic frequency clocking (MVDFC), and multiple supply voltages and multicycling (MVMC) are considered. A new parameter called "Cycle Power Function" (CPF) is defined which captures the transient power characteristics as the equally weighted sum of normalized mean cycle power and normalized mean cycle differential power.

The ILP based schemes provide optimal solutions, however the growth of the problem complexity is exponential in terms of number of operations in the data flow graph. The alternate method is the heuristic based approach. The heuristics based algorithms provide polynomial time bound solutions for the scheduling problem. The reduction in energy and energy delay product was approximately the same for both heuristic and ILP-based methods. Similarly, the peak power (and average power) minimization was appreciably high for peak and average power minimization work. The significant results begin accomplished by cycle power function minimization works, which provided reduction in transient power and energy. Similarly, comparison of multicycling based works with dynamic frequency clocking based works reveal that dynamic frequency clocking based works out-perform in almost all instances.

None of the datapath scheduling algorithms available in current literature minimize transient power. There are few works available that handle peak power minimization. There are no research works handling both voltage and frequency parameters. Thus, we conclude any of the low power datapath scheduling algorithms proposed in this dissertation can create strong impact low power behavioral synthesis research.

The dissertation also involved design of visible and invisible watermarking chips both in spatial and DCT domains. The chips can be easlily integrated with any existing JPEG encoder or still digital camera. While the combined robust-fragile spatial domain invisible watermarking chip consumes $2.05mW$ power the spatial domain visible watermarking chip consumed $6.93mW$. The watermarked images produced by the watermarking chips are comparable with that obtained using the corresponding software implementations. The DCT domain watermarking chip is capable of inserting spread spectrum invisible watermark and an adaptive visible watermark. It operates at dual supply voltages and dual frequency mode. All the watermarking chip designed are the first implementatios in the respective category. At this digital age, when the copyright and piray are threat to indudtrial growths, the secure digital devices integrated with watermarking chips can produce copyrighted multimedia data in real-time.

The scheduling algorithms need to be extended to include pipelined datapaths in both dynamic frequency clocking and multicycling scenarios. The benchmarks used to test the scheduling schemes are data intensive digital signal processing benchmark circuits. The effectiveness of scheduling algorithms for control intensive applications needs to be investigated. Integer Linear Programming(ILP) based techniques for datapath scheduling are optimal, but cannot handle large benchmark circuits. Heuristic algorithms are fast, but generate sub optimal solutions should be used for scheduling of large benchmarks. The power model may be modified to consider the effect of exact switching activity and binding. More research is needed to develop low power dynamic clocking units for the generation of dynamic frequencies in VLSI circuits. The effect of dynamic clocking on the overall clock network has to be studied. Similarly, the design works can be extended to develop pipelined and / or SIMD based designs.

REFERENCES

[1] A. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–483, Apr 1992.

[2] Y. L. Lin, "Recent Developments in High-Level Synthesis," *ACM Transactions on Design Automation of Electronic Systems*, vol. 2, no. 1, pp. 2–21, Jan 1997.

[3] M. C. McFarland, Alice C. Parker, and Raul Camposano, "The High-Level Synthesis of Digital Systems," *Proceedings of the IEEE*, vol. 78, no. 2, pp. 301–318, Feb 1990.

[4] D. Gajski and N. Dutt, *High-Level Synthesis: Introduction to Chip and System Design*, Kluwer Academic Publishers, 1992.

[5] D. Singh, J. M. Rabaey, M. Pedram, F. Catthoor, S. Rajgopal, N. Sehgal, and T. J. Mozdzen, "Power Conscious CAD Tools and Methodologies: A Perspective," *Proceedings of the IEEE*, vol. 83, no. 4, pp. 570–594, Apr 1995.

[6] D. Sylvester and H. Kaul, "Power-Driven Challanges in Nanometer Design," *IEEE Design and Test of Computers*, vol. 13, no. 6, pp. 12–21, Nov-Dec 2001.

[7] D. Sylvester and H. Kaul, "Future Performance Challanges in Nanometer Design," in *Proceedings of the 38th Design Automation Conference*, June 2001, pp. 3–8.

[8] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez, "Reducing Power in High-Performance Microprocessors," in *Proceedings of the ACM / IEEE Design Automation Conference*, 1998, pp. 732–737.

[9] L. Benini, G. De Michelli, and A. Macii, "Designing Low-Power Circuits : Practical Recipes," *IEEE Circuits and Systems Magazine*, vol. 1, no. 1, pp. 6–25, March 2001.

[10] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23–29, July-Aug 1999.

[11] V. De and S. Borkar, "Technology and design challenges for low power and high performance [microprocessors]," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 1999, pp. 163–168.

[12] D. E. Lackey, P. S. Zuchowski, and J. Koehl, "Designing mega-ASICs in nanogate technologies," in *Proceedings of the Design Automation Conference*, 2003, pp. 770–775.

[13] E. Sicard and S. D. Bendhia, *Deep-submicron CMOS Circuit Design (simulator in Hands)*, Brooks/Coles, 2003.

[14] J. S. Lis and D. D. Gajski, "Synthesis from VHDL," in *Proceedings of the International Conference on Computer Design*, 1988, pp. 378–381.

[15] R. Composano and W. Wolf, *High-Level VLSI Synthesis*, Kluwer Academic Publishers, 1991.

[16] A. Raghunathan, N. K. Jha, and S. Dey, *High-Level Power Analysis and Optimization*, Kluwer Academic Publishers, 1998.

