An Accommodative Adaptive Arbitration Algorithm for Maximum CPU Utilization, Fair Bandwidth Allocation and Low Latency

M. Nishat Akhtar, Othman Sidek
Collaborative Microelectronic Design and Excellence Center
Universiti Sains Malaysia, Nibong Tebal
Penang,14300, Malaysia
nishat_akhtar2000@yahoo.com

ABSTRACT: *An adaptive algorithm for fair bandwidth allocation, low latency and maximum CPU utilization is proved to be a promising approach for designing system-on-chip for future applications. Adaptive arbitration focuses on user oriented as well as system oriented performance therefore it is more advantageous then the other conventional arbitration algorithms for several reasons; these include fair bandwidth allocation among different masters, simple design and low cost over head. This article provides a comprehensive picture of research and developments in dynamic arbitration algorithm for masters according to the different traffic behavior. The papers published in standard journals are reviewed, classified according to their objectives and presented with a general conclusion.*

Keywords: Adaptive Arbiter, Fair Bandwidth, Low Latency, Real Time System, System-on-Chip

Received: 17 Febuary 2012, Revised 1 April 2012, Accepted April 2012

1. Introduction

In today's world symmetric multiprocessing has become an operable option for computing in the world of embedded systems and technology is blended with complex chips that incorporate multiple processors dedicated for specific computational needs. In order to realize this complex multiple system-on-chip in the environment of intellectual property based methodologies, the communication architecture plays a major role. In any system-on-chip, in order to solve the bus contention, arbitration algorithm plays a major role. The fairness property plays a very crucial role among the various criteria of arbitration algorithms to solve the bus contention. The performance of multiprocessors systems depends more on the efficient communication among processors and on the balanced distribution of computation among them, rather than on pure speed of processor. Since arbiters are invoked for every transfer on the bus, they are considered to be in the critical path of bus based communication architecture and must be designed with a great care [1]. An efficient contention resolution scheme is required to provide fine-grained control of the communication bandwidth allocated to individual processor and avoid starvation of low priority transactions [2].

Now a day's advanced super IC processors combined with high speed and performance which uses more and more heterogeneous cores have been integrated on a single chip [3]. The traditional bus based architecture cannot satisfy the communication requirement effectively between thousands of homogenous or heterogeneous IP's. Network-on-Chip (NOC) architecture can overcome the demerits of traditional bus based architecture. On the other hand, there are several disadvantage of NOC. First is regarding its physical size as it uses most of the space on the chip. It means, it deals with a heavy design size constraint on the nodes. The second disadvantage is its fixed framework layout of the network that means if a regular equidistant structure is chosen, the area can be lost because of some functional units. Third disadvantage is because of its fixed grid, the communication latency between two functional units becomes bigger due to non-linear data paths.

In symmetric multiprocessing the major architectural bottleneck is the internal bus which connects the processors and peripherals to the memory using an arbitrary network of shared channels [4]. In most cases, the bus bandwidth becomes a dominant barrier because of improper bandwidth allocation. To maintain the bus bandwidth in an efficient manner, the process of memory arbitration cannot be neglected as it is one of an essential factor for concurrent-computing. In an enhanced arbitration environment of SoC, the communication architecture should be fair enough to offer high performance to the wide range of masters according to their traffic behavior as the masters on a SoC bus may issue simultaneous requests. Thus, an arbiter is required to decide that which master should be granted for the bus access. Hence an arbiter should be designed in such a way that it suits the system by keeping high throughput and low starvation among the different masters.

2. Related Works

Many researchers focused on developing multi level arbitration scheme in order to achieve fair bandwidth allocation and to reduce the system latency. Laxmi Bhuyan [5] showed that if the priority to less frequent requests is given, then the probability of their acceptance is dramatically improved but at the same time there is a tremendous decrease in the bandwidth. V. Lakshmi et al. [6] proposed a novel arbitration scheme called time out arbitration, which ensures a degree of fairness for all the low priority requests in a given time slot but the amount of integrated circuits used in their design makes the cost of the device prohibitive for real time applications.

Enrico and Massimo [7] proposed a novel method of automatic synthesis of easily scalable bus arbiters with dynamic priority assignment strategies. They emphasized more on those arbitration mechanisms which can be implemented on silicon as a digital circuit, rather than getting concerned about how the selected arbitration policies can affect the performance of a multiprocessor system. The major disadvantage of common-bus multiprocessor system is the reduction of throughput caused by conflict between processors requiring access to the shared memory [8]. Ideally, throughput should increase directly with the number of processors but the bus contention diminishes this increase. There is a critical number above which the processors show no improvement and this critical number depends naturally, on the extent of bus used by the processors [9].

In order to design multiprocessor system according to their bandwidth requirement Yi Xu et al. [10] proposed an arbiter called an adaptive dynamic arbiter in which they designed a lottery bus algorithm approach where an arbiter can adjust the bandwidth proportion assigned to every processor automatically due to the situations of bus transactions aiming to reduce total task execution time. Compared with conventional architectures their architecture reduces the system latency but it does not allocate fair bandwidth to the processors.

