Performance Analysis of Different Arbitration Algorithms of the AMBA AHB Bus

Massimo Conti, Marco Caldari, Giovanni B. Vece, Simone Orcioni, Claudio Turchetti
DEIT, Università Politecnica delle Marche, via Brecce Bianche, I-60131, Ancona, ITALY
m.conti@univpm.it www.deit.univpm.it

ABSTRACT
Bus performances are extremely important in a platform-based design. System Level analysis of bus performances gives important information for the analysis and choice between different architectures driven by functional, timing and power constraints of the System-on-Chip. This paper presents the effect of different arbitration algorithms and bus usage methodologies on the bus AMBA AHB performances in terms of effective throughput and power dissipation. SystemC and VHDL models have been developed and simulations have been performed.

Categories and Subject Descriptors
B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids.

General Terms
Performance, Algorithms, Design.

Keywords
AMBA AHB BUS, SystemC, Arbitration Algorithm, Performance

1. INTRODUCTION
System-level design and intellectual property (IP) modeling is the key to fast SoC innovation with the capability to quickly try out different design alternatives, to confirm the best possible architecture, HW/SW partition and performance parameters, including power consumption, early in the design process. To innovate quickly, system-level design provides a high level of abstraction, very fast simulation speed and allows a high degree of IP reuse. A possibility to implement an efficient system-level design is the use of object-oriented programming languages, like C++. SystemC 2.0 [1-2] is an emerging standard modeling platform based on C++ that supports design abstraction at the RTL, behavioral and system levels. SystemC 2.0 provides a common design environment consisting of C++ libraries, models and tools providing the ability to exchange and reuse IPs easily and efficiently across different levels of abstraction.
One of the goals of the EDA community is the integration of power analysis and optimization techniques into IP modeling methodologies. This is an important design reuse aspect that is getting increasing relevance in the IP qualification process, whose aim is to establish objective and standardized criteria to check the quality of an intellectual property not only in terms of its functionality.
Power dissipation analysis should be performed in the first phases of the design when some good ideas on power dissipation can drive the choice between different architectures, together with the requested functional and timing specifications.
Great part of power is dissipated in a CMOS circuit in charging node capacitances. A strong reduction of dynamic power dissipation can be obtained if the switching activity of the big capacitances is reduced. In a System on Chip the bus lines have a capacitance order of magnitude bigger than gate capacitances. Many techniques, especially of bus encoding have been studied to reduce bus switching activity [3-8].
In this paper the effect of different arbitration algorithms and bus usage methodologies on the bus AMBA AHB performances, in terms of effective throughput and power dissipation, is shown.

DAC 2004, June 7-11, 2004, San Diego, California, USA

2. AMBA BUS VHDL and SystemC Models
The AMBA protocol defines a standard for on-chip communication and it is an efficient tool for the development of high-performance embedded systems. AMBA specification [9] aims at satisfying four important requirements:
- to allow the *right-first-time* development of embedded controllers with different CPU or DSP cores (multiple masters);
- to be *technology-independent* and to allow a high reusability of different blocks;
- to encourage a *modular system design* to preserve the best possible CPU's independence;
- to facilitate the testing phase.

AMBA specification defines three different bus topologies: the AHB, the Advanced System Bus (ASB) and the Advanced Peripheral Bus (APB). We have used the AHB that is the last generation of AMBA bus, targeted for high level performances.
The AMBA AHB bus can be decomposed in the following main blocks: one arbiter, a decoder and some multiplexing logic for read and write operations.
The AMBA AHB bus has been described in SystemC 2.0 and in VHDL in a clock accurate description [10-11]. The number of bits of ADDR and DATA lines, and the number of masters and slaves connected to the bus are parameters that can be easily changed in the SystemC code.
Five arbitration algorithms have been tested :
1) Priority with break: the masters have a fixed priority. A bus request by a higher priority master breaks a lower priority transmission.
2) Priority: the masters have a fixed priority. Once a master takes the bus, the transmission is not interrupted.
3) Priority with waiting time control: the masters have a fixed priority. When a master waits for a time longer than a fixed

value, his priority becomes the highest. Once a master takes the bus, the transmission is not interrupted.

