Design and Performance Analysis of A Reconfigurable Arbiter

Yu-Jung Huang, Ching-Mai Ko, and Hsien-Chiao Teng

Department of Electronic Engineering
I-Shou University,
Kaohsiung, Taiwan, ROC
email: yjhuang@isu.edu.tw

Abstract: - This paper presents the design and performance analysis of an arbiter with a hybrid arbitration algorithm. The hybrid arbitration algorithm contains static fixed priority algorithm in conjunction with dynamic algorithm to gain better system performance is described. The performance analysis for the various combinations of the arbitration algorithms under different traffic loads is simulated. The results indicate a better performance can be achieved as compared with the traditional arbitration assignment scheme. Based on the performance analysis, the hybrid arbitration can be custom-tuned to meet the design requirements. The implementation of the arbiter with hybrid arbitration scheme for system on chip applications is also explained. The reconfigurable arbiter was implemented by FPGA and synthesized by Synopsys Design Complier with a TSMC 0.18 μm cell library. In addition, the power analysis of the reconfigurable arbiter at various arbitration states is reported. The reconfigurable arbiter can be custom-tuned to obtain high bandwidth utilization, low latency, and power effective for on-chip bus communication.

*Key Words:-*Arbiter, Reconfigurable, System-on-chip, Arbitration Algorithm, FPGA, AMBA

1. Introduction

A typical System-on-Chip (SOC) design contains many heterogeneous cores linked together with sophisticated on-chip bus communication architectures. The on-chip bus communication architecture determines the way these heterogeneous functional units are exchanging and synchronizing data and has a great impact on the system's performance [1]. The SOC design paradigm relies on well-defined interfaces and reuse of intellectual property (IP). Because more and more IPs are integrated into the design platform, the amount of communication between the IPs is on the increase and becomes the source of the performance bottlenecks. The arbiter plays a very important role to manage the resource sharing on the SOC platform. On-chip bus communication is one of the critical components in a SOC platform. An efficient on-chip communication system has to satisfy the interface behavior of each IP block integrated within the complex SOC. With the increasing number of system components in SOC design, it becomes that an efficient arbiter is one of the most critical factors for high system performance. The conventional bus arbitration algorithms, the static fixed priority and the round robin, show several drawbacks on bus communication such as bus starvation [2].

Most existing buses have their own particular bus protocol. However, currently there exists no system bus standardization and communication architectures defined by commercial standards are widespread and available in the market. For example, the PI-Bus [3] of OMI, the AMBA bus [4] of ARM, the FISPbus [5] of Mentor Graphics, the CoreConnect of IBM [6], the SiliconBackplane of Sonics [7], the Wishbone of Silicore [8] and others. The CoreConnect and AMBA make use of a fixed priority arbiter. Lotterybus defines an arbitration method that does not presume any fixed communication topology [9]. Silicon Backplane uses an arbitration method by means of TDMA-based arbitration. Based on the AMBA AHB protocol and a more complex interconnection matrix, the Multi-layer AHB is a different realization of the bus architecture, which enables data transfers between several masters and slaves in a system [10]. Although the arbitration protocol is fixed, the choice of an arbitration scheme is usually depending on the application requirements.

Arbiter performances are extremely important in a platform-based design. System Level analysis of arbiter performances gives important information for the analysis and choice between different

architectures driven by functional, timing and power constraints of the System-on-Chip. Lahiri et al. use the performance analysis technique presented in [11] which can be efficiently analyzed to determine system critical paths, average processing time, number of missed deadlines, etc. Massimo Conti et al. [12] present the effect of different arbitration algorithms and bus usage methodologies on the bus AMBA AHB performances in terms of effective throughput and power dissipation. The arbitration algorithms used in their analysis are first priority and short job first algorithms. Their results indicate the arbitration algorithm have a strong influence on the switching activity of the control signals. Francesco Poletti et al. [13] analyze the impact on multiprocessor SOC performance of different bus arbitration policies under different communication patterns, showing the distinctive features of each policy and the strong correlation of their effectiveness with the communication requirements of an application.

In this paper, the implementation of a reconfigurable arbiter with hybrid arbitration algorithm is proposed. By implementing an efficient arbitration algorithm the system performance can be tuned to better suite the applications. This paper presents the design and implementation of an arbiter with a reconfigurable hybrid arbitration algorithm. The rest of this paper is organized as follows. An overview of AMBA on-chip bus is presented in the next section. The architecture design of the reconfigurable arbiter is presented Section 3. The performance analysis methodology is described in Section 4. The simulation and implementation results are provided in Section 5 and Section 6, respectively.

2. Overview of AMBA On-Chip Bus

The most frequently used on-chip interconnect architecture is the shared medium arbitrated bus, where all communication devices share the same transmission medium. The bus used in the SOC platform requires an arbitration process since multiple components connected to it can act as masters and hence initiate a transaction. The Advanced Microcontroller Bus Architecture (AMBA) is an open System-on-Chip bus protocol for high-performance buses on low-power devices. Three distinct buses are defined within the AMBA specification, including the Advanced High-performance Bus (AHB), the Advanced System Bus (ASB), and the Advanced Peripheral Bus (APB) [4]. Fig.1 shows the AMBA AHB Bus Arbiter/Decoder system. The APB is meant for connecting off-chip peripherals, whereas ASB and AHB are basically designed for on-chip interconnection between microprocessors, memory and peripheral modules. The standard AHB implementation provides fast communication between several master and slave modules; it is capable of pipelined operations, burst transfers and split transactions. There is a central arbiter for granting the masters access to the bus and a decoder which selects the active slave.

