FPGA based Reconfigurable 2D FFT System

Ahmad F. Al-allaf
Lecturer
Technical college-MOSUL
Dept of computer eng.
CTE
Ahmadalallaf@yahoo.com

Shefa A. Dawwd
Lecturer
College of Engineering
Dept of computer eng.
MOSUL University
shefadawwd@yahoo.com

Abstract

This paper develops a system level architecture for implementing a cost-efficient, FPGA-based reconfigurable two dimensional (2D) FFT system. The adopted approach considers both the hardware cost (in terms of FPGA resource requirements), and performance (in terms of throughput). These two extremes are optimized based on using run time reconfiguration, double buffering technique, shared Dual Ported RAM (DPRAM) modules and the "hardware virtualization" to reuse the available processing components. The system employs two one Dimensional (1D) FFT processor each with sixteen reconfigurable parallel FFT cores. Each core represents a 16 complex point parallel FFT engine. Thus the architecture supports transform length of 256X256 complex points, as a demonstrator to the design idea, using fixed-point arithmetic and has been developed using radix-4 butterfly architecture. The simulation results that have been performed using VHDL modeling language and ModelSim software shows that the full design can be implemented using single FPGA platform requiring about 50,000 Slices.

Keywords: 2D Fast Fourier Transform Radix-4. Run Time Reconfiguration

منظومة FPGA قابلة لاعادة التشكيل لتنفيذ خوارزمية FFT ثنائية الابعاد

احمد فالح محمود العلاف
مدرس/ اللكية التقنية- الموصل
قسم هندسة الحاسبات

شفاء عبد الرحمن داؤد
مدرس/ جامعة الموصل- كلية الهندسة
قسم هندسة الحاسبات

ألخلاصة

في هذا البحث تم تطوير معمارية بمستوى النظام جديدة لماكنة FFT قليلة الكلفة لمعالجات الزمن الحقيقي باستخدام دوائر ال FPGA . التصميم المقترح يأخذ بنظر الاعتبار تحقيق اقل الكلفة (بدلالة متطلبات موارد ال FPGA) واعلى انجاز (بدلالة ال throughput) وذلك من خلال استخدام خاصية اعادة التشكيل في زمن التنفيذ و خاصية الخزن المزدوج والذاكرات المشتركة ثنائية الاطراف وخاصية اعادة الاستخدام للمكونات المادية لدوائر ال FPGA. المنظومة المقترحة تستخدم 32 وحدة FFT متوازية قابلة لاعادة التشكيل مقسمة الى مجموعتين كل مجموعة (16 وحدة) تشكل معالج FFT لتنفيذ خوارزمية FFT احادية البعد. كل من هذه الوحدات هي عبارة عن معالج FFT متوازي بطول 16 نقطة مركبة . لذا فأن المعمارية المقترحة تستطيع تنفيذ خوارزمية FFT ذات البعدين بطول 256X256 نقطة مركبة وهو كمثال لتوضيح فكرة التصميم.

ان خوارزمية FFT التي تم اعتمادها في هذا البحث تستخدم معمارية فراشة الاساس-4 وحسابات النقطة الثابتة. في هذا التصميم تم اختيار اسلوب الحجيرة (Booth) المتوازية في بناء دوائر الضرب المركبة المطلوبة لانجاز عمليات الفراشة (وحدة البناء الاساسية لخوارزمية ال FFT) والتي تختصر المكونات المادية المطلوبة لتصميم دوائر الضرب قياسا للاساليب الاخرى. لقد اشارت نتاج المحاكات التي تمت باستخدام لغة VHDL و برنامج Model Sim الى ان التصميم الكامل للمنظومة يتطلب بحدود 50000 وحدة (Slices) ويمكن تنفيذها في FPGA واحدة.

Received: 16 – 2 - 2010 Accepted: 1 – 7 - 2010

1. INTRODUCTION

The two-dimensional Fast Fourier Transform (2D FFT) is widely-used for analyzing 2D signals, such as images. For example, the frequency domain image filtering is one of the most important applications where 2-D FFT can be applied. In general, the FFT algorithms are hardware intensive algorithm, and therefore require a lot of hardware. With the advances in VLSI technology, FFT algorithms are now implemented on programmable Digital Signal Processors (DSPs), FPGA devices and dedicated FFT processor ICs[1-4].

However, with the decreasing the cost and growing in the capacity, FPGAs have been widely used as coprocessors to boost the performance of data-intensive applications including the FFT algorithms[5, 6]. Despite of that, high performance, large-scale DSP algorithms still cannot fit in a single FPGA and require carful design considerations. In this context, the hardware implementation of the FFT algorithm can
be done in either fixed or floating point. Floating-point arithmetic requires much more area per operation (adders and multipliers). Furthermore, it has much higher demands on memory capacity and bandwidth.

To overcome this challenge, fitting the whole Fourier transform processor on a single FPGA chip is approached along two paths: First, the algorithm itself is examined and optimized specifically to minimize the hardware resources. Second, applying different techniques on the architecture (such as RC), to reduce the required hardware. Due to the inherent parallelism of FPGAs and tightly coupled memory and computational units, Reconfigurable Computing (RC) with FPGAs potentially offers high performance at lower chip area and lower power consumption[7].

This paper is concerned with the design of a new, 2D FFT architecture for implementing the algorithm over block of 256X256 complex point. We focused in this paper on finding the most suitable structure for implementing efficient and cost effective 2D FFT algorithm using RC technique. Design strategies have been placed on minimizing the logic and resource utilization of the implementation, leaving resources for additional functionality that required for the desired application. The system uses radix-4 architecture with fixed point arithmetic which is sufficient in many domains. The target hardware for the implementation and verification of proposed system is Xilinx FPGA development board equipped with a Xilinx Virtex-5(XC5VLX330T) of 51,840 slices.