[17] M. Pedram, "Power Minimization in IC Design: Principles and Applications," *ACM Transactions on Design Automation of Electronic Systems*, vol. 1, no. 1, pp. 3–56, Jan. 1996.

[18] L. Benini and G. De Micheli, "System-Level Power Optimization: Techniques and Tools," *ACM Transactions on Design Automation of Electronic Systems*, vol. 5, no. 2, pp. 115–192, Apr 2000.

[19] J. M. Chang and M. Pedram, *Power Optimization and Synthesis at Behavioral and System Levels using Formal Methods*, Kluwer Academic Publishers, 1999.

[20] K. Roy and S. C. Prasad, *Low Power CMOS VLSI Circuits*, John Wiley and Sons, 2000.

[21] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, Inc., 1994.

[22] C. Park, *Task Scheduling in High Level Synthesis*, Ph.D. thesis, University of Illinois at Urbana-Champaign, 1996.

[23] A. C. Parker, J. Pizarro, and M. Mlinar, "MAHA : A Program for Datapath Synthesis," in *Proceedings of the 23rd ACM / IEEE Design Automation Conference*, June 1986, pp. 461–466.

[24] P. G. Paulin and J. P. Knight, "Force Directed Scheduling for the Behavioral Synthesis of ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 6, pp. 661–679, June 1989.

[25] S. Devadas and A. R. Newton, "Algorithms for Allocation in Datapath Synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 7, pp. 768–781, July 1989.

[26] P. G. Paulin and J. P. Knight, "Scheduling and Binding Algorithms for High-Level Synthesis," in *Proceedings of 26th ACM / IEEE Design Automation Conference*, June 1989, pp. 1–6.

[27] C. A. Papachristou and H. Konuk, "A Linear Program Driven Scheduling and Allocation Method," in *Proceedings of the 27th ACM/IEEE Design Automation Conference*, 1990, pp. 77–83.

[28] I. C. Park and C. M. Kyung, "Fast and Near Optimal Scheduling in Automatic Data Path Synthesis," in *Proceedings of the 28th Design Automation Conference*, 1991, pp. 680–685.

[29] R. Jain, A. Majumdar, A. Sharma, and H. Wang, "Empirical Evaluation of Some High-Level Synthesis Scheduling Heuristics," in *Proceedings of the 28th Design Automation Conference*, 1991, pp. 210–215.

[30] C. T. Hwang and J. H. Lee aand Y. C. Hsu, "A Formal Approach to the Scheduling Problem in High Level Synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 4, pp. 85–93, April 1991.

[31] R. A. Walker and S. Chaudhuri, "Introduction to the Scheduling Problems," *IEEE Design and Test of Computers*, vol. 12, no. 2, pp. 60–69, Summer 1995.

[32] S. Raje and M. Sarrafzadeh, "GEM : A Geometric Algorithm for Scheduling," in *Proceedings of the IEEE International Symposium on Circuits and Systems (Vol. 3)*, 1993, pp. 1991–1994.

[33] J. Zhu and D. D. Gajski, "Soft Scheduling in High Level Synthesis," in *Proceedings of the 36th Design Automation Conference*, 1994, pp. 219–224.

[34] M. J .M. Heijligers, L. J. M Cluitmans, and J. A. G. Jess, "High-level Synthesis Scheduling and Allocation using Genetic Algorithms," in *Proceedings of the 28th Design Automation Conference*, 1991, pp. 61–66.

[35] S. Haynal and F. Brewer, "Automata-Based Symbolic Scheduling for Looping DFGs," *IEEE Transactions on Computers*, vol. 50, no. 3, pp. 250–267, Mar 2001.

[36] R. Camposano, "Path-Based Scheduling for Synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 1, pp. 85–93, Jan 1991.

[37] P. G. Paulin and J. P. Knight, "Algorithms for High-Level Synthesis," *IEEE Design and Test of Computers*, vol. 6, no. 6, pp. 18–31, Dec 1999.

[38] E. Musoll and J. Cortadella, "Scheduling and Resource Binding for Low Power," in *Proceedings of the 8th International Symposium on System Synthesis*, 1995, pp. 104–109.

[39] H. J. M. Veendrick, "Short-Circuit Dissipation of Static CMOS Circuitry and its Impact on the Deisgn of Buffer Circuit," *IEEE Journal of Solid-State Circuits*, vol. 19, no. 4, pp. 468–473, Aug 1984.

[40] A. C. Williams, A. D. Brown, and M. Zwolinski, "Simultaneous Optimization of Dynamic Power, Area and Delay in Behavrioal Synthesis," *IEE Proceedings on Computer and Digital Techniques*, vol. 147, no. 6, pp. 383–390, Nov 2000.

[41] R. S. Martin and J. P. Knight, "Optimizing Power in ASIC Behavioral Synthesis," *IEEE Design and Test of Computers*, vol. 13, no. 2, pp. 58–70, Summer 1996.

[42] A. P. Chandrakasan and R.W. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits," *Proceedings of the IEEE*, vol. 83, no. 4, pp. 498–523, April 1996.

[43] H. S. Yun and J. Kim, "Power-Aware Modulo Scheduling for High-Performance VLIW Processors," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2001, pp. 40–45.

[44] R. S. Martin and J. P. Knight, "Using Spice and Behavioral Synthesis Tools to Optimize ASICs' Peak Power Consmpution," in *Proceedings of the 38th Midwest Symposium on Circuits and Systems*, 1996, pp. 1209–1212.

[45] S. P. Mohanty, N. Ranganathan, and S. K. Chappidi, "Peak Power Minimization Through Datapath Scheduling," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, Feb 2003, pp. 121–126.