In multithreaded environment efficient bandwidth allotment plays a major role because of the performance problem which may occur due to contention among application threads for the possession of the bus [4]. Alex Aravind [11] presented an algorithm which is a fully distributed software solution to the arbitration problem in multiport memory systems. His algorithm is purely based on First in First out (FIFO) and Least Recently Used (LRU) fairness criteria but the algorithm does not deal with fair bandwidth allotment to the different masters which may become a barrier to get a better performance.

Chien-Hua et al. [12] designed a real time and bandwidth guaranteed arbitration algorithm for system-on-chip bus communication in which RT_Lottery algorithm has been used to meet both hard real time and bandwidth requirements but in terms of fair bandwidth allocation it cannot compete with adaptive arbiter. Haishan Li et al. [13] proposed an algorithm called adaptive arbitration algorithm in which an arbiter can adjust priority automatically to provide the best bandwidth for different master according to their real time bus bandwidth needs. They showed that, it is possible to allocate fair bandwidth to a given set of processors with a very high degree of fairness.

An arbiter with fair bandwidth allocation and low latency seems to be of interest with many applications like real time computing system with critical requirements. Therefore, an attempt has been made to reduce the latency and to ensure high degree of fairness in terms of bandwidth among the processors designed according to their traffic behavior for real time computing system.

3. Classification of Masters on the Basis of Traffic Behavior

Chien-Hua et al. [12] designed a real time and bandwidth guaranteed arbitration algorithm for system-on-chip bus communication in which RT_Lottery algorithm has been used to meet both hard real time and bandwidth requirements. In their work following masters (processors) has been classified according to their traffic behaviors:

3.1 D_Type (D for Dependent)

D Type masters have no real time requirements and the next request is issued at the time depending on the finish time of the current request. In this case the interval time between two successive requests is the time from the issued time of the former to the finish time of the latter. Figure 1 shows an example. A 4 beat burst is generated by the traffic generator at cycle 2. The request is not granted until cycle 5 is finished at cycle 9. Suppose if the interval time is 10, then the next request will be issued only at cycle 19.

3.2 D_R Type (D for Dependent, R for Real-time)

This is same as D Type master; the only difference is that they have extra real time requirements. Figure 2 shows an example. R_{cycle} is the real time requirement of the master which is set to 10 cycles. It is shown in the figure that the request issued at cycle 2 has to be finished before cycle 12 represented by dotted lines. A real time violation occurs if the request is not completed before cycle 12.

3.3 ND_R Type (ND for Not Dependent, R for Real-time)

The issued time of a request from a ND_R Type of master is independent of the finish time of its previous request, and the interval time is the clock cycles between two successive requests. In figure 3, time interval assumed is 15. At cycle 17 second request is issued which directly depends on cycle 2 of the first request but not its finish time at cycle 9. In this case R_{cycle} is supposed to be smaller than the minimum possible interval time because the current request must be finished before the next request. It means that it is possible for the designers to assign tight real time requirements.

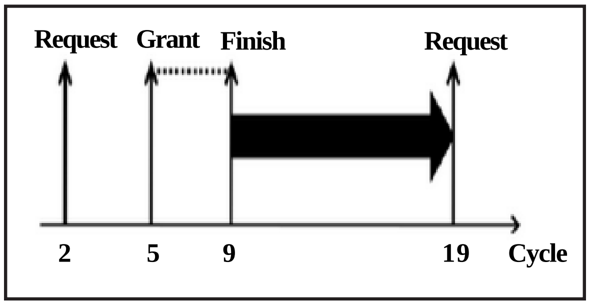

Figure 1. D Type Master

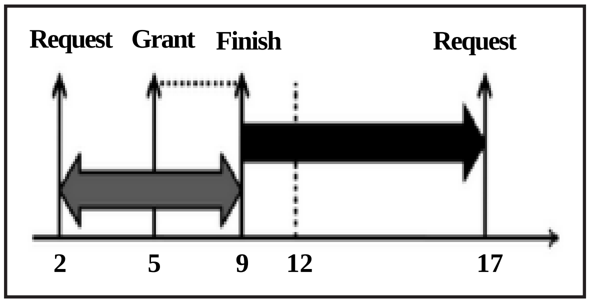

Figure 2. D_R Type Master

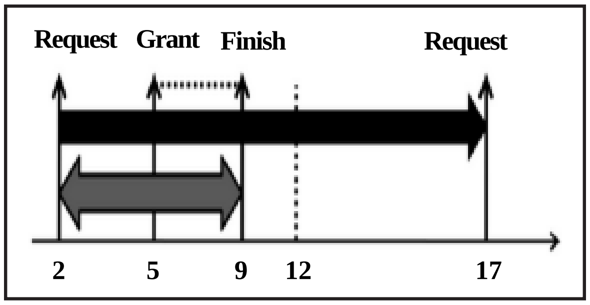

Figure 3. ND_R Type Master

Chien-Hua et al. [12] conducted an experiment where masters designed according to their traffic behavior are put on a bus. For each type, design of a heavy traffic master and a light traffic master has been made. Here, both M1 and M2 are Dependent (D) type masters, M3 is a D_R type master, M4, M5 are ND_R type master and over here the requests issued by M1 have larger beat numbers and shorter average interval than those issued by M2. It is clear that M1 generate a heavier traffic load to the bus than

M2. From table1 it is observed that the maximum bandwidth of each master is very different from each other because there are masters with heavy and light-traffic loads. The evaluated maximum bandwidth and the required bandwidth for each master are shown in the following table.