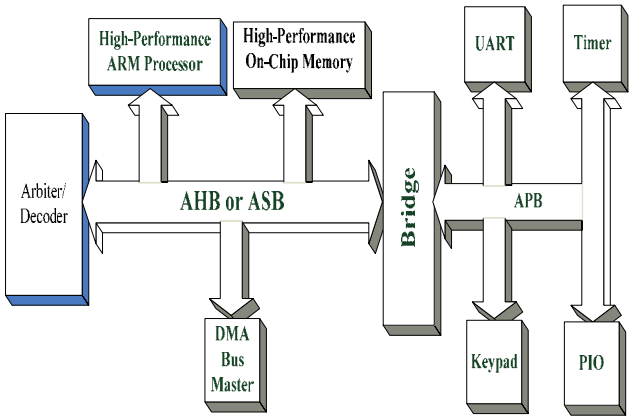

Fig. 1 A typical AMBA architecture

The AMBA AHB Bus Arbiter/Decoder contains sixteen programmable address registers- one address register per slave. These registers hold the base address and size of each slave's address space. The sixteen AMBA bus masters are master 0 through master 15. The connection between master and slave is realized by multiplexors instead of e.g. tri-state busses. The interconnections of multiplexor, decoder and arbiter are illustrated in Fig. 2.

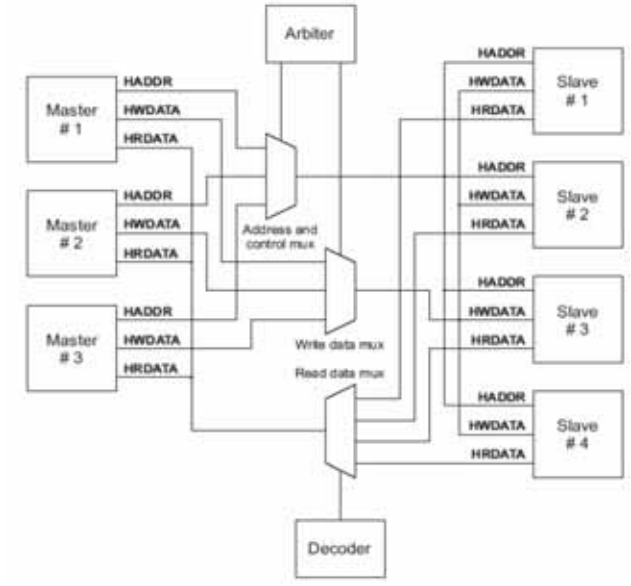

Fig. 2 Interconnection network for the AMBA AHB

The arbiter monitors the AMBA bus for requests and chooses the master with highest priority request as the next AMBA bus transaction master. A bus master asserts a LOCK signal to the bus arbiter to make multiple transactions indivisible. The arbiter must observe the state of the LOCK signal from the currently active bus master before granting access to another bus master. If there are no requests, the default master is chosen as the master to drive the next AMBA Bus transaction. The decoder block monitors the transaction address and decodes it by matching it to the slave base addresses stored in the registers. If the decoder encounters an address that does not match any of the slave address ranges, the default slave is chosen as the transaction target.
The transfer can be a burst transfer in AMBA. Bus burst operation may be of arbitrary length and may be broken down into smaller packets. The arbiter may constrain burst lengths to meet critical interrupt. Two major categories of AHB burst operation are incrementing burst and wrapping burst. For incrementing burst, the address for each transfer is incremented by the transfer size. For A four-beat wrapping burst, it indicates that this is a burst transfer of length 4 with address wrapping. The transfer size can be 8, 16 … to 1024 bits.
For RETRY, the arbiter uses the normal priority scheme. For SPLIT, the arbiter adjusts the priority scheme so that any other master requesting the bus will get access even if it is a lower priority. The SPLIT response causes the arbiter to grant other masters the use of the bus. For SPLIT, the arbiter adjusts the priority scheme so that any other master requesting the bus will get access even if it is a lower priority. The arbiter must be informed when the slave can have the data available. The arbiter re-grants these masters. A higher priority master will be granted for retry. A master may be granted the bus several times before it finally completes the transfer. Request and acknowledge handshake signals support the arbitration signaling for each bus master.

In summary, the AHB is a pipelined system backbone bus, designed for high-performance operation. The bus supports multiple bus masters, all of which communicate in a unified manner with slave devices. It can support up to 16 bus masters and slaves that can delay or retry on transfers. It consists of masters, slaves, an arbiter and an address decoder. It supports burst and split transfers. The address bus can be up to 32 bits wide, and the data buses can be up to 128 bits wide. The arbitration scheme is centralized. Masters must request the bus using a centralized arbiter; the arbitration protocol for each master is strictly defined, but the arbitration priority is left to the designer. By specifying the arbitration protocol rather than the arbitration scheme itself, all decisions about priorities becomes a system design issue to suit the application constraints.

3.Reconfigurable Arbiter Architecture

The AMBA uses conventional fixed priority arbiter. Fig. 3 shows the architecture of the reconfigurable arbiter presented in this work. As shown in Fig. 3, this reconfigurable arbiter can serve up to a maximum 16 masters and all of them are divided into four groups F1-F4 for the first level competition. According to different arbitration algorithms assigned in F1 ~ F4 blocks, each block will select one master from four input masters for the second level competition. The final granted master will then be determined by the arbitration algorithm chosen in block 5. In Fig. 3, each group (F1 ~ F5) can be reconfigurable to adapt a specific dynamic algorithm. For example, either one of the round-robin algorithm, random access algorithm or first-come-first-serve algorithm can be combined together to decide the functionality of the reconfigurable arbiter.
With the reconfigurable functionality, it can assign different arbitration algorithms for specific groups of masters. In the following, each block (F1 ~ F5) will be assigned a number to denote the status of the reconfigurable arbiter. The notation 1, 2, 3, and 4 are used to represents fixed priority, round robin, first come first serve and random access algorithm, respectively. For example, the arbitration state (12141) denotes the fixed priority algorithm is assigned to functional blocks F1, F3 and F5, round robin algorithm is assigned to functional block F2,