[46] S. P. Mohanty, N. Ranganathan, and S. K. Chappidi, "Simultaneous Peak and Average Power Minimization During Datapath Scheduling for DSP Processors," in *Proceedings of the ACM Great Lakes Symposium on VLSI*, Apr 2003, pp. 215–220.

[47] V. Raghunathan, S. Ravi, A. Raghunathan, and G. Lakshminarayana, "Transient Power Management through High Level Synthesis," in *Proceedings of the International Conference on Computer Aided Design*, 2001, pp. 545–552.

[48] S. P. Mohanty and N. Ranganathan, "A Framework for Energy and Transient Power Reduction During Behavioral Synthesis," in *Proceedings of the International Conference on VLSI Design*, Jan 2003, pp. 539–545.

[49] L. Benini, G. Casterlli, A. Macii, and R. Scarsi, "Battery-Driven Dynamic Power Management," *IEEE Design and Test of Computers*, vol. 13, no. 2, pp. 53–60, Mar-Apr 2001.

[50] T. Burd and R. W. Brodersen, "Energy Efficient CMOS Microprocessor Design," in *Proceedings of the 28th Hawaii International Conference on System Sciences*, 1995, pp. 288–297.

[51] J. M. Chang and M. Pedram, "Energy Minimization using Multiple Supply Voltages," *IEEE Transactions on VLSI Systems*, vol. 5, no. 4, pp. 436–443, Dec 1997.

[52] J. Pouwelse, K. Langendoen, and H.Sips, "Energy Priority Scheduling for Variable Voltage Processor," in *Proceedings of the International Symposium on Low Power Electronics and Design*, Aug 2001, pp. 28–33.

[53] J. Rabaey, *Digital Integrated Circuits: A Design Perspective*, Prentice Hall, Inc., Upper Saddle River, NJ, 1996.

[54] S. P. Mohanty, N. Ranganathan, and V. Krishna, "Datapath Scheduling using Dynamic Frequency Clocking," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, Apr 2002, pp. 65–70.

[55] S. P. Mohanty and N. Ranganathan, "Energy Efficient Scheduling for Datapath Synthesis," in *Proceedings of the International Conference on VLSI Design*, Jan 2003, pp. 446–451.

[56] N. K. Jha, "Low Power System Scheduling and Synthesis," in *Proceedings of the International Conference on Computer-Aided Design*, 2001, pp. 259–263.

[57] T. L. Martin and D. P. Siewiorek, "Nonideal Battery and Main Memory Effects on CPU Speed-Setting for Low Power," *IEEE Transactions on VLSI Systems*, vol. 9, no. 1, pp. 29–34, Feb 2001.

[58] T. Pering, T. Burd, and R. W. Brodersen, "Voltage Scheduling in the lpARM Microprocessor System," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2000, pp. 96–101.

[59] N. Ranganathan, N. Vijaykrishnan, and N. Bhavanishankar, "A Linear Array Processor with Dynamic Frequency Clocking for Image Processing Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 4, pp. 435–445, August 1998.

[60] N. Ranganathan, N. Vijaykrishnan, and N. Bhavanishankar, "A VLSI Array Architecture with Dynamic Frequency Clocking," in *Proceedings of the International Conference on Computer Design*, 1996, pp. 137–140.

[61] I. Brynjolfson and Z. Zilic, "FPGA Clock Management for Low Power," in *Proceedings of the International Symposium on FPGAs*, 2000, pp. 219–219.

[62] I. Brynjolfson and Z. Zilic, "Dynamic Clock Management for Low Power Applications in FPGAs," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 2000, pp. 139–142.

[63] J. M. Kim and S. I. Chae, "New MPEG2 Decoder Architecture using Frequency Scaling," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1996, pp. 253–256.

[64] S. P. Mohanty, N. Rangnathan, and S. K. Chappidi, "An ILP-Based Scheduling Scheme for Energy Efficient High Performance Datapath Synthesis," in *Proceedings of the International Symposium on Circuits and Systems (Vol. 5)*, 2003, pp. 313–316.

[65] M. Johnson and K. Roy, "Datapath Scheduling with Multiple Supply Voltages and Level Converters," *ACM Transactions on Design Automation of Electronic Systems*, vol. 2, no. 3, pp. 227–248, July 1997.

[66] M. Igarashi, K. Usami, K. Nogami, F. Minami, Y. Kawasaki, T. Aoki, M. Takano, S. Sonoda, M. Ichida, and N. Hatanaka, "A low-power design method using multiple supply voltages," in *Proceedings of the International Symposium on Low Power Electronics and Design*, Aug 1997, pp. 18–20.

[67] M. Hamada, M. Takahashi, H. Arakida, A. Chiba abd T. Terazawa, T. Ishikawa, M. Kanazawa, M. Igarashi, K. Usami, and T Kuroda, "A Top-Down Low Power Design Technique Using Clusture Voltage Scaling with Variable Supply-Voltage Scheme," in *Proceedings of the 1998 IEEE Costum Integrated Circuits Conference*, 1998, pp. 495–498.

[68] K. Usami, M. Igarashi, F. Minami, T. Ishikawa, M. Kanzawa, M. Ichida, and K. Nogami, "Automated low-power technique exploiting multiple supply voltages applied to a media processor," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 3, pp. 463–472, Mar 1998.

[69] K. Usami, K. Nogami, M. Igarashi, F. Minami, Y. Kawasaki, T. Ishikawa, M. Kanzawa, T. Aoki, M. Takano, C. Mizuno, M. Ichida, S. Sonoda, M. Takahashi, and N. Hatanaka, "Automated low-power technique exploiting multiple supply voltages applied to a media processor," in *Proceedings of the IEEE 1997 Custom Integrated Circuits Conference*, May 1997, pp. 131–134.