The rest of this paper is organized as follows. A background and an overview of the related works are described in Section 2. Review of the proposed reconfigurable architecture for one and two dimensional FFT algorithm is presented in section 3. The details of the hardware design of the main components of the system are given in Section 4. Section 5 discuss the simulation results and performance analysis. Finally in Section 6, we offer some conclusions.

2. BACKGROUND AND RELATED WORKS

The FFT algorithm used in this implementation is based on radix-4 butterfly which is one of the most efficient methods of performing the FFT calculation. Radix-4 based FFT algorithm has better signal-noise ratio than that of radix-2 algorithm[8]. The radix-4 butterfly takes four complex input data words, computes the FFT, and produces four complex output

data words. Figure (1) illustrates the single flow graph of 16-pint radix-4 DIF-FFT algorithm..

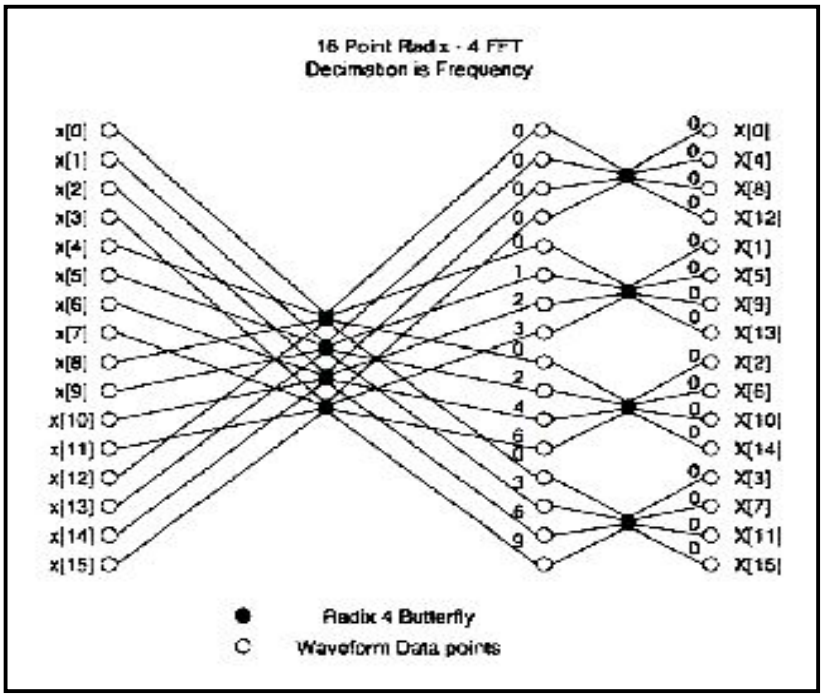


Figure (1) 16-point Radix-4 DIF-FFT signal flow diagram

Each butterfly requires four complex adder / subtractors and three complex multipliers. The mathematical model for each radix-4 butterfly is[9]:

$$X(4k) = \sum_{n=0}^{N/4-1}\left[x(n) + x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) + x\left(n+\frac{3N}{4}\right)\right] W_N^0 W_{N/4}^{kn} \quad \textbf{(1)}$$

$$X(4k+1) = \sum_{n=0}^{N/4-1}\left[x(n) - jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) + jx\left(n+\frac{3N}{4}\right)\right] W_N^n W_{N/4}^{kn} \quad \textbf{(2)}$$

$$X(4k+2) = \sum_{n=0}^{N/4-1}\left[x(n) - x\left(n+\frac{N}{4}\right) + x\left(n+\frac{N}{2}\right) - x\left(n+\frac{3N}{4}\right)\right] W_N^{2n} W_{N/4}^{kn} \quad \textbf{(3)}$$

$$X(4k+3) = \sum_{n=0}^{N/4-1}\left[x(n) + jx\left(n+\frac{N}{4}\right) - x\left(n+\frac{N}{2}\right) - jx\left(n+\frac{3N}{4}\right)\right] W_N^{3n} W_{N/4}^{kn} \quad \textbf{(4)}$$

Typically, 2D DFTs are implemented using the Row-Column (RC) decomposition technique [8] that involves a number of 1 dimensional Fourier transforms. More precisely, a 2D transform is achieved by first transforming each row, replacing each row with its transform and then transforming each column, replacing each column with its transform. Thus a 2D transform of a 256 by 256 image requires 512 1D transforms. Many different researches for the implementation of 2D FFT algorithms on FPGA have been proposed since the introduction of this technology. For example, XU et al. [10] proposed an FPGA-based reconfigurable, hierarchical-SIMD (H-SIMD) machine with its co-design of the Pyramidal

Instruction Set Architecture (PISA). He assumes a multiple FPGA board where each FPGA is configured as a separated SIMD machine to implement 2D FFT. While Shirazi et.al. [11] implemented a 2-D FFT on a custom computing machine called Splash-2, using floating point arithmetic. Dick [12] proposed a reconfigurable architecture for 2-D FFT using polynomial transforms on XC4000E FPGA device. He showed that his architecture is %46 efficient than a row column processor. K. D. Underwood et. al[13] develop an intelligent network adapter for cluster-based parallel computing using FPGA to implement 2D FFT algorithm. In [14], I. L. dalal present a single FPGA engine to perform real-time 2D FFT image reconstruction using array of up to 16 coils for parallel MRI. Recently, I. S. Uzun et. al,[15] an FPGA-based parametrizable environment based on the developing parallel 2D FFT architecture was presented for image filtering applications.

