DESIGN AND SIMULATION OF AN INTELLIGENT ADAPTIVE ARBITER FOR MAXIMUM CPU USAGE OF MULTICORE PROCESSORS

MOHAMMAD NISHAT AKHTAR

UNIVERSITI SAINS MALAYSIA

2013

DESIGN AND SIMULATION OF AN INTELLIGENT ADAPTIVE ARBITER FOR MAXIMUM CPU USAGE OF MULTICORE PROCESSORS

by

MOHAMMAD NISHAT AKHTAR

**Thesis submitted in fulfillment of the requirements
for the degree of
Master of Science**

July 2013

ACKNOWLEDGEMENTS

First and foremost, All Praises be to Allah the Almighty, for delivering me the patience, the potency and the guidance to conclude this thesis fruitfully.

I owe the greatest debt to my parents and all my relatives for not only their never-ending affection, love and patience, but also their encouragement and unconditioned support to me in all my decisions. Their guidance helped me to keep my motivations high throughout my studies.

I would like to express my sincere gratitude and appreciation to my supervisor, Professor Othman Bin Sidek for his guidance, supports, and motivations and above all to offer me opportunity by accepting me as a Master's student.

I am grateful to Miss. Asmah Mat Taib for translating abstract into bahasa melayu. I thank all of my lab friends, my housemates Mr. Thaif, Mr. Manjur, Mr. Nazmul and all staffs of University Sains Malaysia for their supports. I would also like to thank the Institute of Postgraduate Studies (IPS) for their financial support by awarding me the USM Research Scholarship.

Finally, I would like to thank all my family members in India, who always inspired me to do something better and worked as a hidden power in all steps of my life. Without their speechless indirect inspiration and given concerns from Allah, maybe I would have diverted from my mission.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AA	Adaptive Arbitration
ATM	Asynchronous Transfer Mode
CPU	Central Processing Unit
DARPA	Defence Advanced Research Projects
EE	Express Edition
FIFO	First in First Out
FPGA	Field Programmable Gate Array
IAA	Intelligent Adaptive Arbitration
IC	Integrated Circuit
IP	Intellectual Property
LRU	Least Recently Used
MPSoC	Multi Processor System on Chip
MTD	Multi Threaded Debug
NoC	Network on Chip
ISR	Interrupt Service Routine
RAG	Round Robin Arbiter Generator
RR	Round Robin
SFP	Static Fixed Priority
STREAM	Sustainable Memory Bandwidth in High Performance Computers
TDMA	Time Division Multiple Access

LIST OF PUBLICATIONS

Journals:

1. **M. Nishat Akhtar** and O. Sidek, “ An Intelligent Adaptive Arbiter for Maximum CPU Utilization, Fair Bandwidth Allocation and Low Latency”, in IETE Journal of Research- (in Press,Under issue preparation) (ISI cited publications).

2. **M. Nishat Akhtar** and O. Sidek, “An Accommodative Adaptive Arbitration Algorithm for Maximum CPU Utilization, Fair Bandwidth Allocation and Low Latency” in Journal of Networking Technology. PP 71-80.

Book Chapter

1. **M.Nishat Akhtar** and O.Sidek, “An Adaptive Arbitration Algorithm for Fair Bandwidth Allocation, Low Latency and Maximum CPU Utilization”, Part I, CCIS 293, pp. 330–343, 2012. © Springer-Verlag Berlin Heidelberg 2012.

Conferences:

1. **M. Nishat Akhtar** and O. Sidek, "An arbiter with fair bandwidth allocation and low latency for real time computing system," in 3rd International Conference on Computer Technology and Development, 2011,(Paper Selected for ASME) pp. 189-195.

2. **M.Nishat Akhtar** and O.Sidek, “An intelligent arbiter for fair bandwidth allocation,” presented at the 9th IEEE SCORED 2011, pp. 322-327.