[70] S. Katzenbeisser and F. A. P. Petitcolas, *Information Hiding techniques for steganography and digital watermarking*, Artech House, Inc., MA, USA, 2000.

[71] N. Memon and P. W. Wong, "Protecting Digital Media Content," *Communications of the ACM*, vol. 41, no. 7, pp. 34–43, Jul 1998.

[72] S. P. Mohanty, "Watermarking of Digital Images," M.S. thesis, Indian Institute of Science, Bangalore, India, 1999.

[73] G. W. Braudaway, K. A. Magerlein, and F. Mintzer, "Protecting Publicly Available Images with a Visible Image Watermark," in *Proceedings of the SPIE Conference on Optical Security and Counterfiet Deterrence Technique (Vol. SPIE-2659)*, 1996, pp. 126–132.

[74] S. P. Mohanty, K. R. Ramakrishnan, and M. S. Kankanhalli, "A DCT Domain Visible Watermarking Technique for Images," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2000, pp. 1029–1032.

[75] J. Meng and S. F. Chang, "Embedding Visible Video Watermarks in the Compressed Domain," in *Proceedings of the International Conference on Image Processing (Vol. 1)*, 1998, pp. 474–477.

[76] Y. Hu and S. Kwong, "Wavelet Domain Adaptive Visible Watermarking," *IEE Electronics Letters*, vol. 37, no. 20, pp. 1219–1220, Sep 2001.

[77] S. P. Mohanty, K. R. Ramakrishnan, and M. S. Kankanhalli, "An Adaptive DCT Domain Visible Watermarking Technique for Protection of Publicly Available Images," in *Proceedings of the International Conference on Multimedia Processing and Systems*, 2000, pp. 195–198.

[78] P. Wayner, *Disappearing Cryptography, Information Hiding : Steganography and Watermarking*, Morgan Kaufmann, CA, USA, 2002.

[79] M. Kankanahalli, Rajmohan, and K. R. Ramakrishnan, "Content Based Watermarking for Images," in *Proceedings of the 6th ACM International Multimedia Conference*, 1998, pp. 61–70.

[80] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia," *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1673–1687, Dec 1997.

[81] W. Zhu, Z. Xiong, and Y. Q. Zhang, "Multiresolution Watermarking for Images and Video," *IEEE Transanctions on Circuits and Systems for Video Technology*, vol. 9, no. 4, pp. 545–550, June 1999.

[82] R. G. Wolfgang and E. J. Delp, "A Watermark for Digital Images," in *Proceedings of the IEEE International Conference on Image Processing (Vol. 3)*, 1996, pp. 219–222.

[83] S. P. Mohanty, K. R. Ramakrishnan, and M. S. Kankanhalli, "A Dual Watermarking Technique for Images," in *Proceedings of the 7th ACM International Multimedia Conference (Vol. 2)*, 1999, pp. 49–51.

[84] J. Fridrich and M. Goljan, "Images with Self-Correcting Capabilties," in *Proceedings of the International Conference on Image Processing (Vol. 3)*, 1999, pp. 792–796.

[85] K. Balakrishnan, "A Dual Voltage and Dual Frequency Low Power VLSI Implementation of DCT Domain Image Watermarking Schemes," M.S. thesis, University of South Florida, Fall, 2003.

[86] M. Johnson and K. Roy, "Optimal Selection of Supply Voltages and Level Conversions during Datapath Scheduing under Resource Constraints," in *Proceedings of the International Conference on Computer Design*, Oct 1996, pp. 72–77.

[87] M. Johnson and K. Roy, "Scheduling and Optimal Voltage Selection for Low Power Multiple-Voltage DSP Datapaths," in *Proceedings of the IEEE Symposium on Circuits and Systems (Vol. 3)*, June 1997, pp. 2152–2155.

[88] J. M. Chang and M. Pedram, "Energy Minimization Using Multiple Supply Voltages," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 1996, pp. 157–162.

[89] Y. R. Lin, C. T. Hwang, and A. C. H. Wu, "Scheduling Techniques for Variable Voltage Low Power Design," *ACM Transactions on Design Automation of Electronic Systems*, vol. 2, no. 2, pp. 81–97, Apr 1997.

[90] M. Sarrafzadeh and S. Raje, "Scheduling with Multiple Voltages under Resource Constraints," in *Proceedings of the IEEE Symposium on Circuits and Systems (Vol. 1)*, 1999, pp. 350–353.

[91] A. Kumar and M. Bayoumi, "Multiple Voltage-Based Scheduling Methodology for Low Power in the High Level Synthesis," in *Proceedings of the International Symposium on Circuits and Systems (Vol. 1)*. July, July 1999, pp. 371–379.

[92] A. Kumar and M. Bayoumi, "A novel scheduling-based CAD methodology for exploring the design space of ASICs for low power," in *Proceedings of the 11th Annual IEEE International ASIC Conference*, Sep 1998, pp. 115–118.

[93] A. Kumar and M. Bayoumi, "A novel scheduling-based CAD methodology for exploring the design space of ASICs for low power," in *Proceedings of the 1998 IEEE Asia-Pacific Conference on Circuits and Systems*, Nov 1998, pp. 391–394.

[94] M. A. Elgamel and M. Bayoumi, "On low-power high-level synthesis using genetic algorithms," in *Proceedings of the 9th International Conference on Electronics, Circuits and Systems (Vol. 2)*, Nov 2002, pp. 725–728.