3. SYSTEM ARCHITECTURE

As mentioned before, the fundamental operations in implementing 2D FFT algorithm is equivalent to doing a 1D-FFT on the rows of the block of data and then doing a 1D-FFT on the columns of the result. In other words, we can implement 2D FFT algorithm by: Compute the 1D-FFT for each row, Transpose the matrix, Compute the 1D-FFT for each row, and Transpose the matrix. Figure 2 illustrates the structure of proposed reconfigurable pipelined 2-D FFT architecture. The architecture is structured with two *FFT processor*s; Rows FFT Processor (RFFTP) and Columns FFT Processor (CFFTP). The two processors are identical and each FFT processor is essentially a 1-D reconfigurable FFT processor with attached four DPRAM memory banks. DPRAMs are used to temporally store frames between two the processors, thus allowing a frame to stream from the input to the output. The four Dual-Ported RAM (DPRAM) memory banks (*DPRAM1…DPRAM4*) are used to facility the parallel accessing to those shared banks by FFT core of each processor under the control of one Address Generation and Control Unit (AGCU). We will begins by describing the architecture of reconfigurable 1D FFT processor.

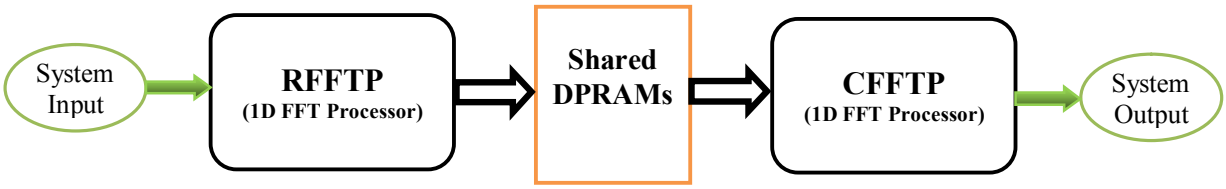


Figure (2) 2D FFT system block diagram

3.1Reconfigurable 1D FFT processor

The designed 1D FFT processor is implemented using reconfigurable parallel-pipeline architecture. The considered architecture focuses on minimizing and optimizing the hardware resources without large scarifying in performance. Each processor is designed to implement 256 point 1D FFT algorithm.

As shown in figure (4), each processor contains 16 FFT cores and each core can implement 16-point radix-4 FFT algorithm. The input data are grouped in 16 blocks; each block consists of 16 complex points which then distributed to 16 FFT cores for execution. When using Radix-4, the *N*-point FFT consists of log4 *(N)* stages, with each stage containing *N/4* Radix-4

butterflies. The signal flow graph for implementing 1D 256 complex point decimation in frequency (DIF) readix-4 FFT algorithm in proposed system is shown in figure (3).

The computation of 1D 256 point radix-4 FFT algorithm requires four stages implemented in four steps. In each step, one stage is executed. The result of each step is stored and then reused by the same hardware to execute the next step. The time required to compute an entire stage is equal for all stages. Data exchange is required between each group of four FFT cores after executing stage-1 and stage-2 of the algorithm [see figure (3)].

To speed up the operation of data transfer between the FFT cores and the I/O system and improving the throughput, a separate input and output buses are used for data input/output [see figure (4)]. Considering that there are two separate external memory models to download and upload the 2D FFT system with the I/O samples in parallel. A torus network is chosen to connecting the FFT cores to facilities data exchange between the stages and for the data output. This allows the sharing of data among neighboring cores which reduce the communication overhead.

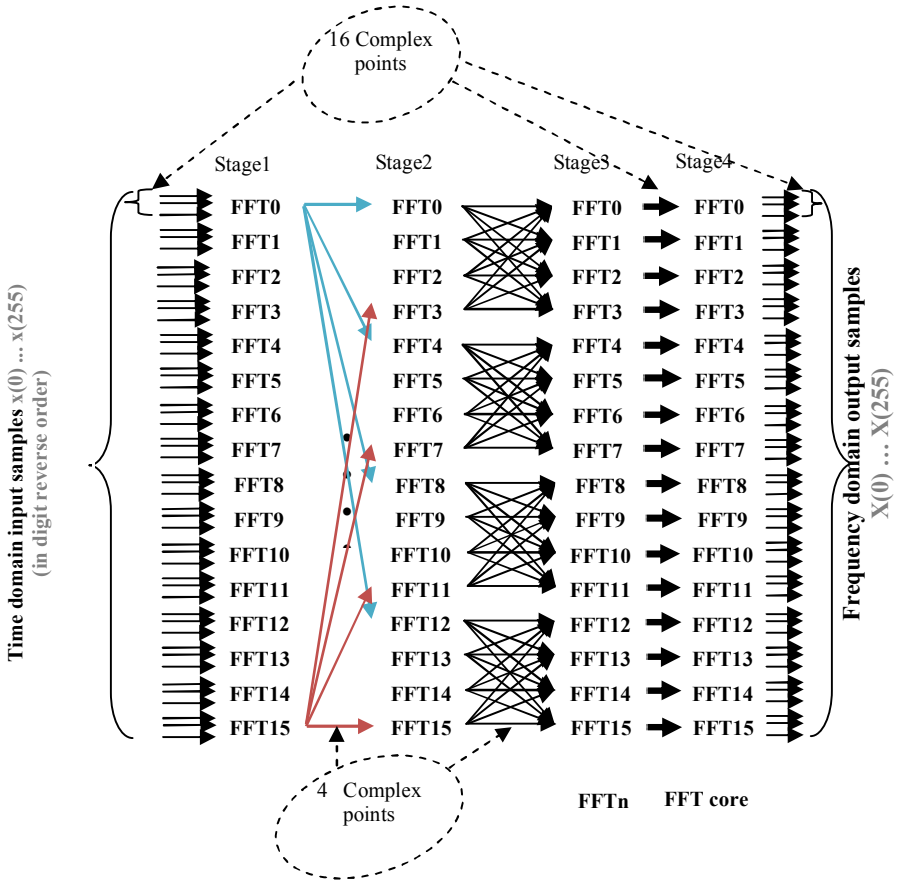


Figure (3) Flow graph of reconfigurable 1D FFT

Four Horizontal Buses (HB), and four Vertical Buses (VB), are used to connect FFT cores. Every bus connects group of four FFT cores. Vertically connected FFT cores exchange data between them after stage-1. Horizontally connected FFT cores exchange data between them after stage-2 [see figure (3)]. In either cases, every core send/receive four complex point of the intermediate result to/from other cores within its group.