	M1	M2	M3	M4	M5
Maximum Bandwidth (%)	63	18	63	19	17
Required Bandwidth (%)	20	5	40	10	17

Table 1. Bandwidth Evaluation (Ref. Chien-Hua et al.)

It is observed that the total required bandwidth uses 94% of the total bandwidth.

4. Adaptive Arbitration Algorithm

Haishan Li et al. [13] proposed a novel arbitration algorithm which is known as Adaptive Arbitration (AA) algorithm. This algorithm is a dynamic priority algorithm. An arbiter adapting to this algorithm can adjust priority automatically to provide the best bandwidth allocation for different masters according to their real time bandwidth requirements. The experimental result shows that this novel algorithm meets both priority and equity in solving the bus bandwidth allocation. In a bus system the arbiter records the number of time each master has requested for the bus and the total time that all master have requested for the bus access. Using these two values the arbiter can calculate the bus access probability of the corresponding master by the division operation method. The priority weight of the master is decided by its probability of getting the bus access. A master who has the bigger weight owns the higher priority. It is unnecessary for an arbiter to recalculate all the probabilities and weights and to reorder the priority of masters when a new bus access request appears. The solution to this problem is to reduce the frequency of weight calculation and priority reordering [13]. The frequency here is defined as the adaptive cycle. The arbiter recalculates the weight and reorders the priority of the entire master when their total bus request arrives at the adaptive cycle. In adaptive arbiter with a two-level priority, the first level adopts adaptive arbitration algorithm and in the second level in order eliminate the starvation among all the masters, the static fixed priority algorithm has been adopted. The hardware in adaptive arbiter includes a public counter to record the total number of bus access requests. The value of public counter gets cleared all the time whenever it reaches the adaptive cycle. To record the number of bus access from each master there is a private counter dedicated to each master. Apart from all these there is a weight register and a left weight register which is assigned a weight value and is used to decide the priority of the corresponding master in the upcoming adaptive cycle. The higher priority is given to the master who has got a bigger weight. The left weight register records whether the weight of the master has been used in the adaptive cycle or not. If the value of the left weight register becomes zero then the master is put into the second level priority queue. At the end of every adaptive cycle all the registers are assigned new values.

5. Proposed Combination of Masters According to Their Traffic Behavior with Adaptive

5.1 Adaptive Arbiter

It is very important for shared system-on-chip bus system to be designed in such a way that it meets both hard real time and bandwidth requirement. The difficulty to meet both real-time and bandwidth requirements generally depends on the total required bandwidth in a system [12]. In an unbalanced workload environment, different process requires different bandwidth. Due to the unbalanced bus requirements, the improvement due to parallelization of the code is not as high as compared to the sequential execution [14].

From section 3 it is clear that masters designed according to their traffic behavior uses 94% of the total bus bandwidth. Adaptive Arbiter (AA) used by Haishan Li et al. [13] is better than the conventional arbiters in terms of fair bandwidth allotment and latency. Since arbiter is an important functional module in multiprocessor, therefore it should be carefully designed in high performance systems [15]. In order to optimize the performance of the bus, the time required to handle the request should be minimized [16]. Adaptive arbiter can provide the best bus bandwidth allocation as it distributes different bus bandwidth according to the real-time needs of different masters. If masters designed according to their traffic behavior are put on to the adaptive arbiter architecture then not only bus bandwidth will be saved by 6% for the critical requirements but also it will help to get best possible bandwidth allocation for any real time system by utilizing the CPU cores to its maximum. In the following table the masters classified according to their traffic behavior are deployed to the adaptive arbitration architecture and a comparative analysis has been done in terms of fair bandwidth allotment.

Masters(processors)	Type	Fair Bandwidth allotted using AA
M1	D	27.3%
M2	D_R	26.5%
M3	ND_R	24.1%
M4	ND_R	22.1%

Table 2. Comparative Analysis (Ref. Haishan Li et al. and Chien-Hua et al.)

In the above table all masters requests for 25% of bus bandwidth where M1 is a heavy traffic master and generates a heavier traffic load to the bus. In the following figure architecture of the proposed system is shown.