[95] W. T. Shiue and C. Chakrabarti, "Low-Power Scheduling with Resources Operating at Multiple Voltages," *IEEE Transactions on Circuits and Systems-II : Analog and Digital Signal Processing*, vol. 47, no. 6, pp. 536–543, June 2000.

[96] W. T. Shiue and C. Chakrabarti, "Low power scheduling with resources operating at multiple voltages," in *Proceedings of the 9th International Symposium on Circuits and Systems (Vol. 2)*, June 1998, pp. 437–440.

[97] A. Manzak and C. Chakrabarti, "A Low Power Scheduling Scheme with Resources Operating at Multiple Voltages," *IEEE Transactions on VLSI Systems*, vol. 10, no. 1, pp. 6–14, Feb 2002.

[98] A. Manzak and C. Chakrabarti, "A Low Power Scheduling Scheme with Resources Operating at Multiple Voltages," in *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems (Vol. 1)*, July 1999, pp. 354–357.

[99] N. Kumar, S. Katkoori, L. Rader, and R. Vemuri, "Profile-driven Behavioral Synthesis for Low Power VLSI System," *IEEE Design and Test of Computers*, vol. 12, no. 3, pp. 70–84, Fall 1995.

[100] S. Katkoori, N. Kumar, and L. Rader and; R. Vemuri, "A profile driven approach for low power synthesis," in *Proceedings of the International Conference on Asian and South Pacific Design Automation Conference (ASP-DAC)*, 1995, pp. 759–765.

[101] A. Raghunathan and N. K. Jha, "SCALP: An Iterative-Improvement Based Low-Power Datapath Synthesis System," *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 16, no. 11, pp. 1260–1277, Nov 1997.

[102] A. Raghunathan and N. Jha, "Behavioral Synthesis for Low Power," in *Proceedings of the International Conference on Computer Design*, 1994, pp. 318–322.

[103] S. Bhatia and N. K. Jha, "Behavioral Synthesis for Hierarchical Testability of Controller / Datapath Circuit with Conditional Branches," in *Proceedings of the International Conference on Computer Design*, Oct. 1994.

[104] L. Y. Chiou, K. Muhammand, and K. Roy, "DSP data path synthesis for low-power applications," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (Vol2)*, 2001, pp. 1165–1168.

[105] K. S. Khouri, G. Lakshminarayana, and N. K. Jha, "High-level synthesis of low-power control-flow intensive circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 12, pp. 1715–1729, Dec 1999.

[106] R. Henning and C. Chakrabarti, "An approach to switching activity consideration during high-level, low-power design space exploration," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 5, pp. 339–351, May 2002.

[107] R. Henning and C. Chakrabarti, "Activity models for use in low power, high-level synthesis," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (Vol. 4)*, Mar 1999, pp. 1881–1884.

[108] W. T. Shiue and C. Chakrabarti, "ILP Based Scheme for Low Power Scheduling and Resource Binding," in *Proceedings of the IEEE International Symposium on Circuits and Systems (Vol. 3)*, 2000, pp. 279–282.

[109] M. Lundberg, K. Muhammad, K. Roy, and S. K. Wilson, "High-level modeling of switching activity with application to low-power DSP system synthesis," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (Vol.4)*, Mar 1999, pp. 1877–1880.

[110] M. Lundberg, K. Muhammad, K. Roy, and S. K. Wilson, "A novel approach to high-level switching activity modeling with applications to low-power DSP system synthesi," *IEEE Transactions on Signal Processing*, vol. 49, no. 12, pp. 3157–3167, Dec 2001.

[111] M. K. Shin and C. H. Lin, "An efficient resource allocation algorithm with minimal power consumption," in *Proceedings of the IEEE Region 10 International Conference on Electrical and Electronic Technology (Vol. 2)*, 2001, pp. 703–706.

[112] J. Rabaey, C. Chu, P. Hoang, and M. Potkonjak, "Fast Prototyping of Datapath-Intensive Architectures," *IEEE Design and Test of Computer*, vol. 8, no. 2, pp. 40–51, June 1991.

[113] J. Monteiro, S. Devadas, P. Ashar, and A. Mauskar, "Scheduling Techniques to Enable Power Management," in *Proceedings of the ACM / IEEE Design Automation Conference*, 1996, pp. 349–352.

[114] R. V. Cherabuddi and M. A. Bayoumi, "A low power based partitioning and binding technique for single chip application specific DSP architectures," in *Proceedings of the Second Annual IEEE International Conference on Innovative Systems in Silicon*, Oct 1997, pp. 350–361.

[115] J. S. Lee, H. D. Lee, C. W. Park, and S.-Y. Hwang, "Power-conscious scheduling algorithm for performance-driven datapath synthesis," *IEE Electronics Letters*, vol. 32, no. 17, pp. 1574–1576, Aug 1996.

[116] S. Gupta and S. Katkoori, "Force-directed scheduling for dynamic power optimization," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, 2002, pp. 68–73.

[117] A. Murugavel and N. Ranganathan, "A Game Theoritic Approach for Binding in Behavioral Synthesis," in *Proceedings of the International Conference on VLSI Design*, Jan 2003, pp. 452–458.

[118] R. V. Cherabuddi, M. A. Bayoumi, and H. Krishnamurthy, "A low power based system partitioning and binding technique for multi-chip module architectures," in *Proceedings of the 7th Great Lakes Symposium on VLSI*, Mar 1997, pp. 156–162.

[119] W. T. Shiue, "High Level Synthesis for Peak Power Minimization using ILP," in *Proceedings of the IEEE International Conference on Application Specific Systems, Architectures and Processors*, 2000, pp. 103–112.