3.2 Reconfigurable 2D FFT System

Figure (4) illustrates the structure of proposed reconfigurable pipelined 2-D FFT system. As mentioned earlier, this architecture is structured with two *FFT processor*s; Rows FFT Processor (RFFTP) to perform 1D FFT algorithm on rows, and Columns FFT Processor (CFFTP) to perform 1D FFT algorithm on columns. The two processors share four Dual-Ported RAM (DPRAM) memory banks (*DPRAM1…DPRAM4*) under the control of one Address Generation and Control Unit (AGCU). The size of DPRAM modules is depend on the size of the matrix and the word length. Selecting of the word lengths makes a compromization between two factors: (quantization noise) due to iterative process of the FFT algorithm, and precision of the FFT computations. It is desirable to have short wordlengths and less complex circuitry. Short word lengths lower the cost in terms of chip area and power consumption. On the other hand, short word lengths lower the precision of the FFT computation. Therefore great care must be shown in selecting word lengths.

Since the target output is the frequency components of the input signal. Therefore, the amplitude of the input signal is effective less and can be normalized. Based on this fact, we assumed that the input signal is in the range ±1, and using fixed point implementations, each of the real and imaginary parts of the I/O data are represented in 16-bit format with one bit for sign, one bit for integer and 14-bit for fraction. Therefore, all data pathways (buses) are also in 16-bit two's complement signed format.

During the 1D FFT computation results at a particular stage are scaled and rounded to 16-bit. Based on this assumption, the required shared memory to store intermediate results of implementing FFT algorithm on matrix of 256X256 complex point is 256KB. Therefore each of the four DPRAM banks is of size 64KB organized as 32KX16bit in order to store the real and imaginary parts of complex numbers in successive memory locations. Every of the 16 FFT cores share one memory bank. This memory architecture allows subsequent input blocks to be processed in a continuous, pipeline manner so that all of the FFT cores in the two processors can be engaged all the time. The twiddle factors are precomputed in 8-bit format and are stored in local memory within the FPGA at configure time. Interprocessor synchronization is required in accessing the shared memory by the two processors. Synchronization is done via AGCU such that CFFTP begin executing FFT on columns of block (*i*), only after RFFTP finished executing FFT on rows of block (*i*).

In addition for the communication between FFT cores, the vertical buses are also used as data input buses (to down load the input data from the external memories) in case of RFFTP and as data output buses (to upload the result of computation to the external memories) in case of CFFTP. While the horizontal buses are also used to upload (download) the intermediate result to (from) shared DPRAMs for the PFFTP (CFFTP). The input and output systems are a set of input/output circuitry within the FPGA.

The address generation and control Unit (AGCU) is a state machine provides the required addresses to manage the communication interface between the two FFT processors. In addition, it provides a number of control signals to coordinate and to synchronize the activity of different units in the 2D FFT system and to initiate processing and monitor its completion. Also, the AGCU provides the addresses of DPRAMs and producing the global clock and control signals to manage data load/store and transportation process in the system, as we will be explained in paragraph 5. With suitable hardware permutations schemes, this circuitry allows for the scheduling of intermediate results in shared DPRAMs, so that the correct

butterfly operations on the block columns are preserved. Finally, the ACGU controls and configures the input/output system to route the input and output data between FFT processors and external world. Our implementation of the algorithm focuses, in addition to area cost, on optimizing the execution time by hidden the matrix transpose time with the FFT execution time.