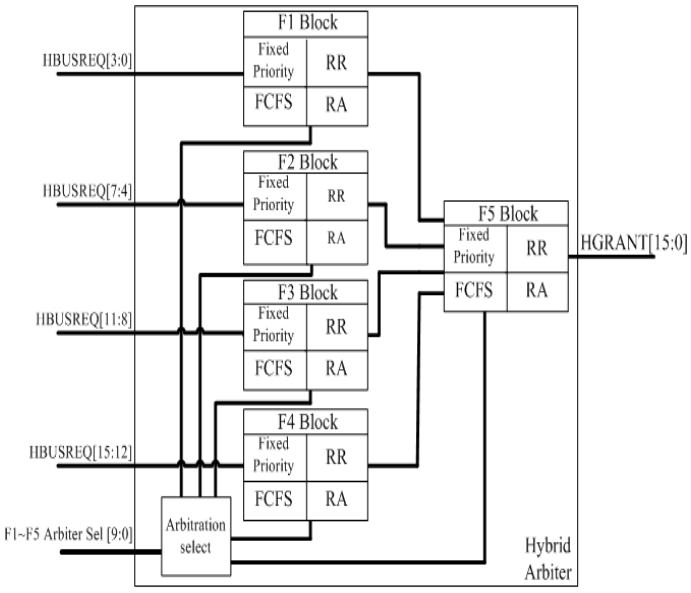

Fig. 3 Architecture of the reconfigurable arbiter

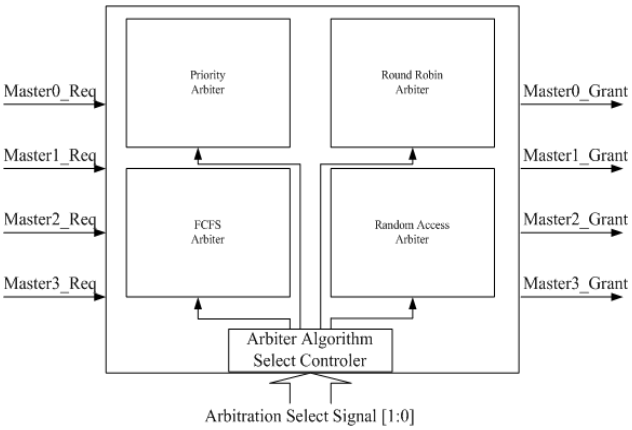

Fig 4. Functional block of reconfigurable arbitration algorithms

and random access algorithm is assigned to functional block F4, respectively.

The function of reconfiguration controller is shown in Fig. 4 and briefly described as follows. Each block F1 to F5 shown in Fig. 3 can be assigned different algorithms by the arbitration selection signal. The arbitration selection signal is user defined and can be used to change the algorithmic states of each block F1 to F5. Table 1 listed the reconfiguration input to blocks F1 to F5. Various arbitration algorithms can be assigned based on the choice of the reconfiguration input set by the reconfiguration controller. When the choice is "00", a fixed priority algorithm is assigned at the designated blocks. "01" is round robin algorithm, "10" is first come first serve algorithm, and "11" is random access algorithm, assigned to the designated blocks.

Table 1 The controller signals of the reconfigurable arbiter

Input	Arbitration Selection Algorithm
00	Enable Fix-Prtority Arbitration
01	Enable Round Robin Arbitration
10	Enable FCFS Arbitration
11	Enable Random Access Arbitration

4. System Performance Analysis

Since the arbitration efficiency is specific to the input stimuli used, it is important to characterize the traffic pattern in terms of parameters that are not tied to any specific sequence of input stimuli. The efficiency of the hybrid arbitration algorithm is examined by a proposed system performance analytical module.

The simulation strategy for performance analysis of the arbiter with hybrid arbitration schemes is divided into two parts: Traffic Pattern Generation (TPG) and Arbitration Simulator Generation (ASG) as shown in Fig. 5. The traffic patterns are generated based on the statistic distribution such as Bernoulli, Binomial, equilikely, Geometric, Pascal, Possion, Uniform, Exponential, Erlang, Normal, lognormal, Chisquare … etc. The simulations in this paper are based on the TPG which assign various distributions to indicate the data amount and bus request time for each master. For ASG, a behavior bus arbiter model based on Fig. 3 is used to take into account the effect of the shared bus communication architecture on system performance. The hybrid arbitration behavior including the fixed priority, round robin, first come first serve, and random access are implemented using C language and can be reconfigurable to obtain various arbitration states.

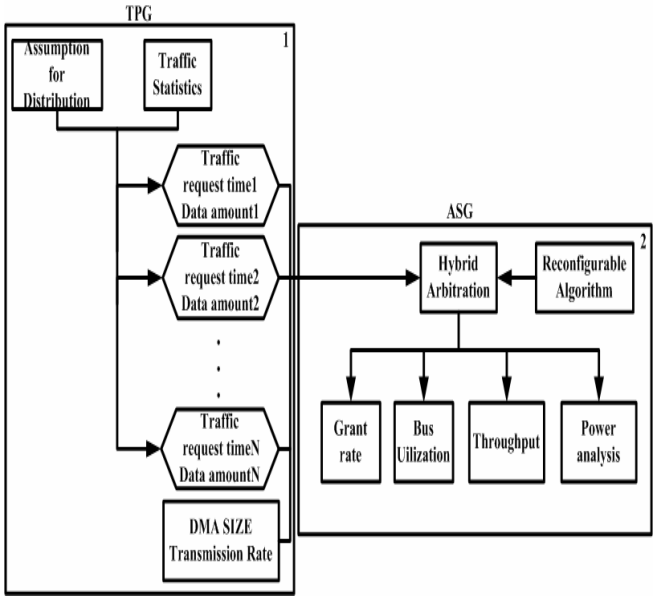

Fig. 5 System performance analytical module

The status parameters and the output of the system performance analysis are described in details as follows:

a. Waiting time: Wait time measures the time that a transaction has waited since it made a bus request. A long wait time implies a higher likelihood due to the need to prevent a master from being starved. Conversely, a short wait time implies a lower likelihood. The wait time is incremented at the end of an arbitration cycle if

the request is not acknowledged. The wait time is reset when a request is granted.