[120] W. T. Shiue, "Low Power VLSI Design : Peak Power Minimization using Novel Scheduling Algorithm Based on an ILP Model," in *Proceedings of the 10th NASA Symposium on VLSI Design*, Mar 2002.

[121] W. T. Shiue, J. Denison, and A. Horak, "A Novel Scheduler for Low Power Real Time Systems," in *Proceedings of the 43rd Midwest Symposium on Circuits and Systems*, Aug 2000, pp. 312–315.

[122] J. Pouwelse, K. Langendoen, and H.Sips, "Dynamic Voltage Scaling on a Low-Power Microprocessor," in *Proceedings of the 7th International Conference on Mobile Computing Network*, July 2001.

[123] T. Ishihara and H. Yasura, "Voltage Scheduling Problem for Dynamic Variable Voltage Processors," in *Proceedings of the International Symposium on Low Power Electronics and Design*, Aug 1998, pp. 197–202.

[124] T. Okuma, T. Ishihara, and H. Yasuura, "Real-time task scheduling for a variable voltage processor," in *Proceedings of the 12th International Symposium on System Synthesis*, Nov 1999, pp. 24–29.

[125] T. Okuma, H. Yasuura, and T. Ishihara, "Software energy reduction techniques for variable-voltage processors," *IEEE Design and Test of Computers*, vol. 18, no. 2, pp. 31–41, Mar-Apr 2001.

[126] I. Hong, M. Potkonjak, and M. B. Srivastava, "On-line scheduling of hard real-time tasks on variable voltage processor," in *Proceedings of the IEEE / ACM International Conference on Computer-Aided Design*, Nov 1998, pp. 653–656.

[127] I. Hong, D. Kirovaski, G. Qu, M. Potkonjak, and M. B. Srivastava, "Power optimization of variable-voltage core-based systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 12, pp. 1702–1714, Dec 1999.

[128] M. M. Mansour, M. M. Mansour, I. Hajj, and N. Shanbhag, "Instruction Scheduling for Low Power on Dynamically Variable Voltage Processors," in *Proceedings of the 7th IEEE International Conference on Electronics, Circuits and Systems*, 2000, pp. 613–618.

[129] A. Azevedo, I. Issenin, R. Cornea, R. Gupta, N. Dutt, A. Veidenbaum, and A. Nicolau, "Profile-based dynamic voltage scheduling using program checkpoint," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, 2002, pp. 168–175.

[130] A. Azevedo, R. Cornea, I. Issenin R. Gupta, N. Dutt, A. Nicolau, and A. Veidenbaum, "Architectural and compiler strategies for dynamic power management in the COPPER project ," in *Proceedings of the International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems*, 2001, pp. 25 –34.

[131] V. Swaminathan and K. Chakrabarty, "Investigating the effect of voltage-switching on low-energy task scheduling in hard real-time systems," in *Proceedings of the Asia and South Pacific Design Automation Conference*, 2001, pp. 251–254.

[132] V. Swaminathan and K. Chakrabarty, "Pruning-based energy-optimal device scheduling for hard real-time system," in *Proceedings of the Tenth International Symposium on Hardware / Software Codesign*, 2002, pp. 175–180.

[133] C. H. Hsu, U. Kremer, and M. Hsiao, "Compiler-Directed Dynamic Frequency and Voltage Scheduling," in *Proceedings of the Workshop on Power-Aware Computer Systems*, Nov 2000, pp. 65–81.

[134] C. H. Hsu, U. Kremer, and M. Hsiao, "Compiler-Directed Dynamic Voltage/Frequency Scheduling for Energy Reduction in Microprocessors," Tech. Rep., Departament of Computer Science, Rutgers University, 2001.

[135] Y. H. Lee and C. M. Krishna, "Voltage-Clock Scaling for Low Energy Consumption in Real-Time Embedded Systems," in *Proceedings of the 6th International Conference on Real-Time Computing Systems and Applications*, 1999, pp. 272–279.

[136] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, Oct 1995, pp. 374–382.

[137] J. Luo and N. K. Jha, "Power-profile driven variable voltage scaling for heterogeneous distributed real-time embedded systems," in *Proceedings of the 16th International Conference on VLSI Design*, 2003, pp. 369–375.

[138] J. Luo and N. K. Jha, "Static and dynamic variable voltage scheduling algorithms for real-time heterogeneous distributed embedded systems," in *Proceedings of the 15th International Conference on VLSI Design*, 2002, pp. 719–726.

[139] J. Luo, S. Peh, and N. K. Jha, "Simultaneous dynamic voltage scaling of processors and communication links in real-time distributed embedded systems ," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, 2003, pp. 1150–1151.

[140] N. Vijaykrishnan, N. Ranganathan, and N. Bhavanishankar, "DFLAP : A Dynamic Frequency Linear Array Processor," in *Proceedings of the International Conference on Image Processing*, 1996, pp. 1007–1010.

[141] N. Vijaykrishnan, N. Ranganathan, and N. Bhavanishankar, "A Dynamic Frequency Linear Array Processor for Image Processing," in *Proceedings of the International Conference on Pattern Recognition*, 1996, pp. 611–615.

[142] V. Krishna, N. Ranganathan, and N. Vijaykrishnan, "Energy Efficient Datapath Synthesis using Dynamic Frequency Clocking and Multiple Voltages," in *Proceedings of the International Conference on VLSI*, 1999, pp. 440–445.