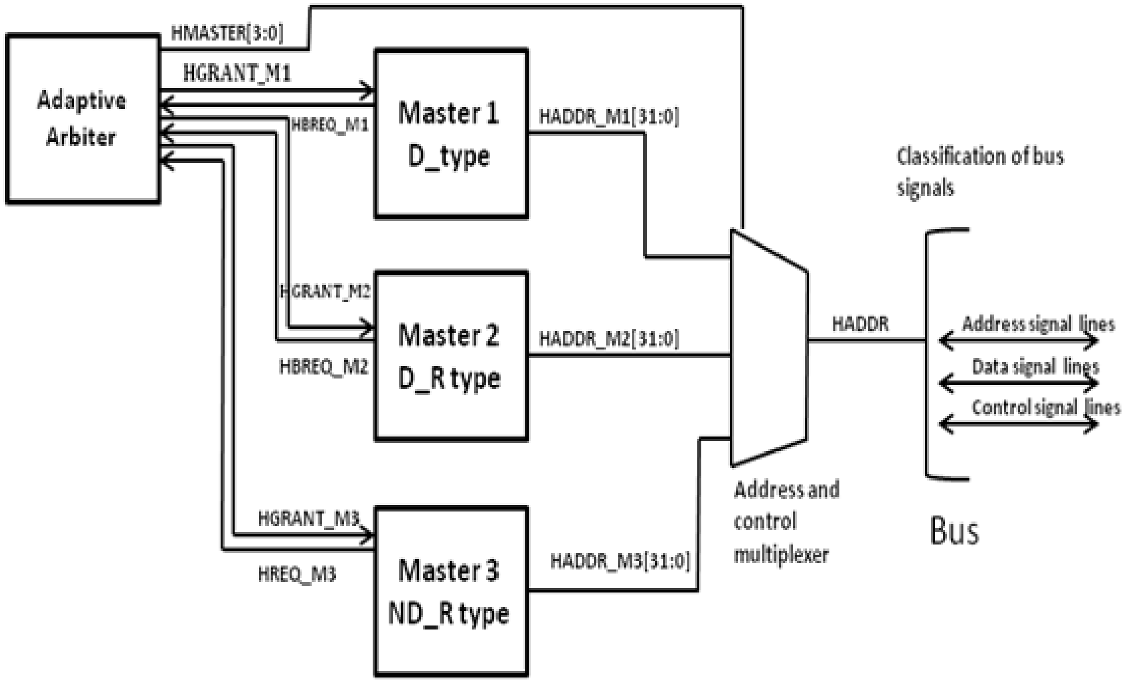

Figure 4. Prototype of the proposed Architecture

6. Results and Discussion

Now days by using the most cost effective means, the designers are trying to get the most out of their designs. In order to analyze the impact of multi-threading on masters according to their traffic behavior using adaptive architecture, we have conducted an experiment using SystemC whose libraries were ported in an integrated development environment composed of suitable profiler tools to measure the CPU usage.

In our case we used visual studio 2010 (EE) to implement SystemC using AMD Athlon ™ II X2 260 processor with 1 MB dedicated L2 cache. Graph in figure 5 shows an average CPU utilization for the threads running for the multiple modules of STREAM [21] without using adaptive arbitration algorithm in a standalone mode.

In our first case the race conditions occur because the programmer does not anticipates the fact that a thread can be preempted at any awkward position. This might allow another thread to read the block of code first. However the use of threads requires some precaution in communication libraries in order to avoid the race condition when the threads access the library concurrently [22]. This type of process does not utilize the CPU cores to the maximum and as the number of threads increases the bandwidth becomes the major barrier for the performance.   Processor architects have to trade off the speed versus a lot of the features it may offer. In order to maximize the instruction per clock cycle, the instruction, operands and destination must be accessible at the same time rather than sequentially. As the CPU are getting faster much more quickly, so the speed of the systems main memory may also become a limiting factor in terms of throughput after a certain threshold point.

Graph in figure 6 shows an average CPU utilization for the threads running for the multiple modules of STREAM using the proposed arbitration algorithm. Thread safety is one of the major criteria of multi-threading support. This means that communication in a multi-threaded application can be performed in multiple threads. Appropriate techniques should be used to utilize the multiple cores in order to make non-blocking communication primitives to progress in the background [22]. Various thread-scheduling policies try to achieve optimal utilization of the CPU as well as the bus bandwidth during each quantum [4]. Three major benchmarks have to be dealt over here simultaneously which consists of CPU utilization, bus bandwidth consumption and system latency. If we compare figure 6 with figure 5, it can be seen that, not only the CPU usage is utilized to its maximum using optimal bandwidth utilization as the threads are running in a preemptive environment but it also shows a decrement in the system latency. Graph in figure 7 shows the fair bandwidth allotment for the masters requiring the same bandwidth using the proposed intelligent adaptive arbitration method where the fluctuation rate recorded is just 1.7159%.