b. Grant rate: A bus request signal needs to be generated by the master, and the bus can be accessed upon reception of the bus grant signal from the arbiter. A proper algorithm for arbitrating a bus grant among a plurality of master devices for access to a shared bus is very important for system performance. The master grant rate is obtained by calculating the ratios of the number of granting bus utilization to the number of requesting a data transfer by one of the master devices.

c. Throughput: Throughput is the number of bits transmitted per second through a communication system. Throughput varies over time with traffic and congestion. Throughput is based on many factors, such as the total number of masters, the data transfer size, …etc. A point that is often overlooked when testing for throughput performance is the actual arbitration algorithm being used. For present study, the reconfigurable arbiter is very flexible, it can define different arbitration algorithm and able to deal with variations in performance.

d. Power analysis: In a System on Chip, the bus lines capacitance has an order of magnitude bigger than transistor gate capacitances. Many techniques, especially of bus encoding have been studied to reduce bus switching activity [141. 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 arbitration algorithms, together with the requested functional and timing specifications of different implementation technologies.

The reconfigurable hybrid arbitration algorithms can be achieved by assigning different arbitration algorithms into functional blocks F1 – F5. To study the effects of the various reconfigurable arbitration schemes, the performance analysis are further simulated based on various permutations of arbitration algorithms for blocks F1 ~ F5 under the same traffic distribution pattern. For the evaluation of relative performances of different configuration states, the average waiting time, bus granted rate, throughput, bus utilization can then be calculated. An example of the system initial condition and the results of the data transfer for the arbitration state (23235) is listed at Table 2.

Table 2 Evaluation of System Performance Analytical Module

Bus request time distribution: Bernoulli probability: 0.5 Data transfer distribution: Equilikely mean:12 variance:6.67 Arbitration algorithm :F1:1,F2:1,F3:1,F4:1,F5:1 Data transfer width:32bits Transfer cycle:16 M0_data:1.75KB、M1_data:1KB、M2_data:1.875KB、M3_data:1KB、M4_data:1.375KB、M5_data:1.125KB、M6_data:1.5KB、M7_data:1KB、M8_data:1.125KB、M9_data:1.125KB、M10_data:1.875KB、M11_data:1.875KB M12_data:1.125KB、M13_data:1.25KB、M14_data:1.125KB、M15_data:1.25KB		
	Average waiting time	Time of completing data transfer
Master0	1211.857178	2187
Master1	3819.625000	4014
Master2	2065.800049	2893
Master3	1754.500000	1897
Master4	226.500000	1035
Master5	3795.611084	4320
Master6	5942.750000	6226
Master7	3370.562500	4179
Master8	3600.166748	5359
Master9	519.500000	664
Master10	2703.766602	3723
Master11	5042.700195	5646
Master12	3241.000000	3394
Master13	5663.000000	6251
Master14	1188.500000	1333
Master15	4522.500000	4704

5. Simulation Results

.

In our work, we use performance analytical module described in previous section as a basis for evaluating communication events at different arbitration states. The simulation results can be analyzed to examine the impact of individual (or groups of) communication events on the system's performance and identify the effects of arbitration algorithm on the system performance. Since bus loading limits the system performance, the traffic pattern will generate different statistic distribution to represent the data amount and bus request time issued from the individual master. Therefore, in addition to identifying the correlation between the communication events and states of arbitration algorithm, we can also apply different traffic pattern to correlate the system performance to arbitration state and data it is processing. If an analysis of the

simulation result reveals that the occurrence of a critical data-transfer is highly correlated to a specific arbitration algorithm in the behavior of the component executing the data transfer, the assignment of the specific algorithm might be used as a predictor for the criticality of the data transfers generated by the component. The following example examines some tradeoffs in designing these predictors.

The grant rate and average waiting time are simulated based on 1024 permutations of arbitration algorithms for blocks F1 ~ F5 under the same traffic distribution pattern. The request time adopts Bernoulli distribution and the probability is 0.5. The amount of traffic data adopts Equilikely distribution, the mean is 12 and the variance is 6.67. The data transmission rate is 32 bit/s, and the period is 16 clock cycles. Results of the software modeling for average waiting time at various combinations of the arbitration algorithm in function block F1 to F5 are reported in Fig. 6. The x-axis depicts all the possible arbitration combinations where 1, 2, 3, and 4 represent fixed priority, round robin, first come first serve and random access algorithm, respectively. From Fig. 6, it shows that the hybrid arbitration algorithm has a predictable average waiting time. So, user can assign the optimal combination for the application specific tasks. It also shows that the hybrid arbitration scheme with a proper assignment of the arbitration algorithm to F's functional blocks can achieve better performance as compared to traditional arbitration schemes such as fixed priority and round robin.