[143] V. Krishna, N. Ranganathan, and N. Vijaykrishnan, "An Energy Efficient Scheduling Scheme for Signal Processing Applications," in *Proceedings of the thirty-second Asilomar Conference on Signal, Systems and Computers (Vol. 2)*, 1998, pp. 1057–1061.

[144] C. Papachristou, M. Spining, and M. Nourani, "A Multiple Clocking Scheme for Low Power RTL Design," *IEEE Transactions on VLSI Systems*, vol. 7, no. 2, pp. 266–276, June 1999.

[145] T. Burd, T. Pering, A. Stratakos, and R. W. Brodersen, "A Dynamic Voltage Scaled Microprocessor System," in *Proceedings of the IEEE International Solid-State Circuits Conference*, Feb 2000, pp. 294–295.

[146] T. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, "A Dynamic Voltage Scaled Microprocessor System," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1571–1580, Nov 2000.

[147] A. Acquaviva, L. Benini, and B. Ricco, "Processor frequency setting for energy minimization of streaming multimedia application," in *Proceedings of the 9th International Symposium on Hardware / Software Codesign*, 2001, pp. 249–253.

[148] L. Benini, E. Macii, M. Pnocino, and G. De Micheli, "Telescopic Units : A New Paradigm for Performance Optimization of VLSI Design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 3, pp. 220–232, Mar 1998.

[149] L. Benini, G. De Micheli, A. Lioy, E. Macii, G. Odasso, and M. Poncino, "Automatic Synthesis of Large Telescopic Units Based on Near-Minimum Timed Supersetting," *IEEE Transactions on Computers*, vol. 48, no. 8, pp. 769–779, Aug 1999.

[150] V. Raghunathan, S. Ravi, and G. Lakshminarayana, "High-Level Synthesis with Variable-Latency Components," in *Proceedings of the International Conference on VLSI Design*, Jan 2000, pp. 220–227.

[151] K. J. Nowka, G. D. Carpenter, E. W. MacDonald, H. C. Ngo, B. C. Brock, K. I. Ishii, T. Y. Nguyen, and J. L. Burns, "A 32-bit powerPC system-on-a-chip with support for dynamic voltage scaling and dynamic frequency scaling," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1441–1447, Nov 2002.

[152] K. Nowka, G. Carpenter, E. MacDonald, H. Ngo, B. Brock, K. Ishii, T. Nguyen, and J. Burns, "A 0.9 V to 1.95 V dynamic voltage-scalable and frequency-scalable 32 b PowerPC processor," in *Proceedings of the International Solid-State Circuits Conference (Vol. 1)*, 2002, pp. 340–341.

[153] Y. H. Lu, L. Benini, and G. De Micheli, "Dynamic frequency scaling with buffer insertion for mixed workloads," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System*, vol. 21, no. 11, pp. 1284–1305, Nov 2002.

[154] T. Pering, T. Burd, and R. W. Brodersen, "Dynamic Voltage Scaling and the Design of a Low-Power Microprocessor System," in *Proceedings of the Workshop on Power Driven Microarchitecture*, June 1998.

[155] T. Burd and R. W. Brodersen, "Design Issues for Dynamic Voltage Scaling," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2000, pp. 9–14.

[156] S. Hassoun and C. Ebeling, "Architectural Retiming : Pipelining Latency Constrained Circuits," in *Proceedings of the 33rd ACM / IEEE Design Automation Conference*, 1996, pp. 708–713.

[157] S. Nowick, "Design of a low-latency asynchronous adder using speculative completion," *IEE Proceedings on Computer Digital Techniques*, vol. 143, no. 9, pp. 301–307, Sep 1996.

[158] L. D. Strycker, P. Termont, J. Vandewege, J. Haitsma, A. Kalker, M. Maes, and G. Depovere, "Implementation of a Real-Time Digital Watermarking Process for Broadcast Monitoring on Trimedia VLIW Processor," *IEE Proceedings on Vision, Image and Signal Processing*, vol. 147, no. 4, pp. 371–376, Aug 2000.

[159] N. J. Mathai, D. Kundur, and A. Sheikholeslami, "Hardware Implementation Perspectives of Digital Video Watermarking Algortithms," *IEEE Transanctions on Signal Processing*, 2003.

[160] T. H. Tsai and C. Y Lu, "A System Level Design for Embedded Watermark Technique using DSC System," in *Proceedings of the IEEE International Workshop on Intelligent Signal Processing and Communication System*, 2001.

[161] A. Garimella, M. V. V. Satyanarayan, R. S. Kumar, P. S. Murugesh, and U. C. Niranjan, "VLSI Impementation of Online Digital Watermarking Techniques With Difference Encoding for the 8-bit Gray Scale Images," in *Proceedings of the International Conference on VLSI Design*, 2003, pp. 792–796.

[162] A. Antola, V. Piuri, and M. Sami, "A Low-Redundancy Approach to Semi-Concurrent Error Detection in Datapaths," in *Proceedings of the Design Automation and Test in Europe*, 1998, pp. 266–272.

[163] P. Kollig and B. M. Al-Hashimi, "Simultaneous Scheduling, Allocation and Binding in High Level Synthesis," *IEE Electronics Letters*, vol. 33, no. 18, pp. 1516–1518, Aug. 1997.

[164] G. Fetweis, J. Chiu, and B. Fraenkel, "A Low-Complexity Bit-Serial DCT/IDCT Architecture," in *Proceedings of the IEEE International Conference on Communications*, 1993, pp. 217–221.

[165] K. Balakrishnan, "Peak Power Minimization through Datapath Scheduling using ILP Based Models," M.S. thesis, University of South Florida, Spring, 2003.