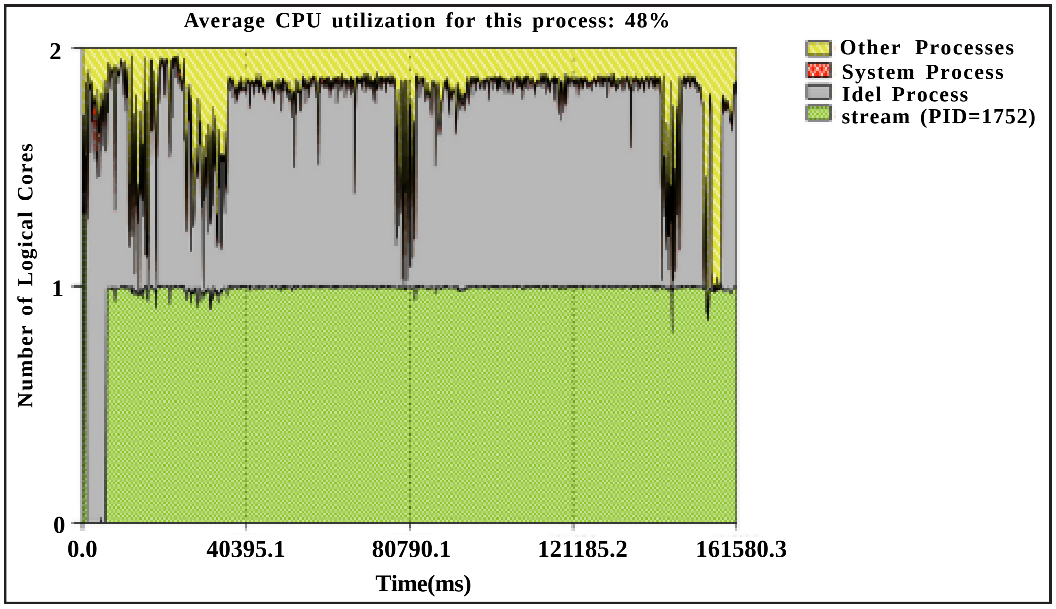

Figure 5. Threads running in a non-preemptive environment

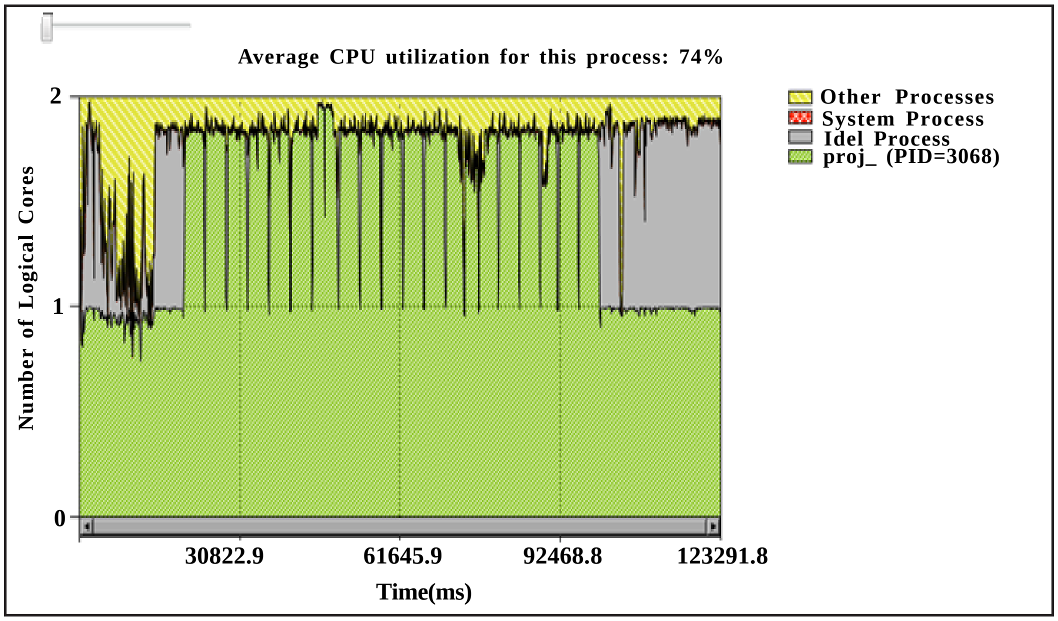

Figure 6. Threads running in a preemptive environment

It is observed from the table II that in terms of fair bandwidth allotment for the processors according to their traffic behavior the adaptive arbiter comes out to be superior if compared with other conventional arbiters. In a multi-microprocessor bus it was found that the point of saturation reached when five processors issued low level requests for the contention of the bus [23]. Haishan Li et al. [13] presented an experimental study by taking two separate cases into consideration. The set of masters taken are M1, M2, M3 and M4. In their first case all the masters require the same amount of bus bandwidth. In figure 8 the graph shows the fluctuation in the bus bandwidth for the different arbitration algorithms when all the four masters requests for same amount of bus bandwidth. It is observed from the graph that in case of adaptive arbitration algorithm the difference between the requested bandwidth and the allotted bandwidth is ± 2% where as in the case of static fixed priority algorithm its ± 8%, in case of intelligent adaptive arbitration its ± 1.71% and the minimum is in the case of round robin algorithm that is ± 0.15%.In the second case of Haishan Li et al. [13] all the four masters require different bandwidth and the proportion are 40%, 30%, 20% and 10% separately. In figure 9 the graph shows the fluctuation in the bus bandwidth for the different arbitration algorithms when all the four masters requests for variable bus bandwidth. It is observed from the graph that in case of adaptive arbitration algorithm the difference between the requested bandwidth and the allotted bandwidth is ± 3.5% where as in the case of round robin algorithm its ± 4.75%, in case of intelligent adaptive arbitration its ± 3.4% and the minimum is in the case of static fixed priority algorithm that is ± 2.15%.