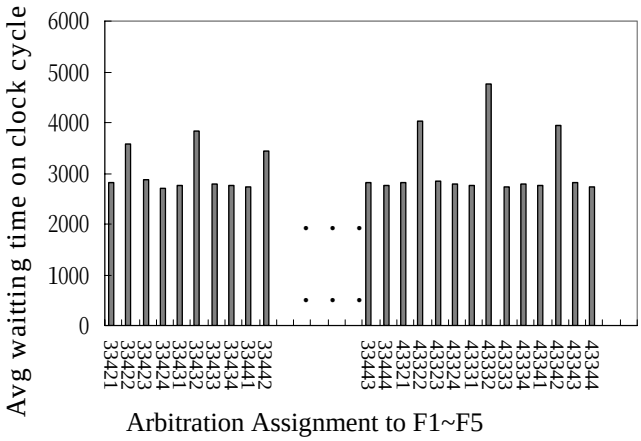

Fig. 6 Average waiting time under hybrid arbitration

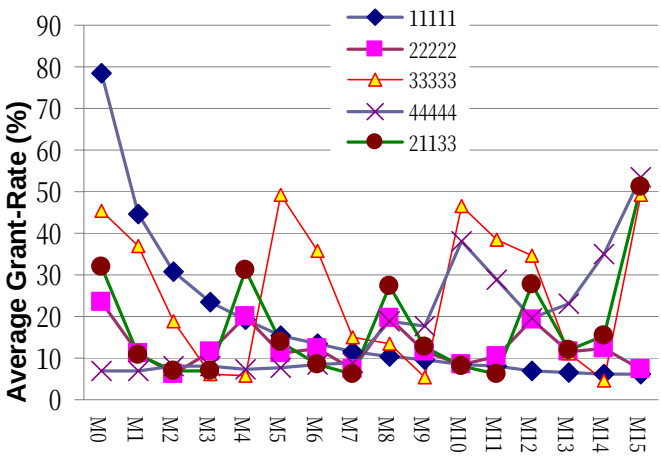

Fig. 7 Grant rates of Master 1 to 15 at different arbiter configuration states

The grant rates of Master 1 to 15 at different arbiter configuration states are shown in Figs. 7. From the average grant rate shown in Fig. 5, the starvation occurs for low priority master (master 15) under fixed priority arbitration scheme. A lower priority master wanting to use a shared resource gets blocked when a higher priority master holds the resource. However, this starvation issue can be resolved by reconfiguring the arbitration state. As indicated in Fig. 7, master 15 can obtain 53 % grant rate with arbitration state 44444, 51% grant rate with arbitration state 21133, and 49% grant rate with arbitration state 33333, which is much better than the arbitration state 11111.

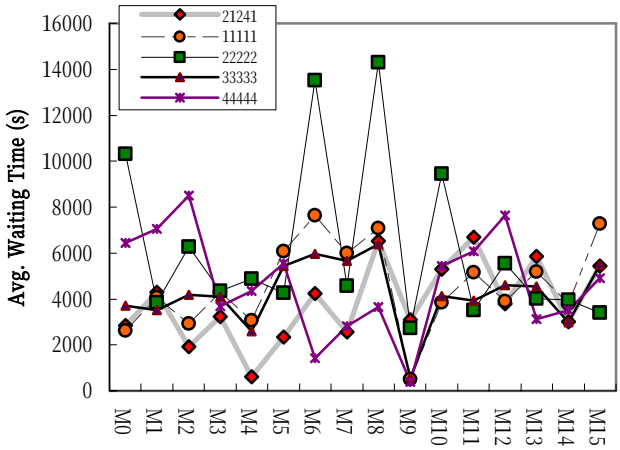

Fig. 8 Average waiting time of Master 1 to 15 at different arbiter configuration states

Average waiting time of Master 1 to 15 at different arbiter configuration states is shown in Fig. 8. As shown in Fig. 8, the average waiting time of arbiter configuration state 21241 is much shorter as compared with conventional round robin arbitration

scheme. The longest waiting time is obtained when the arbiter configuration is 22222. The arbitration state 21241 can gain the shortest average waiting time, which is better than the conventional fixed priority arbitration scheme.

Shared buses are very commonly used to facilitate communication between the various system components. In order to speed efficient transmission of larger bursts of data, bus protocols may also provide a direct memory access (DMA) or block transfer mode. A master is granted the right to use the bus for multiple bus cycles. Fig. 9 shows the system performance of the effect of block size on different arbitration configuration. The block size c16b32 denotes in the x-axis of Fig. 9 indicating the data transmission rate is 32 bit/s, and the period is 16 clock cycles. The worst case occurs when the arbitration configuration is set to be 43332 for F1~F5 block and the block size is chosen to be 16x32 bits/s. The best performance is obtained by 32314 assignment state and the block size is 32x64 bits/s.

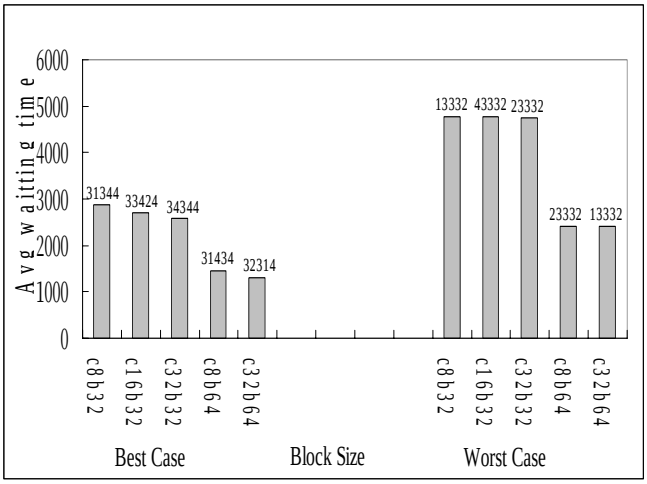

Fig. 9 Block size effect for various arbitration scheme

Simulation tests using normal distribution for data amount and exponential distribution for master request are conducted for 1000 times and results are average out. Based on 1024 permutations of arbitration algorithms for blocks F1 ~ F5 under the same traffic distribution pattern, Table 3 shows the simulation results of the grant rate and through put at various configurations. For the present case study, it indicates that the reconfigurable arbiter with 22221 configuration has the best bus utilization rate, 13431 configuration has the best average grant rate, 22221 configuration has the best throughput performance.