[166] R. Fourer, D. Gay, and B. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Thomson Brooks Cole, 2003.

[167] P. E. Landman and J. M. Rabaey, "Architectural Power Analysis : The Dual Bit Type Method," *IEEE Transactions on VLSI Systems*, vol. 3, no. 2, pp. 173–187, Jun 1995.

[168] J. H. Satyanarayan and K. K. Parhi, "Theoritical Analysis of Word-Level Switching Activity in the Presence of Glitch and Correlation," *IEEE Transactions on VLSI Systems*, vol. 8, no. 2, pp. 148–159, Apr 2000.

[169] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, "Analytical Estimation of Signal Transition Activity from Word-Level Statistics," *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 16, no. 7, pp. 718–733, Jul 1997.

[170] S. P. Mohanty, N. Rangnathan, and S. K. Chappidi, "ILP Models for Energy and Transient Power Minimization During Behavioral Synthesis," in *Proceedings of the 17th International Conference on VLSI Design*, 2004, p. to appear.

[171] S. P. Mohanty, N. Rangnathan, and S. K. Chappidi, "Transient Power Minimization Through Datapath Scheduling in Multiple Supply Voltage Environment," in *Proceedings of the 10th IEEE International Conference on Electronics, Circuits and Systems*, 2003, p. to appear.

[172] S. S. Rao, *Engineering Optimization : Theory and Practice*, Addison-Wesley, 1996.

[173] M. J. Panik, *Linear Programming : Mathematics, Theory and Practice*, Kluwer Academic Publishers, 1996.

[174] B. A. McCarl and T. H. Spreen, *Applied Mathematical Programming using Algebric Systems*, Online Book at : http://agecon.tamu.edu/faculty/mccarl/regbook.htm, 1997.

[175] S. P. Mohanty, N. Rangnathan, and S. K. Chappidi, "Power Fluctuation Minimization During Behavioral Synthesis using ILP-Based Datapath Scheduling," in *Proceedings of the 21st IEEE International Conference on Computer Design*, 2003, p. to appear.

[176] S. P. Mohanty, N. Ranganathan, and R. K. Namballa, "VLSI Implementation of Invisible Digital Watermarking Algorithms Towards the Developement of a Secure JPEG Encoder," in *Proceedings of the IEEE Workshop on Signal Processing Systems*, 2003, pp. 183–188.

[177] A. Tefas and I. Pitas, "Robust Spatial Image Watermarking Using Progressive Detection," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (Vol. 3)*, 2001, pp. 1973–1976.

[178] F. Bartolini, M. Barni A. Tefas, and I. Pitas, "Image authentication techniques for surveillance applications," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1403–1418, Oct 2001.

[179] S. P. Mohanty, N. Rangnathan, and R. K. Namballa, "VLSI Implementation of Visible Watermarking for a Secure Digital Still Camera (S^2DC) Design," in *Proceedings of the 17th International Conference on VLSI Design*, 2004, p. to appear.

[180] V. P. Nelson, H. T. Nagle, J. D. Irwin, and B. D. Caroll, *Digial Logic Analysis and Design*, Prentice Hall, Upper Saddle River, New Jersey, USA, 1995.

[181] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design : A Systems Perspective*, Addison Wesley, Boston, MA, USA, 1999.

[182] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure Spread Spectrum Watermarking of Images, Audio and Video," in *Proceedings of the IEEE International Conference on Image Processing (Vol. 3)*, 1996, pp. 243–246.

[183] I.J.Cox, "A secure robust watermarking for multimedia," in *Proc. of First International Workshop on Information Hiding*, 1996, vol. 1174, pp. 185–206.

[184] M. Kaul, R. Vemuri, S. Govindarajan, and I. Ouaiss, "An Automated Temporal Partitioning and Loop Fission approach for FPGA based reconfigurable synthesis of DSP applications," in *Proceedings of the IEEE/ACM Design Automation Conference*, 1999, pp. 616–622.

[185] S. Govindarajan, I. Ouaiss, M. Kaul, V.Srinivasan, and R. Vemuri, "An Effective Design System for Dynamically Reconfigurable Architectures," in *Proceedings of the Sixth Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 1998, pp. 312–313.

[186] Karen Miller, *Assembly Language Introduction to Computer Architecture: Using the Intel Pentium*, Oxford University Press, 1999.

[187] J. B. Sulistyo and D. S. Ha, "Developing Standard Cells for TSMC 0.25um Technology under MOSIS DEEP Rules," Tech. Rep., Department of Electrical and Computer Engineering, Virginia Tech, VISC-2002-02, 2002.

ABOUT THE AUTHOR

Saraju P. Mohanty received the Bachelor of Technology degree in Electrical Engineering in 1995 from College of Engineering and Technology, Orissa University of Agriculture and Technology, Bhubansewar, India. He recieved the Master of Engineering degree in Systems Science and Automation from the Indian Institute of Science, Bangalore, India in 1999. He has taught several courses as instructor at department of Computer Science and Engineering, University of South Florida, USA and also at College of Engineering and Technology, Orissa University of Agriculture and Technology, Bhubaneswar, India. He has published several research papers in areas of VLSI design automation, VLSI design and Digital watermarking, and so on. His paper was nominated for best paper award at international conference in VLSI Design 2003. In the year 2002 and 2003, he recieved certificate of recognition from Provost, University of South Florida for outstanding teaching. His research interests are High-Level Synthesis for Low Power, Low-Power VLSI Design for Multimedia Applications, Computer Architecture, Digital Watermarking. He is a member of IEEE-CS and ACM-SIGDA.