4) Short Job First: the master with the shortest bus occupation request takes the bus. Once a master takes the bus, the transmission is not interrupted.
5) Short Job First with waiting time control: when a master waits for a time longer than a fixed value, it takes the bus, otherwise the master with the shortest bus occupation request takes the bus. Once a master takes the bus, the transmission is not interrupted.

3. SIMULATIONS AND RESULTS

Many SystemC and VHDL simulations have been performed to evaluate the effect of different arbitration algorithms and bus usage methodologies on bus AMBA AHB performances in terms of effective throughput and power dissipation.

The bus performances have been evaluated in test cases where the bus is intensively used. For this reason the transmission used is of the type "incrementing burst" that is usually chosen for the transfer of large amount of data. The masters are simple traffic generators. A C++ program generates a random traffic for each master respecting the desired statistics.

3.1 Inter-burst idle insertion

A first set of VHDL simulations have been performed to verify the effect of different bus occupation techniques on bus performances and power consumption.

The bus arbitration algorithm is the simple fixed priority algorithm (algorithm 1 in Sect. 2). The simulated architecture consists of a default master, a master with high priority (M1), a second master with low priority (M2), and a slave (SL) connected to the 32 bit data bus. Each master transmits sequences of 512 bytes for a total of 5k bytes transmitted, with a random idle period between the transmissions.

The low priority master M2 transmits 10 bursts of 128 beats each. Master M1 transmits each packet of 512 bytes in different ways in the different simulations: 1 burst of 128 beats, 2 bursts of 64 beats, 4 bursts of 32 beats, 8 bursts of 16 beats, 16 bursts of 8 beats. Each burst is followed by an idle period of a length of 4, 8 or 16 clock cycles in the different simulations. An example of this transmission sequence is reported in Fig.1.

The master with high priority intentionally divides its transmission in bursts to enable the use of the bus by other master during its idle periods. This technique avoids long waiting times to low priority masters, keeping simple the arbitration algorithm.

The aim of the simulations is to verify the effect of this technique on bus throughput and bus switching activity.

Table 1 reports the number of clock cycles needed to complete the 5k bytes transmission of the masters M1 and M2, for the 15 types of simulations (different burst length and idle cycles).

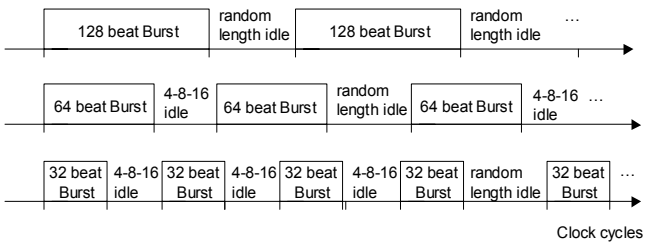

Figure 1 - An example of master M1 transmission sequences

Table 1. Number of clock cycles to complete the transmissions

Inter-burst cycles	16	8	4
Bursts length	**Clock cycles to complete transmissions**		
128 beats	2734	2734	2734
64 beats	2796	2785	2787
32 beats	2872	2882	2881
16 beats	3054	3059	3077
8 beats	4409	3432	3468

Table 2. Second Master waiting time

Inter-burst cycles	16	8	4
Bursts length	**Second master waiting time (clock cycles)**		
128 beats	66,7	66,7	66,7
64 beats	44,4	47,5	45,9
32 beats	29,9	29,7	29,6
16 beats	19,1	18,5	18,3
8 beats	11,4	11,7	11,5

Table 3. Percentage of bus occupation

Inter-burst cycles	16	8	4
128 beats bursts	**Percentage bus occupation**		
Def.	0,15	0,15	0,15
M1	94,00	94,00	94,00
M2	5,85	5,85	5,85
64 beats bursts	**Percentage bus occupation**		
Def.	1,14	0,29	0,29
M1	84,48	88,80	91,54
M2	14,38	10,91	8,17
32 beats bursts	**Percentage bus occupation**		
Def.	0,77	0,28	0,21
M1	72,21	81,05	87,86
M2	27,02	18,67	11,93
16 beats bursts	**Percentage bus occupation**		
Def.	1,24	0,39	0,19
M1	56,78	70,92	81,29
M2	41,98	28,69	18,52
8 beats bursts	**Percentage bus occupation**		
Def.	2,18	0,58	0,23
M1	43,60	59,79	73,59
M2	54,22	39,63	26,18

The length of the transmission increases considerably for 16 and 8 burst length transmissions, in fact master M1 leaves the bus free for a time longer than what it is required by master M2.