Table 3 Simulation results of the performance analysis at various configuration schemes

Arbitration configure	Avg. Bus Uilization (%)	Avg. Grant Rate (%)	Avg. Throughput (bit/s)
12314	78.582281	21.847687	25.10701
12413	78.70257	12.832688	25.144301
13431	78.708633	22.378313	25.146475
13411	78.741391	21.03675	25.156859
34341	78.879539	21.743875	25.200553
22224	81.990609	13.027	26.195668
12214	83.080711	16.205313	26.543143
12411	84.082711	18.282812	26.862924
12424	84.190094	17.3065	26.897707
12421	84.772875	18.18225	27.083648
22221	85.31818	12.873	27.257682
11111	78.558523	19.754125	25.09885
22222	60.248195	10.371563	19.24841
33333	75.091805	15.752938	23.990609
44444	73.05175	19.795437	23.339273

6. Implementation Results

To describe clearly the architecture of the proposed reconfigurable arbiter, we present the hardware architectures for fixed priority, round-robin, first come first serve, and random access, respectively. The functional block of the fixed priority is depicted in Fig. 10. The request signals of Master 0 (M0), Master 1 (M1), Master 2 (M2), and Master 3 (M3) are assigned to BUSREQ0, BUSREQ0, BUSREQ0 and BUSREQ0, respectively. Then, the priority order is assigned to M0 > M1 > M2 > M3, where M0 owns the highest priority. The granted signals Grant0, Grant1, Grant2, and Grant3 are for M0, M1, M2, and M3, respectively.

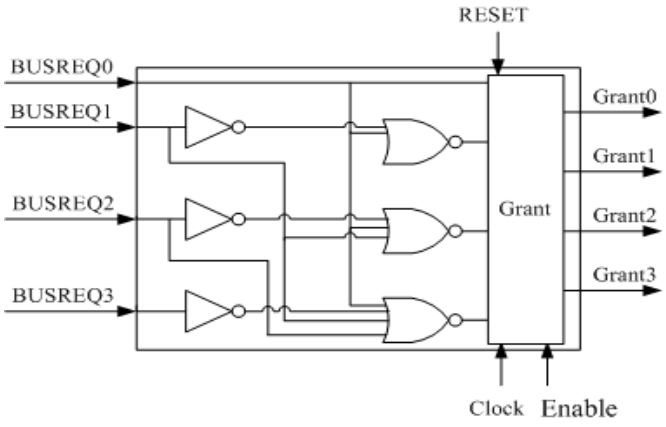

Fig. 10 Circuit Functional block of fixed priority algorithm

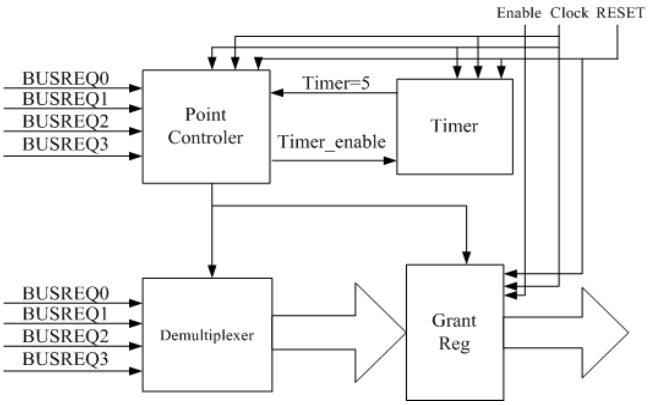

Fig 11. Circuit Functional block of round robin algorithm

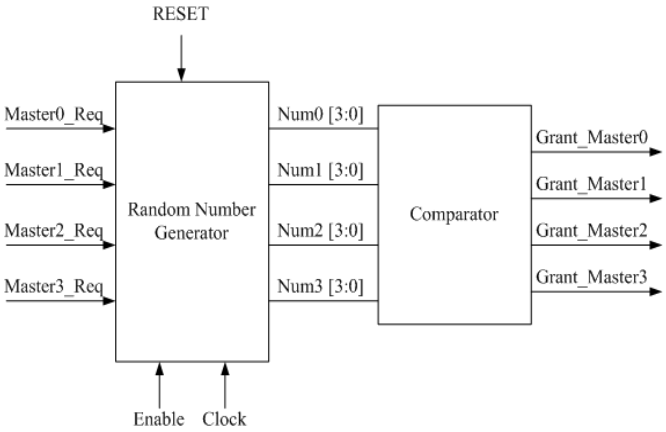

Fig 13. Circuit Functional block of ramdom access algorithm

The implementation of a round robin is shown in Fig. 11. A round-robin arbiter consists of a point controller, a timer, a demultiplexer, and a grant register to memorize the granted master. The point controller will point to each master to check the bus request signal. If the master requests the use of the bus, the address of the master will be demultiplexed and store at the grant register. The hardware of the FCFS algorithm is shown in Fig. 12. It contains a FIFO stack counter, a priority sorting controller, a FIFO buffer, and a grant register. The FIFO is full when the FIFO stack counter reaches a predetermined full value and the FIFO is empty when the FIFO stack counter is zero. When FIFO_write_enable signal is 0 , the data_out_f signal will retrieve the granted master from the FIFO buffer and stored it in the grant register.

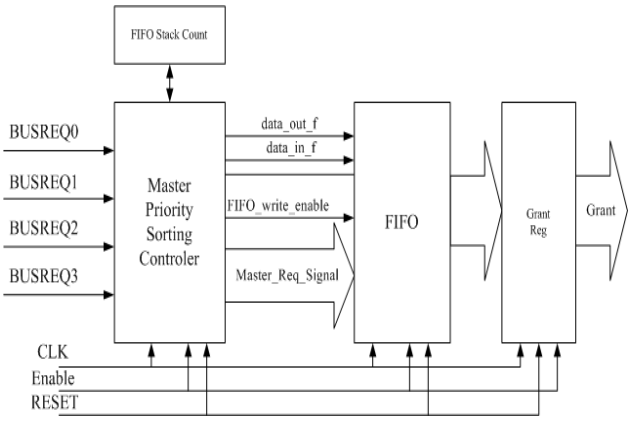

Fig 12. Circuit Functional block of FCFS algorithm

The random access algorithm is implemented using a random number generator and a comparator as illustrated in Fig. 13. The random number generator is implemented by linear feedback shift register approach. Each master will be assigned a random number. The master with the maximum random number is granted from the comparator output.