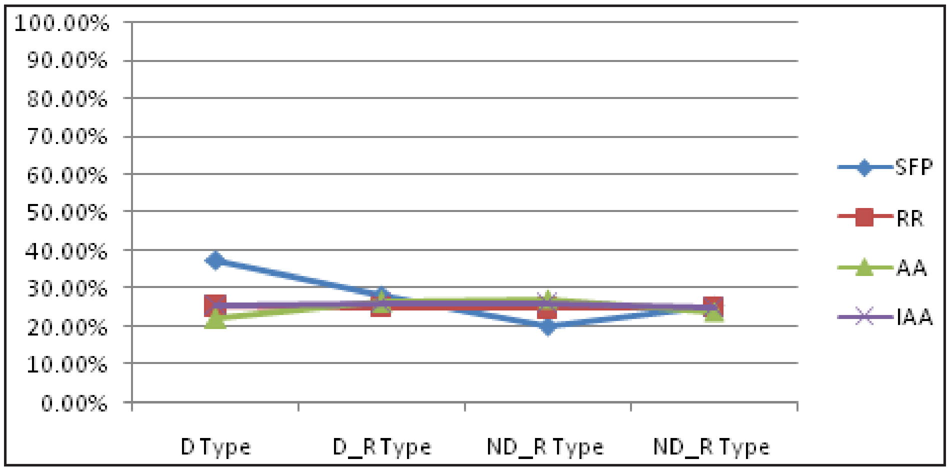

Figure 7. Comparison between conventional arbiter and intelligent adaptive arbiter

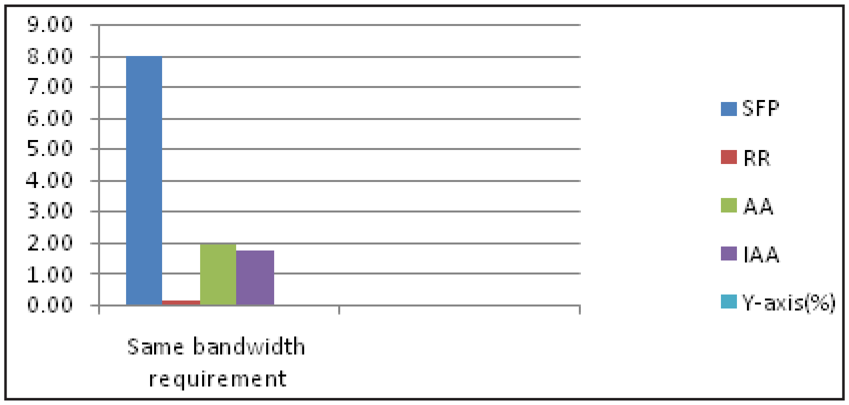

Figure 8. Fluctuation in the bus bandwidth (Case I)

From these three graphs it can be concluded that the static fixed priority algorithm or the round robin algorithm can provide the best bandwidth allocation in one case but the worst bandwidth allocation in the other case. If compared with static fixed priority algorithm and round robin algorithm, the proposed algorithm holds relatively good in all the cases.

Yi-Xu et al. [10] also conducted an experiment to evaluate the bus bandwidth utilization using lottery bus algorithm for four masters where each master requires similar amount of bus bandwidth. The following graph shows the fluctuation in the bus bandwidth for the adaptive arbitration algorithm and the lottery bus algorithm when all the four masters requests for similar bus bandwidth.

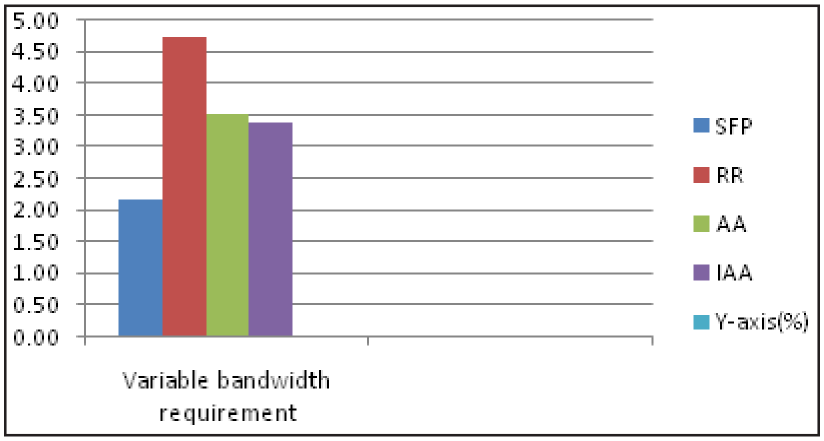

Figure 9. Fluctuation in the bus bandwidth (Case II)