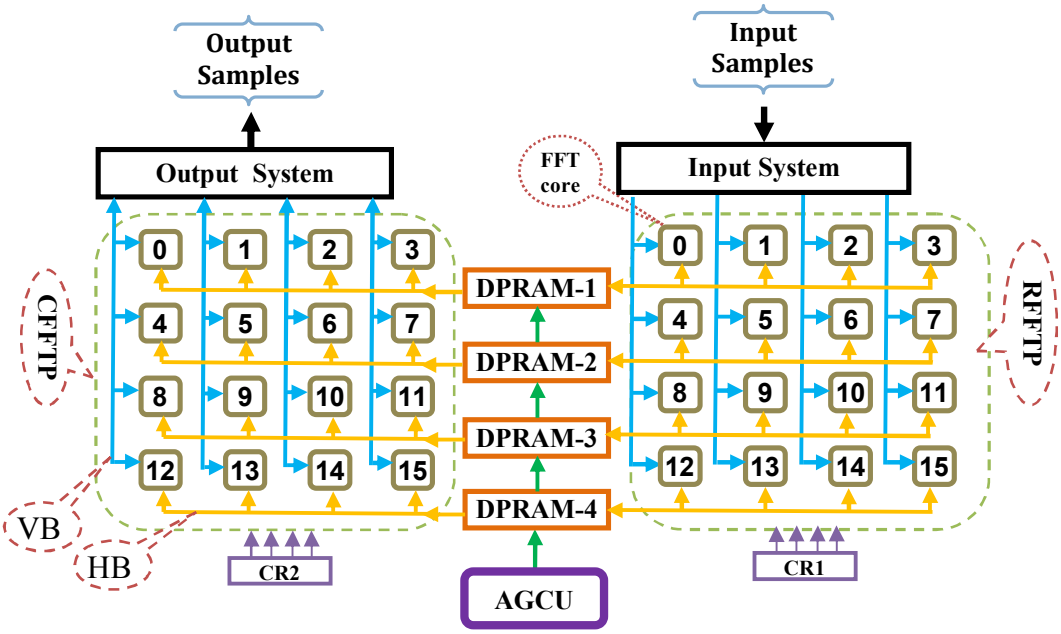


Figure (4) 2D FFT system architecture

4. FFT CORE HARDWARE DESIGN

As illustrated in previous section, each 1D FFT processor contains 16 FFT cores for performing the butterfly computations needed for the FFT algorithm. Every FFT core consists of four FFT radix-4 butterflies (which are referred as the basic radix-4 butterfly (BR4B) processing element) in a full parallel configuration, a local FIFO buffers of size 16 complex points (Local 16FIFO) to store the intermediate results of the computation, two sets of 16X32 bit deep FIFO buffers, and the Local Control and Configuration Unit (LCCU). The FPGA receives input samples from external source and then distributing them, under the control of AGCU, to one of the two sets of 16FIFO buffers in each FFT core. By utilizing this double buffering structure, the two sets of 16FIFO buffers are used to feed the FFT cores alternatively, thus solving the problem of data latency in data distributing process. Moreover, double buffering architecture allows subsequent input frames (rows or columns) to be processed in a continuous, pipeline fashion. Also, this allows real-time processing is achieved, and all of the butterflies in all FFT cores can be engaged all the time. The block diagram representation of the FFT core is depicted in figure (5).

In the beginning of processing of a new data vector, the input data are loaded from one of the two 16FIFO input buffers and distributed to the operand registers within each BR4B. During the computation, the intermediate results of the BR4B units are stored in local 16FIFO buffer. Then routed through the VB or HB to other cores (data exchange), depending on the stage being executed. After the end of computation of the current frame,

the result is return back to the same 16FIFO input buffer, (replacing the input frame) - in which is sent in next time - through the HB to the shared memory (for RFFTP), and through the VB to the outside world (for CFFTP).

The LCCU is mainly responsible for sequencing the execution of local hardware on the FFT core during different execution phases. It sends number of control signals to set the MUXs and DMUX within the core to appropriate configuration to rout the data between components of the cores. Furthermore, it sends enable signals to the FIFO buffers of the core and to the operand registers of the BR4B units to perform the data exchange between the FFT cores. The AGCU provides different clock signals to keep track of data input/output in FIFO buffers and switching between the 16FIFO input buffers during data load/store phase and FFT computation phase. It implies that a particular stage of the FFT computation is done, either the input or output process is done, and the FFT computation process is accomplished. Finally, is also responsible of producing counting signals to address the coefficient ROM within every butterfly to provide the multipliers with the correct twiddle factors.

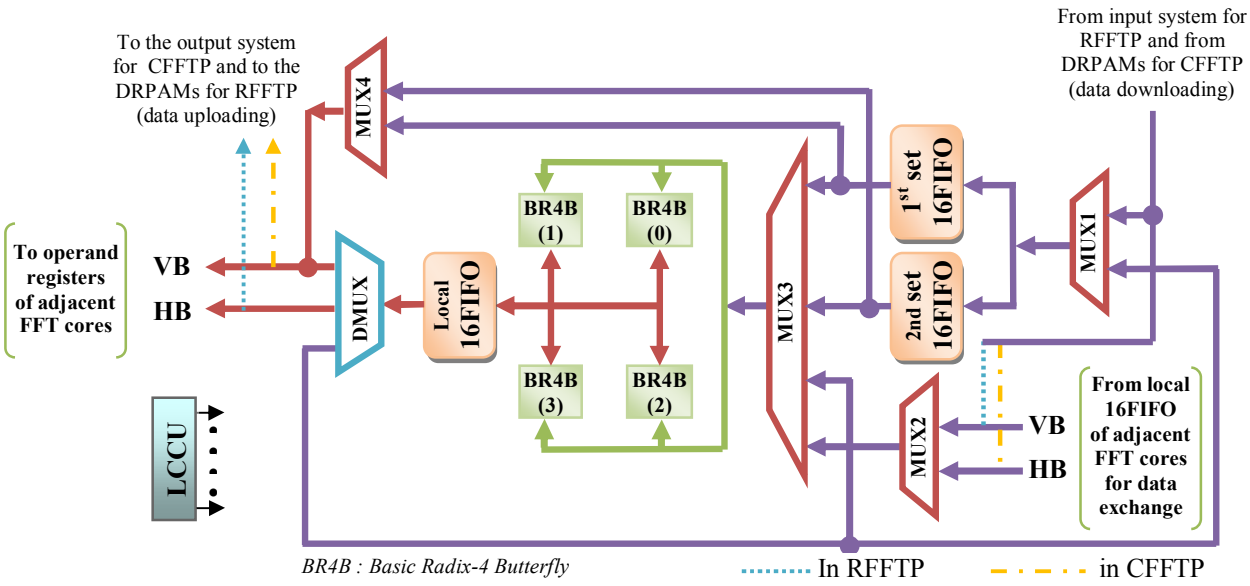


Figure (5) Architecture of the FFT core

4.1 Basic Radix-4 Butterfly (BR4B) elements

The main advantages in utilizing a radix-4 butterfly operation is that it has better speed performance, in spite of its complexity, and require less hardware compared to Radix-2. It requires 3 complex multiplies and 4 complex additions. Therefore, the total cost in complex multipliers is 75% of radix-2 FFT, although it uses the same number of complex additions. Figure (6) shows the main components of the basic radix-4 butterfly (BR4B) elements.