The hardware description language verilog is then used for digital circuit design including fixed priority, round robin, first come first serve and random access algorithms. The reconfigurable arbiter was synthesized using Altera's Quartus II 4.2, which is an integrated environment for logic design and synthesis. After finishing the verilog simulation, we download the program into EPF10K100ARC240-1chip. An 8051 microcontroller was used to drive the test signal into the chip. The arbiter can be reconfigured many times by modifying the program code for the 8051 microcontroller, and, thus, can be changed and updated real-time. The Tektronix logic analyzer is connected to the FPGA pins to check the functionality of the reconfigurable arbiter. Fig. 14 shows the reconfigurable arbitration state (12341) measured by the logic analyzer.

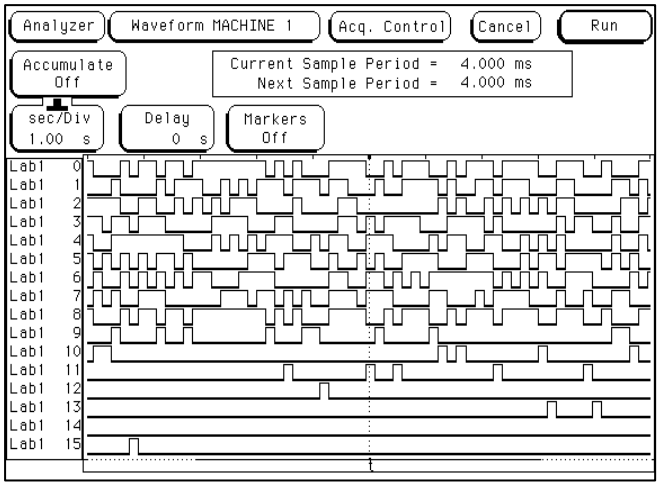

Fig. 14 Verification of 12341 configuration by logic analyzer

The synthesized verilog files are simulated in ModelSim. A Logic Synthesis System converts a description of the reconfigurable arbiter circuit into an interconnection of logic gates (a gate-level net-list). In this study, the RTL model of the reconfigurable arbiter is synthesized to gate-level using Synopsys Design Vision with TSMC 0.18 μm technology file. In order to provide a fast evaluation of the energy impact of various arbitration states, the switching activities are then used by Synopsys PrimePower for power calculation. The same RTL model is also used and mapped into Xilinx Vitrex and Altera Straitx GX development tools. Fig. 15 shows the synthesis and power calculation flow.

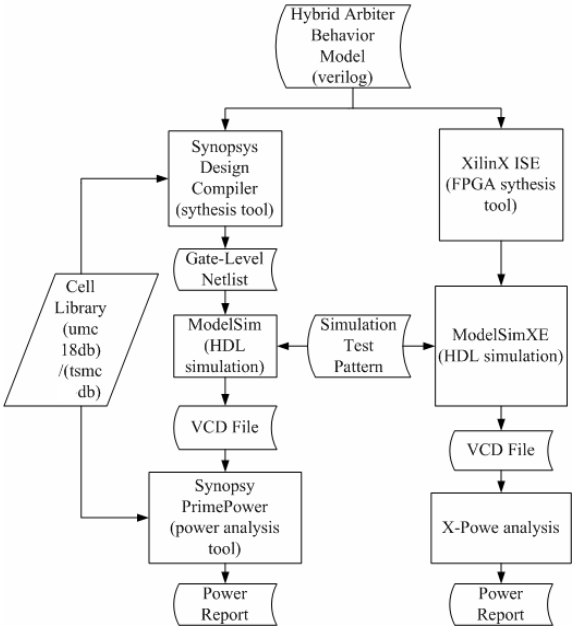

Fig. 15 Synthesis and power calculation flow

Table 4 shows the FPGA implementation results of the reconfigurable arbiter for single layer share bus system. Table 5 lists the area and power performance summary of the design complier report based on TSMC 0.18 μm technology file. The Synopsys Synthesis tool, Design Vision, is used to generate a synthesized net-list of the reconfigurable arbiter as shown in Fig. 16.

Table 4 FPGA Implementation Results

Xilinx VirtexE XCV2000e-BG560			Atera StratixGX EP1SGX25CF672C5	
Delay (ns)	Slice Flip Flops	4 input LUTs	Delay (ns)	Logic elements
16.758	657	1153	9.286	1108

Table 5 Design Complier Report

Combinational area (μm^2)	28956.04883
Noncombinational area (μm^2)	36467.35547
Net Interconnect area (μm^2)	882564.1875
Total cell area (μm^2)	65423.63672
Total area (μm^2)	947987.8125
Cell Internal Power (mW)	3.9
Net Switching Power (mW)	2.9415
Total Dynamic Power (mW)	6.8415
Cell Leakage Power (μW)	2.0263
NAND2X1 : 5.04 µm (height) x 1.98 µm (width)	

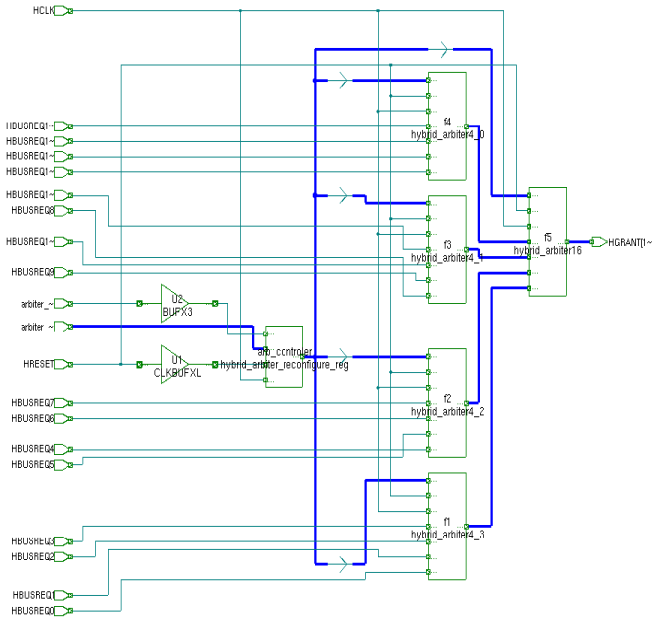

Fig. 16 Synthesized net-list of the reconfigurable arbiter

The power performance of the reconfigurable arbiter under various configuration schemes for TSMC 0.18 μm technology is shown in Fig. 17.