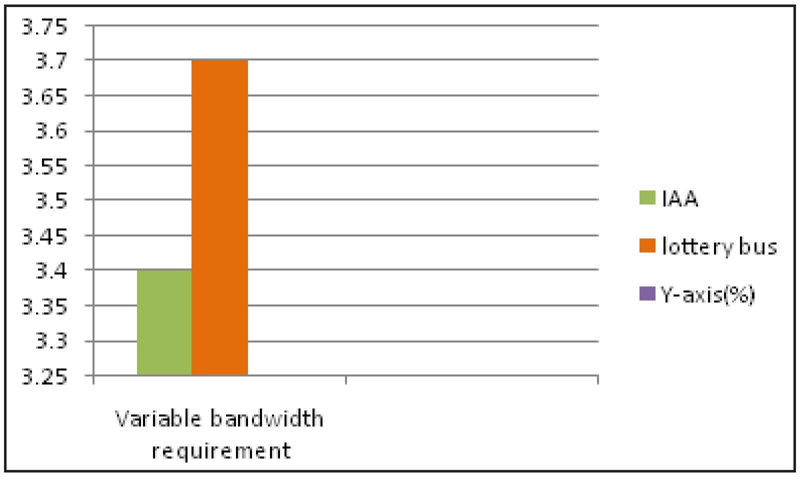

Figure 10. Fluctuation in the bus bandwidth (Case III)

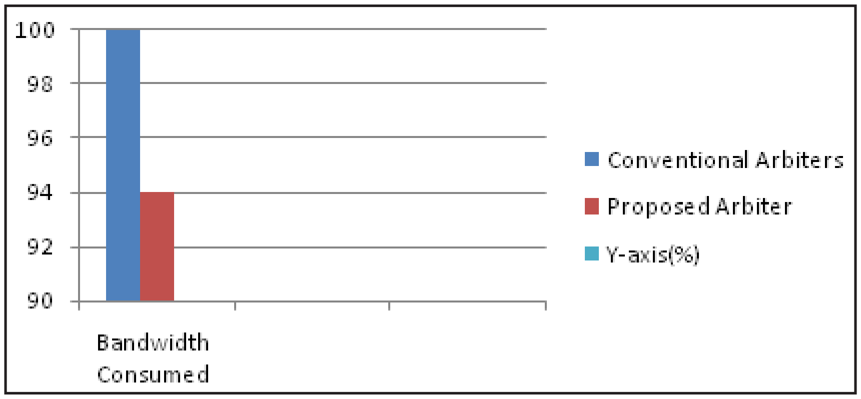

Figure 11. Comparison between Conventional and Proposed Arbiter

It is observed from the above graph that in terms of variable bandwidth requirement, intelligent adaptive arbitration algorithm still holds better if compared with lottery bus algorithm by a margin of ± 0.30%.It is observed from the table II that in terms of fair bandwidth allotment for the processors according to their traffic behavior the proposed arbiter comes out to be superior if compared with other conventional arbiters. In a multi-microprocessor bus it was found that the point of saturation reached when five processors issued low level requests for the contention of the bus [18]. The following graph shows the comparison between the conventional arbiter and the proposed arbiter in terms of their bus bandwidth consumption.

If compared with other conventional arbiter, the proposed arbiter reduces the consumption of bus bandwidth by 6% [23]. During the process of adaptive arbitration if the time slot of any upcoming master is not to be utilized then immediately the second level of the arbiter comes into an action. If hardware overhead would not have been the major factor then the overall performance of the system could have been enhanced up to a large margin just by increasing the levels of arbitration and splitting the bus into multiple layers. After analyzing the above graphs it can be said that the proposed arbiter can solve the problem of maximum CPU utilization, fair bandwidth allotment and system latency for a real time computing system in a very efficient manner and with a very simple design if compared with conventional arbiters [24],[25].

7. Conclusion

An attempt on reducing the system latency and achieving a fair bandwidth allocation by utilizing the CPU cores to its maximum is being made**.** More investigation has to be done on reducing the system latency up to a large margin**.** The proposed arbiter architecture is purely a system level architecture and can be implanted on any real time systems such as biomedical devices which need a guaranteed fair bandwidth with negligible amount of latency. In a nutshell this arbitration technique helps in achieving a fair bandwidth allocation for the critical requirements by utilizing the CPU cores to its maximum and it also reduces the system latency up to an adequate margin. This article provides a comprehensive picture of the global scenario of an arbiter for real time computing system so as to enable researchers to decide the direction of further investigation.

8. Acknowledgement

This research is supported by Institute of Post Graduate Studies (IPS) unit of Universiti Sains Malaysia. I would like to thank both CEDEC and IPS for their support in this research.