Each BR4B consists of three booth 16*8bit complex multipliers, and 16-bit adders/subtractors. Also each BR4B has eight 16-bit operand registers that accept (under the control of LCCU) the real and imaginary parts of four complex input points.

Coefficients (or twiddle factors) are pre-calculated and stored in local coefficient ROM as 8-bit two's complement signed fixed-point words for each butterfly to achieve parallel access to twiddle factors by all butterflies. The adder/subtractors perform the butterfly

operations on the data stored in the operand registers. The result is then send to the complex multiplier to multiply it by the twiddle factors.

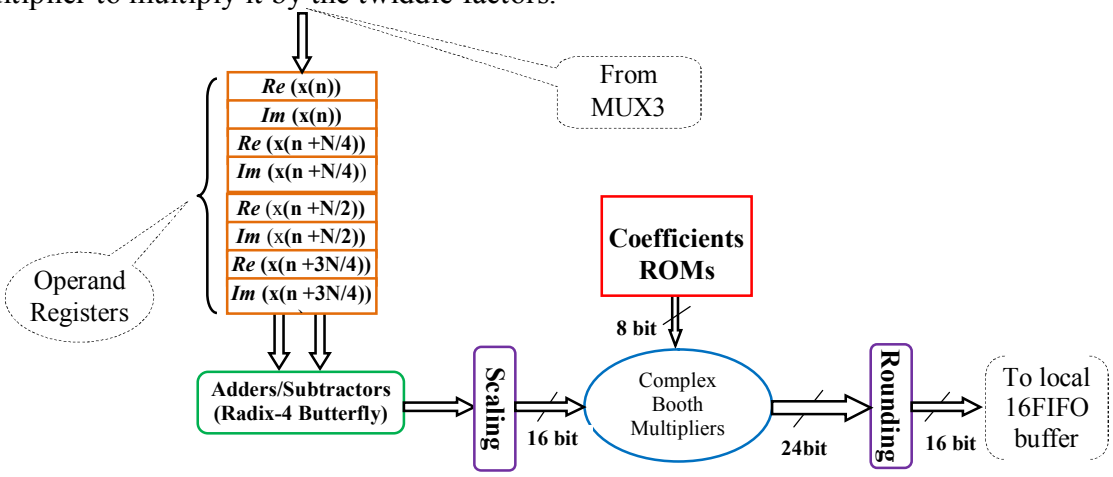


Figure (6) Basic Radix-4 Butterfly (BR4B) Datapath

The scaling of the intermediate results in FFT computation is necessary in order to prevent overflows with fixed-point arithmetic[14]. The 16 bit results of the butterfly operation are scaled by ¼ by applying right shift. The 24 bit result, growth of the fractional bits created from the multiplication, are rounded to drives the dynamics range back to 16-bit, which is sufficient in most practical cases[16]. The results are then stored for further processing in next stages. The design of the butterfly unit was simulated using the gate level simulator ModelSim. Both functional and post layout simulations were performed to confirm the correct operation of the design.

4.2 Complex multiplier

It is important to minimize the silicon area of the FPGA circuits, which is achieved by reducing the number of functional units (such as adders and multipliers), registers, multiplexers and interconnection wires.

The complex multiplier is the key component in the data processing. The direct implementation of complex multiplier requires 4 real multipliers, one adder and one subtractor. Furthermore, the multiplications are the most power dissipating arithmetic operations. Today's FPGA, contain a number of speed optimized signal processing building blocks, such as multipliers, RAM blocks or I/O structures with propagation delays in the range of a few nanoseconds [19].

The complex multiplier used in this work requires only three real multipliers and four adder/subtractors. The multiplication of two complex numbers in component form is given by:

$$U=(x_r+jx_i)(a_r+ja_i) = (x_ra_r-x_ia_i)+j(x_ia_r+x_ra_i) \quad (5)$$

Inherent dependencies that can be used to reduce the circuit costs. Several possibilities exists for carrying out complex multiplication with just three instead of four multiplications. For instance, eq(5) can be replaced by the following equation as an alternative:

$$U=[x_r(a_r+a_i) - (x_r+x_i)a_i] +j [x_i(a_r-a_i) + (x_r+x_i)a_i] \quad (6)$$

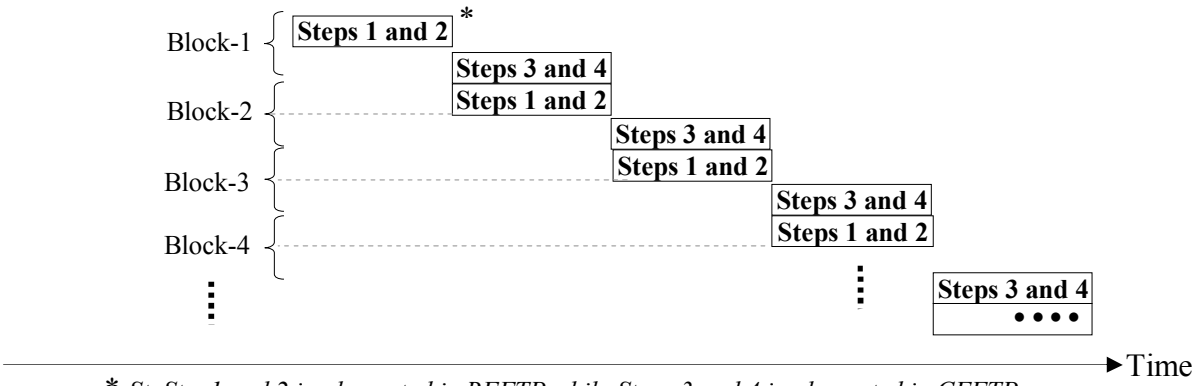
Note that the result of the term $(x_r + x_i)a_i$ is used to evaluate both the real and the imaginary parts of U, then the complex multiplication is implemented with three multiplications and five additions.