Table 2 reports the mean value of master M2 waiting time (in number of clock cycles) for the different simulations. Master M1 takes the bus one clock cycle after his request. The result show that the waiting time is not effected by the inter-burst idle length.

The waiting time decreases as the burst length decreases, as expected. Table 3 reports the percentage of bus occupation for the first half of each simulation for the 3 masters: default, M1 and M2. At the end of the simulation M1 and M2 have, of course, the same percentage, in fact they transmit the same amount of data.

The low priority master has no time windows to transmit in the case of 128 beats bursts. It can use the bus only during the random idle periods between M1 transmissions. The percentage of bus occupation for master M2 increases decreasing the M1 burst length or increasing the idle time between two bursts.

The percentage of time the bus is not used is always low, but increases for long inter-burst idle periods.

Table 4. Switching activity

Burst type	*128 beats (adr inc.)*			*128 beats (no adr inc.)*		
Idle cycles	16	8	4	16	8	4
Signals	**Number of signals transitions**					
Inputs	189	189	189	133	133	133
Flip-flops	204	204	204	210	210	210
Internal	420	420	420	426	426	426
Outputs	133	133	133	137	137	137
HADDR	5242	5242	5242	36	36	36
HWDATA	39004	39004	39004	39038	39038	39038

Burst type	*64 beats (adr inc.)*			*64 beats (no adr inc.)*		
Idle cycles	16	8	4	16	8	4
Signals	**Number of signals transitions**					
Inputs	291	287	285	195	193	193
Flip-flops	292	288	290	290	288	292
Internal	634	620	626	628	620	632
Outputs	199	195	197	197	195	199
HADDR	5324	5320	5308	84	82	84
HWDATA	39152	39168	39172	39198	39164	39202

Burst type	*32 beats (adr inc.)*			*32 beats (no adr inc.)*		
Idle cycles	16	8	4	16	8	4
Signals	**Number of signals transitions**					
Inputs	483	481	477	313	309	311
Flip-flops	446	442	442	448	440	448
Internal	1012	1010	1010	1020	1004	1018
Outputs	313	311	311	315	309	315
HADDR	5528	5550	5518	220	216	216
HWDATA	39494	39486	39434	39448	39432	39440

Burst type	*16 beats (adr inc.)*			*16 beats (no adr inc.)*		
Idle cycles	16	8	4	16	8	4
Signals	**Number of signals transitions**					
Inputs	859	869	869	537	547	545
Flip-flops	744	744	754	750	758	754
Internal	1756	1772	1794	1774	1796	1794
Outputs	535	537	545	541	547	545
HADDR	5902	5884	5918	490	476	476
HWDATA	40102	40038	40060	40090	40048	39982

Burst type	*8 beats (adr inc.)*			*8 beats (no adr inc.)*		
Idle cycles	16	8	4	16	8	4
Signals	**Number of signals transitions**					
Inputs	1527	1657	1663	1019	1017	1025
Flip-flops	1662	1388	1392	1658	1390	1398
Internal	3376	3356	3386	3364	3362	3394
Outputs	1157	1017	1023	1153	1019	1027
HADDR	6618	6736	6750	1112	1154	1106
HWDATA	42298	41188	41092	42330	41168	41100

Table 5. Total normalized switching activity

	ADDR increment			*no ADDR increment*		
Idle cycles	16	8	4	16	8	4
Burst type	**Total normalized switching activity**					
128 beats	1	1	1	0,88	0,88	0,88
64 beats	1,01	1,01	1,01	0,89	0,88	0,89
32 beats	1,03	1,03	1,02	0,91	0,90	0,90
16 beats	1,06	1,06	1,06	0,94	0,93	0,93
8 beats	1,16	1,14	1,13	1,02	0,99	0,99

Table 4 reports the switching activity of HADDR and DATA bus lines, arbiter input signals (HSELx, HWRITE, HTRANS, HSIZE, HBURST, HRESET, HMASTER, HMASTLOCK), arbiter output signals (HREADY, HREAS HSPLIT), arbiter internal nodes and arbiter flip flops.