References

[1] Pasricha, S., Dutt, N. (2008). On–ChipCommunication Architectures: System-on-Chip Interconnect. Morgan Kaufmann, USA.
[2] Poletti, F., Bertozzi, D., Benini, L., Bogliolo, A(2003). Performance Analysis of Arbitration Policies for SoC Communication Architectures. *Des. Autom. Embed. Syst*. 8 (2) 189-210.
[3] Jian, W., Yubai, L., Qicong, P., Taiqiu, T. (2009). A Dynamic Priority Arbiter for Network-on-Chip. *In*: IEEE International Symposium on Industrial Embedded Systems, p. 253-256.
[4] Antonopoulos, C., Dimitrios, S. N., Theodore, S. P (2003). Scheduling Algorithms with Bus Bandwidth Considerations for SMP's. *In*: IEEE International Conference on Parallel Processing, p. 547-554.
[5] Laxmi, N. B. (1987). Analysis of Interconnection Networks with Different Arbiter Designs, *J. Parallel Distrib. Comput.* 4 (4) 384-403.
[6] Lakshmi, V., Wood, K., Downs, T. (1994). A Four-Channel Communications Arbiter for Multiprocessor Arrays. *Microprocess. Microsyst.* 18 (5) 253-260.
[7] Macii, E., Poncino, M. (1998). Automatic Synthesis of Easily Scalable Bus Arbiters with Dynamic Priority Assignment Strategies, *Comput. Electr. Eng*. 24 (3) 223-228.
[8] Nelson, J. C. C., Refai, M. K. (1984). Design of a Hardware Arbiter for Multi Microprocessor Systems. *Microprocess. Microsyst.* 8 (1) 21-24.
[9] Bowen, B. A., Buhr, R. J. A. (1980). The Logical Design of Multiprocessor Systems. Prentice Hall, Eaglewood Cliffs, NJ, USA.
[10] Yi, X., Li, L., Ming-lun, G., Bing, Z., Zhao-yu, J., Gao-ming, D., Wei, Z. (2006). An Adaptive Dynamic Arbiter for Multi-Processor SoC. *In*: 8th IEEE International Conference on Solid-State and Integrated Circuit Technology, p. 1993-1996.
[11] Aravind, A. A. (2005). An Arbitration Algorithm for Multiport Memory Systems. *ELEX*. 2 (19) 488-494.
[12] Chien-Hua, C., Geeng-Wei, L., Juinn-Dar, H., Jiang-Yang, J. (2006) A Real Time Bandwidth Guaranteed Arbitration Algorithm for SoC Bus Communication. In: 11th Asia and South Pacific IEEE International Conference on Design Automation, p. 24-27.
[13] Haishan, L., Ming, Z., Wei, Z., Dongxiao, L. (2007). An Adaptive Arbitration Algorithm for SoC Bus. *In*: IEEE International Conference on Networking, Architecture and Storages, p. 245-246.
[14] Bourgade, R., Rochange, C., De Michiel, M., Sainrat, P. (2010). A Multi-Bandwidth Bus Arbiter for Hard Real Time. *In*: IEEE 5th International Conference on Embedded and Multimedia Computing, p. 1-7.
[15] Shanthi, D., Amutha, R. (2011). Performance Analysis of On-Chip Communication Architecture in MPSoC. *In*: IEEE International Conference on Emerging Trends in Electrical and Computer Technology, p. 811-815.

[16] Shanthi, D., Amutha, R. (2011). Design of Efficient On-Chip Communication Architecture in MPSoC. *In*: IEEE International Conference on Recent Trends in Information Technology, p. 364-369.
[17] Trahay, F., Brunet, E., Denis, A. (2009). An Analysis of the Impact of Multi-Threading on Communication Performance. *In*: IEEE International Symposium on Parallel & Distributed Processing, p. 1-7.
[18] Scarabottolo, N., Bedina, A., Distante, F. (1982). Implementation Guidelines of a Modular General Purpose Multi-Microcomputer. J. Syst. Architect. 9 (5) 309-313.
[19] Ruibing, L., Aiqun, C. (2007). SAMBA Bus- A High Performance Bus Architectures for System-on-Chips. *IEEE Trans. Very Large Scale Integr.(VLSI) Syst*. 15 (1) 69-79.
[20] Lin, B. C., Lee, G. W., Huang, J. D., Jou, J. Y. (2007). A Precise Bandwidth Controller Arbitration Algorithm for Hard Real-Time SoC Buses. *In*: Asia and South Pacific IEEE International Conference on Parallel & Design Automation, p. 165-170.
[21] McCalpin, J. D. (1995). Memory Bandwidth and Machine Balance in Current High Performance Computers, *In*: IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter.
[22] Trahay, F. et al.(2009). An analysis of the impact of multi-threading on communication performance, *In*: IEEE International Symposium on Parallel & Distributed Processing, p. 1-7.
[23] Akhtar, M. N., Sidek, O. (2011). An Arbiter with Fair Bandwidth Allocation and Low Latency for Real Time Computing System. *In*: International Conference on Computer Technology and Development, p. 189-196.
[24] Akhtar, M. N., Sidek, O. (2011). An Intelligent Arbiter for Fair Bandwidth Allocation. *In*: IEEE Students Conference on Research and Development, p. 304-309.
[25] Akhtar, M. N., Sidek, O. (2012). An Intelligent Arbiter for Maximum CPU Utilization, Fair Bandwidth Allocation and Low Latency: Survey, *In*: International Colloquium on Signal Processing and its Applications, p. 266—271.