5. PROCESSING 2D FFT ALGORITHM IN PROPOSED SYSTEM

In general, algorithm for 2D FFT is four serialized steps:

1. *FFT of Rows*
2. *Matrix Transpose*
3. *FFT of Rows*
4. *Matrix Transpose*

In present architecture, matrix transposition is implied during data storing/loading in the DPRAMs. The two processors in steps 1 and 3 operate independently on their local data of sequential blocks in pipeline architecture. Steps 2 and 4 are implemented by the AGCU in parallel on sequential frames.
These four steps are overlapped between RFFTP and CFFTP as shown in figure (7).



* *St Step1 and 2 implemented in RFFTP while Steps 3 and 4 implemented in CFFTP*

Figure (7) Overlapping in implementing 2D FFT algorithm in proposed system

The 2D FFT algorithm starts by loading the first row of the matrix from the external memory in digit reverse order and distributed to the 1st set of the input 16FIFO buffers in the RFFTP under the control of AGCU. Then 1D DIF FFT is computed on the stored data. When this phase is completed, the data is returned to the same input 16FIFO buffer that read from it. After that the RFFTP start directly to process the second row of input data from the 2nd set of input 16FIFO buffers. The result of the first row is read out from the 1st set of 16FIFO buffers in the RFFTP processor (under the control of AGCU) and stored (in-order) as a column vector in the shared DPRAMs. By this method the transposition phase is done in parallel with the data computation phase.

When full block (256 rows) is processed by the RFFTP and stored as column vectors in DPRAMs, the CFFTP start by processing the first row vector of the block.
Under the control of AGCU, row vector of data are read from DPRAM and stored in the 1st set of input 16FIFO buffers in the CFFTP. After that, a 1D DIF FFT is computed on stored data as described for RFFTP. After every processed row vector, the AGCU read the data from the input 16FIFO buffer in the CFFTP and stored through the output system in external RAM in transposition order.

In real time operation, the CFFTP start the operation after a block of 256 input vector has been processed by the RFFTP. This is the case for the first block only. In subsequent blocks both the PFFTP and CFFTP operates concurrently on sequential blocks with no delay penalty as follows:

In the first block of 256X256 complex points, the result of RFFTP is stored as a column vectors and read from the DPRAMs (by the CFFTP) as a row vectors (to perform the transposition). In the next block of 256X256 complex point, the result of the RFFTP is stored as a row vectors (replacing the processed row vector by the CFFTP) and read from the DPRAM (by the CFFTP) as a column vectors. This operation which is repeated alternatively, in subsequent blocks, is necessary to allow both processors to use the same shared memory at the same time with different blocks. The correct addresses of both ports is generated by ACGU.

The data input, computation and data output operations are overlapped, so that the FFT system is never left in an idle state waiting for an I/O operation. This provides high throughput rates for real-time applications, in which the input data is a sequential stream.

6. SIMULATION RESULTS AND PERFORMANCE DISCUSSION

This paper presents a proposed architecture for the development of a 256X256 point radix-4 2D FFT system targeting low-cost FPGA technologies. Designing DSP algorithms using FPGA presents great advantages due to its parallel processing method and its flexible structure, high integration and velocity. The proposed architecture uses reconfigurable computing to carefully integrates two orthogonal methods for trading-off hardware cost and performance. This type of implementation leads to decrease in the silicon area at the cost of increasing of processing time. However, different methods are used to increase the performance such as using double buffering technique, parallel butterflies' execution, shared memory, and pipeline scheme. Additionally, using these techniques and partitioning application into coarse-grain tasks, the communication overheads can be hidden.

In order to overcome the data I/O latency problem, the proposed architecture employs double buffering technique within each 1D FFT processor by using two sets of input FIFO buffers. The switching between those pairs of input FIFO buffers overlaps data communications with computations. Also double buffering technique allows to continues data stream processing, thus real-time processing is achieved. The global controller (AGCU) lies in the FPGA and controls all the transactions between the two processors and between the FPGA and the external world.

In our implementation, the communications among the butterflies in the 1D FFT processor are based on a nearest neighbor's grid interconnection. Data needed by every butterfly can be routed from its neighbor by using a set of operand registers and FIFO buffers. The system has been simulated using Modelsim software. Results of simulation showed that the entire system can be implemented in one FPGA Virtex-5(XC5VLX330T) of 51,840 slice, 648 of 2kByte BRAMs and multipliers. According to Table 1, one can see that the designed circuit totally consumes 256kB of block RAMs and 128 complex multipliers(384 real multipliers).This means that 19% of the available BRAMs and 60% of embedded multipliers are consumed. The architecture supports scaled fixed point arithmetic

methods. The inputs to the FFT are 16 bits wide, 14 bits of fraction, signed bit, and 1 bits for integer(assuming that the input signal in the range between (+1 and -1). We have implemented basic radix-4 butterfly element on Spartan-3E (XC3S500E) evaluation board of 4656 slices requiring 346 slices and 3 embedded multipliers. Based on this implementation, we can estimate that one FFT core approximately requires (346*4) slices, which consumes 30% of the total number of slices. The entire proposed system (with 32 FFT cores) can be implemented using a single FPGA platform of 50,000 slices or more.