The arbiter code has been written in VHDL at RTL level and implemented on FPGA Altera (1385 equivalent gates and 24 FlipFlops on a MAX7000 device, 591 equivalent gates and 24 FlipFlops on a ACEX 1K device) and on an ASIC (237 equivalent gates and 24 FlipFlops with a STMicroelectronics library). The number of gates of the arbiter is low, therefore great part of power is assumed to be dissipated by the address, data and control lines (arbiter inputs and outputs) of the bus.

Table 5 reports total (ADDR + DATA + control lines) switching activity normalised to the case of 128 beats bursts .The switching activity dependence with inter-burst idle length is not relevant. Conversely, switching activity increases decreasing the burst length.

The results shown in Tables 1-5 are useful to evaluate the bus performances in the different cases. If the high priority master uses short bursts the low priority master waiting time decreases but switching activity increases.

A good compromise seems to be a burst length of 32 beats with 4 inter-burst clock cycles. The waiting time is half with respect to the 128 beats case, with just a 2,6% increment of switching activity.

In the burst transmission the address is incremented by the master at each beat following the AMBA AHB specifications. A smart slave can calculate the value of the address with just the address of the first beat.

The switching activity have been evaluated and reported in Tables 4 and 5 in the two cases: the ADDR bus lines are incremented (adr inc.) and the ADDR bus lines are not incremented (no adr inc.). A switching activity reduction of about 10% can be reached if the ADDR bus lines are not incremented.

The CPU time required for a VHDL simulation of 4000 clock cycles is about 27 secs on a Pentium II at 400MHz.

3.2 Arbitration algorithms

A second set of SystemC simulations have been performed to verify the effect of different bus arbitration algorithms (2-5 n Sect.2) on bus performances and power consumption. Two types of traffic generators have been used:

HIGH) intensive bus use: random sequence of 8 or 16 beats bursts (with equal probability) followed by a random idle period with gaussian distribution (mean 7 clock cycles, standard deviation 3 clock cycles)

LOW) infrequent bus use: random sequence of single or 4 beats bursts (with equal probability) followed by a random idle period with gaussian distribution (mean 12 clock cycles, standard deviation 3 clock cycles)

The simulated architecture consists of a default master, 2 masters (M1 and M2) with a traffic of type HIGH, and 3 masters (M3, M4 and M5) of traffic type LOW. The priority order is: M1 high … M5 low.

With this type of bus requests, the bus is almost always used by the masters: this is the case in which the effect of the arbitration algorithm on bus performances is relevant.

Table 6 reports the results of the SystemC simulations for each master and for the different arbitration algorithms:

- the maximum waiting time in clock cycles
- the average waiting time in clock cycles
- the percentage of bus use.

The average values of waiting time is acceptable for the four algorithms, but the maximum value is not acceptable for the algorithms without waiting time control: algorithm (2) penalizes low priority masters, algorithm (5) penalizes long length bursts.

Short Job First with waiting time control optimises the bus use: the default master takes the bus only 0.3% of the time.

The switching activity of the bus signals have been evaluated during the SystemC simulations. Table 7 reports the bus switching activity normalized to the switching activity obtained with the priority algorithm (2).

The arbitration algorithm have a strong influence on the switching activity of the control signals.

The Short Job First with waiting time control (5) algorithm allows a relevant switching activity reduction, about 22%. This is due to the more efficient use of the bus obtained with this algorithm: the default master controls the bus only for 0.3% of the time. The bus goes under the control of one master directly to another without going under the control of the default master. This fact reduces the commutations of the control signals due to the change of the master controlling the bus.

In conclusion, the Short Job First with waiting time control shows the best performances in terms of low waiting time and low switching activity.

The CPU time required for a SystemC simulation of 54000 clock cycles is about 8.2 secs on a Pentium II at 400MHz. Comparing the SystemC and VHDL simulations reported in this paper, a speed up of about 50 times is obtained with SystemC simulator. The real speed up is higher considering that the architecture simulated in SystemC is more complex: 5 masters against 2 and the arbitration algorithm is more complex.