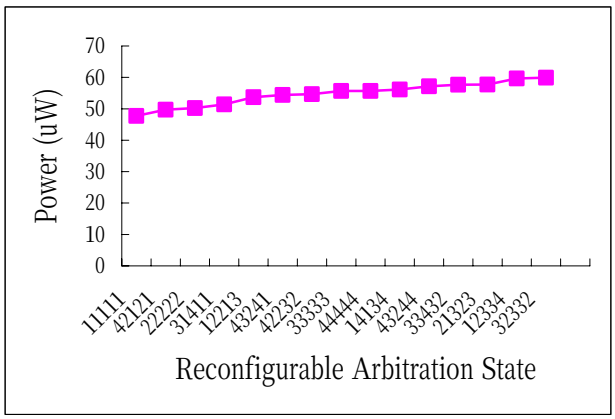

Fig. 17 Power performance at various reconfigurable arbitration states

Table 6 listed the power analysis of the reconfigurable arbiter obtained by PrimePower simulation at various arbitration states. It can be seen the maximum total power is 59.92 μW for arbitration state (32332) in the shared bus system. The minimum power is 47.73 μW for arbitration state (11111) configuration.

Table 6 Power Analysis Report

Arbitration State	Total Power
33432	57.66
32332	59.92
42232	54.69
21323	57.7
12213	53.72
43244	57.14
43241	54.41
12334	59.67
14134	56.13
31411	51.39
42121	49.73
11111	47.73
22222	50.24
33333	55.67
44444	55.67
Power (uW) 1: Priority 2: Round-Robin 3: FCFS 4. Random Access	

7. Conclusion

The arbiter with hybrid arbitration algorithm for single layer shared bus system is presented in this study. The simulation results not only provide performance analysis for the various combinations of the arbitration algorithms. The gate-level power analysis is also applied to explore power dissipation in various reconfigurable arbiter architectures. These results can be feed into the reconfigurable arbiter to obtain the optimal condition under different system workloads. The reconfigurable arbiter can be custom-tuned to obtain high bandwidth utilization, low latency, and power effective for on-chip bus communication. The results obtained show that the framework of the reconfigurable arbiter can be used to explore the space of possible configurations to evaluate the performance/power trade-off. In addition, the present study indicates the designers may implement a fixed-priority scheme or a more complex round-robin or adaptive arbitration mechanism, depending on the real time requirements.

Acknowledgments

This research was partially supported by National Science Council, R. O. C., under Grant NSC96-2221-E-214-068.

References

[1] K. Lahiri, A. Raghunathan, and G. Lakshminarayana, "LOTTERYBUS: a new high-performance communication architecture for system-on-chip designs", Proceedings of the 38th conference on Design automation, p.15-20, June 2001, Las Vegas, Nevada, United State

[2] YJ Huang, YH Chen, CK Yang, and SJ Lin, "Design and Implementation of A Reconfigurable Arbiter," 7th WSEAS Int. Conf. on SIGNAL, SPEECH AND IMAGE PROCESSING (SSIP '07), Beijing, China, 2007

[3] Open Microprocessor systems Initiative, DRAFT STANDARD OMI 324: PI-Bus Rev. 0.3d, 1994.

[4] Advanced RISC Machines Ltd (ARM). AMBA Specification (Rev.2.0), ARM IHI 0011A. http://www.arm.com/products/solutions/AMBASpec.html.

[5] http://www.mentor.com

[6] http://www01.ibm.com/chips/techlib/techlib.nsf/productfamilies/CoreConnect_Bus_Architecture

[7] "Sonics Integration Architecture, Sonics Inc." Available: http://www.sonicsinc.com.

[8] W. Peterson. Design Philosophy of the Wishbone SoC Architecture. In Silicore Corporation, 1999. Available: http://www.silicore.net/wishbone.htm.

[9] K. Lahiri , A. Raghunathan , and G. Lakshminarayana, " The Lotterybus on-chip communication architecture", IEEE Trans. on Very Large Scale Integration (VLSI) Systems, Vol. 14, No. 6, 2006, pp. 596 – 608.

[10] www.arm.com/pdfs/DVI0045B_multilayer_ahb_overview.pdf

[11] K. Lahiri , A. Raghunathan and S. Dey, "System-level performance analysis for designing on-chip communication architectures," IEEE Trans. Computer-Aided Design, vol. 20, June 2001, pp. 768–783

[12] M. Conti, M. Caldari, G. B. Vece, and S. Orcioni, C. Turchetti, "Performance analysis of different arbitration algorithms of the AMBA AHB bus", Design Automation Conference, 2004. Proceedings. 41st ,2004, pp. 618 – 621

[13] F. Poletti, D. Bertozzi, L. Benini and A. Bogliolo, "Performance Analysis of Arbitration Policies for SoC Communication Architectures", Design Automation for Embedded Systems, Vol. 8, Numbers 2-3, 2003

[14] G. Ascia, V. Catania, M. Palesi, A. Parlato, " Switching activity reduction in embedded systems: a genetic bus encoding approach", Computers and Digital Techniques, IEE Proceedings - Volume 152, Issue 6, 4 Nov. 2005, pp. 756 – 764