Table 1: Component utilization summary for Virtex-5(XC5VLX330T)

	Available components	Consumed components
No. of slices	51,840	50,000
No. of MUL	648	384
No. of 2kByte BRAMs	648	128

The system cost is compared with previous work in which FFT is implemented sequentially[17]. A pipelined 256x256 points 2D FFT module is designed. This module is realized by 1D FFT along rows and 1D FFT along columns. A 1D FFT circuit is based on the Xilinx 64-point FFT IP core and dual port RAMs. Four 1D-FFT circuits are applied for 256x256 points 2D FFT module. 5149 slices and 8 Block RAMs are used to implement this module. The Xilinx FFT IP core[18] uses the Radix-4 and Radix-2 decomposition for computing the DFT. One Radix-4 or Radix-2 butterfly is implemented and then used sequentially. If it is required to compute all the 256x256 points in parallel based on the FFT IP core, then a huge number of FPGA recourses should be consumed.

7. CONCLUSIONS

An efficient architecture for the implementation of 2-D FFTs has been proposed and implemented. The performances of implementations have been discussed. The system uses two 1D FFT processor, one for implementing 1D FFT algorithm on the rows and the other processor for implementing 1D FFT algorithm on the columns to realize the pipeline execution. Also the system uses shared DPRAM blocks to allow simultaneous communication between the two processors. Thus, hiding the communication overheads leads to improve the performance.

The complex fixed point 256X256 2D-FFT algorithm has been implemented as an example of an application that benefits from reconfigurable computing . The architectural enhancement that we propose is the insertion of reconfigurable computing technology in the data path of the 2D FFT system. According to figures 3 and 4, one can see that the proposed reconfiguration strategy enables to reduce the number of real multiplier to the half. Also, area efficiency and low cost have been attained for the parallel reconfigurable implementation of 2-D FFT compared to existing works[17,18].

The proposed architecture has lower resource usage than the others such as fully parallel architecture. The stage processing of the algorithm uses shared adder/subtractor, hence reducing resources at the expense of an additional delay per 1D FFT calculation.

REFERENCES

[1] Datasheet, "Analog Devices DSP Selection Guide, 2002 Edition", Analog Devices, 2002.
[2] Datasheet, "Motorola DSP 56600 16-bit DSP Family Datasheet", Motorola Ltd.,2002.
[3] Datasheet, "Xilinx 1024-point FFT/IFFT Core Datasheet", Xilinx Ltd., 2002.
[4] P. A. Jackson, C. P. Chan, J. E. Scalera, C. M. Rader, and M. M. Vai, "A Systolic FFT Architecture for Real Time FPGA Systems", High Performance Embedded Computing Workshop, 2004.
[5] T. Dillon, "Two Virtex-II FPGAs Deliver Fastest, Cheapest, Best High-Performance Image Processing System", Xilinx Xcell Journal, 41, 2001.
[6] Actel Corporation "CoreFFT Fast Fourier Transform" May 2007, www.actel.com
[7] Weidong Li and Lars Wanhammar, "Efficient Radix-4 and Radix-8 Butterfly Elements", available at: http://www.es.isy.liu.se/publications/papers_and_reports/1999/weidonglNorChip 99. pdf
[8] S.Y. Kung, *VLSI Array processors*, Englewood Cliffs, New Jersey: Prentice-Hall, 1988.
[9] Charles Wu, "Implementing the Radix-4 Decimation in Frequency (DIF) Fast Fourier Transform (FFT) Algorithm Using a TMS320C80 DSP", application report: SPRA152, www.ti.com, 1998.
[10] Xizhen XU*y and* Sotirios G. Ziavras, "A Coarse-Grain Hierarchical Technique for 2-Dimensional FFT on Configurable Parallel Computers.," IEICE Transc. Inf. & Syst., Vol. E89D, No. 2 Feb. 2006.
[11] Nabeel Shirazi, Peter M. Athanas, and A. Lynn Abbott, "Implementation of a 2-D Fast Fourier Transform on a FPGA-Based Custom Computing Machine", IEEE Symposium on FPGAs for Custom Conf. Comp. Mach., September 1999.
[12] C. Dick, "Computing Multidimensional DFT using Xilinx FPGAs", The 8th Intr. Conf. On Sig. Pro. App. and Tech., September, 1998.
[13] Keith D. Underwood, Ron R. Sass, and Walter B. Ligon, "acceleration of a 2D FFT on an adaptable computing cluster" proceedings of the 9th annual IEEE symposium on field programmable custom computing machine (FCCM'01), 2001.
[14] Ishaan L. Dalal and Fred L. Fontaine "A Reconfigurable FPGA-based 16-Channel Front-End for MRI,", at the 40th Asilomar Conference on Signals, Systems and Computers in Pacific Grove, CA, on October 30, 2007.
[15] I. S. Uzun , A. Amira and A. Bouridane, "FPGA Implementations of Fast Fourier Transforms for Real-Time Signal and Image Processing", VISP(152), No. 3, pp. 283-296, June, 2005,.
[16] Bernd Jähne, "Digital Image Processing", Springer-Verlag Berlin Heidelberg 2005.
[17] Kazuhiro Shimizu , Shinichi Hirai, "Implementing Planar Motion Tracking Algorithms on CMOS+FPGA Vision System", Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 9 - 15, 2006, Beijing, China.
[18]"Fast Fourier Transform v5.0", DS260 October 10, 2007, Product Specification, Xilinx, Inc.
[19] *Virtex-II Pro Platform FPGA Handbook,* Xilinx, Inc., San Jose, CA, 2002

The work was carried out at the college of Engg. University of Mosul