4. CONCLUSIONS

In this paper a SystemC and a VHDL clock accurate model of the AMBA AHB bus are shown. The models have been used to evaluate the performances of the bus with different methodologies of bus use and arbitration algorithms. A reduction of bus power dissipation of more than 22% can be used applying the bus use techniques and arbitration algorithms shown in this work.

5. REFERENCES

[1] T. Grotker, S. Liao, G. Martin, S. Swan: "System design with SystemC" Kluwer Academic Publishers, 2002.

[2] Open SystemC Iniative (OSCI), SystemC documentation: http://www.systemc.org, 2001.

[3] S.Osborne, A.T. Erdogan, T.Arslan, D.Robinson, "Bus encoding architecture for low-power implementation of an AMBA-based SoC platform", Computers and Digital Techniques, IEE Proceedings- , Volume: 149 Issue: 4 , July 2002, pp152 -156

[4] Bertozzi, D.; Benini, L.; Ricco, B.; "Energy-efficient and reliable low-swing signaling for on-chip buses based on redundant coding", Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on , Volume: 1 , 26-29 May 2002, pp I-93-96

[5] Wei-Chung Cheng; Pedram, M.; "Memory bus encoding for low power: a tutorial", Quality Electronic Design, 2001 International Symposium on, 26-28 March 2001,Page(s): 199 –204

[6] Pedram, M.; "Power optimization and management in embedded systems", Design Automation Conference, 2001. Proceedings of the ASP-DAC 2001. Asia and South Pacific , 30 Jan.-2 Feb. 2001, pp 239 –244

[7] Henkel, J.; Lekatsas, H.; "A2BC: adaptive address bus coding for low power deep sub-micron designs", Design Automation Conference, 2001. Proceedings , 18-22 June 2001, Page(s): 744 -749

[8] Narayanan, U.; Ki-Seok Chung; Taewhan Kim; "Enhanced bus invert encodings for low-power ", Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on , Volume: 5 , 26-29 May 2002, Page(s): V-25 -V-28 vol.5

[9] ARM: "AMBA specification," rev. 2, May 1999.

[10] Caldari, M.; Conti, M.; Coppola, M.; Curaba, S.; Pieralisi, L.; Turchetti, C.; "Transaction-level models for AMBA bus architecture using systemC 2.0", Conference DATE 2003 , March 3-7, 203, pp. 26 –31

[11] Caldari, M.; Conti, M.; Crippa, P.; Orcioni, S.; Solazzi, M.; Turchetti, C.; "Dynamic power management in an AMBA-based battery-powered system", Conference ICECS 2002,

Table 6. Bus performance for different arbitration algorithms

Master #	M1	M2	M3	M4	M5	default
Traffic type	High	High	Low	Low	Low	-
	Priority					
Max wait (clock cycles)	15	17	83	481	602	-
Average wait (clock cycles)	0.46	0.46	4.74	5.88	6.40	-
Bus use (%)	27.7	28.5	9.0	10.2	9.8	14.8
	Priority with waiting time control					
Max wait (clock cycles)	26	40	89	96	98	-
Average wait (clock cycles)	0.47	0.75	5.68	5.38	5.9	-
Bus use (%)	32.0	30.4	10.4	11.2	10.4	5.6
	Short Job First					
Max wait (clock cycles)	85	494	20	24	22	-
Average wait (clock cycles)	0.80	1.89	1.45	1.98	1.84	-
Bus use (%)	30.3	29.5	10.6	9.9	9.7	9.9
	Short Job First with waiting time control					
Max wait (clock cycles)	82	95	32	41	41	-
Average wait (clock cycles)	1.25	1.49	1.98	2.63	2.78	-
Bus use (%)	33.5	32.6	12.1	10.7	10.7	0.3

Table 7. Normalized switching activity

	Normalized Switching activity		
Arbitration algorithm	ADDR	DATA	CONTROL
Priority	1	1	1
Priority with wait control	1.00	0.97	0.85
SJF	1.03	0.98	0.86
SJF with wait control	1.02	0.95	0.78