3. **M.Nishat Akhtar** and O.Sidek, “An Intelligent Arbiter for Maximum CPU Utilization, Fair Bandwidth Allocation and Low Latency:Survey”, in IEEE-CSPA-2012. PP 266-271

REKABENTUK DAN SIMULASI PENGADIL ADAPTIF CERDIK BAGI PENGGUNAAN MAKSIMUM CPU PEMPROSES MULTIKOR

ABSTRAK

Teknologi terkini dalam dunia mikro dicampur dengan cip yang kompleks yang menggabungkan pelbagai pemproses khusus untuk keperluan pengiraan tertentu. Oleh itu, dalam mana-mana sistem memori yang dikongsi, teknik arbitrasi memainkan peranan yang penting untuk memperuntukkan akses kepada sumber-sumber yang dikongsi bersama. Cabaran utama ditangani dalam penyelidikan yang dicadangkan adalah pencapaian penggunaan CPU maksimum dengan mengeksploitasi teras berganda dengan bas sederhana peruntukan jalur lebar dan sistem kependaman rendah. Dalam usaha untuk menangani masalah-masalah yang tersebut di atas, satu teknik arbitrasi penyesuaian pintar telah dicadangkan untuk unit-unit tuan direka mengikut tingkah laku trafik aliran data. Cadangan teknik arbitrasi penyesuaian pintar dilaksanakan menggunakan STREAM, yang merupakan program penanda aras sintetik yang mengukur kadar pengiraan dan jalur lebar memori mampan. Dari segi analisis prestasi, teknik arbitrasi yang dicadangkan itu telah dibandingkan dengan teknik arbitrasi baru-baru ini, seperti teknik penyesuaian arbitrasi, loteri arbitrasi bas dinamik, pusingan robin arbitrasi dan statik arbitrasi keutamaan tetap. Bagi meningkatkan penggunaan CPU dan pengoptimuman jalur lebar, teknik arbitrasi yang dicadangkan itu telah dimodelkan menggunakan benang SystemC dan OpenMP menggunakan kaedah pengaturcaraan selari bagi membolehkan pengkomputeran pelbagai teras. Beberapa teknik arbitrasi baru-baru ini mencapai bas adil peruntukan jalur lebar sehingga sedikit tetapi gagal untuk mencapai penggunaan CPU maksimum, sebagai pemproses menghabiskan 95-96% daripada masa mereka terbiar dan menunggu untuk cache tersasar akan berpuas hati. Teknik arbitrasi yang dicadangkan adalah kes yang kuat memihak kepada penggunaan CPU maksimum dan pengoptimuman jalur lebar, kerana ia menggunakan teras pemproses sehingga 74% dan juga mengurangkan jalur lebar turun naik serta kependaman.

DESIGN AND SIMULATION OF AN INTELLIGENT ADAPTIVE ARBITER FOR MAXIMUM CPU USAGE OF MULTICORE PROCESSORS

ABSTRACT

The recent technology in the world of microprocessor is blended with complex chips that incorporate multiple processors dedicated for specific computational needs. Therefore, in any shared memory system, an arbitration technique plays an important role to allocate access to the shared resources. The major challenge dealt in the proposed research is the achievement of maximum CPU utilization by exploiting its multiple cores with moderate bus bandwidth allocation and low system latency. In order to tackle the aforesaid problems, an intelligent adaptive arbitration technique has been proposed for the masters designed according to the traffic behaviour of the data flow. The proposed intelligent adaptive arbitration technique is implemented using STREAM, which is a synthetic benchmark program that measures computational rate and sustainable memory bandwidth. In terms of performance analysis, the proposed arbitration technique has been compared with the recent arbitration technique, such as adaptive arbitration technique, dynamic lottery bus arbitration, round robin arbitration and static fixed priority arbitration. To enhance the CPU utilization and bandwidth optimization, the proposed arbitration technique has been modelled using SystemC and OpenMP threads using the method of parallel programming to enable multi-core computing. Some recent arbitration technique achieves fair bus bandwidth allocation up to some extent but fails to achieve maximum CPU utilization, as the processor spends 95-96 % of their time idle and waits for cache misses to be satisfied. The proposed arbitration technique is a strong case in favour of maximum CPU usage and bandwidth optimization, as it consumes the processor cores up to 74% and also reduces the bandwidth fluctuation as well as latency.

CHAPTER 1

INTRODUCTION

1.1 Background

An arbiter is considered to be an electronic device which allocates access to the shared resources. In an environment of multi-core systems, the common bus of the system-on-chip (SoC) is the sharing resource which is shared by multiple cores of the master. An arbiter plays a crucial role when it comes to granting an authority to utilize the shared resource efficiently. It ensures that at a time, at least one master gets access to the bus by observing the number of request issued by different number of masters during any cycle. It samples the multiple requests and decides which master should be given access to the shared bus.The main aim of an arbitration process is to assign processes to be implemented by the processor in such a manner that it meets the objectives such as efficient processor utilization, bandwidth optimization and low latency. As the technology is scaling towards the deep submicron, the feasibility for the integration of multiple processors on a chip is becoming possible. Every year, more amount of transistors are made compatible to fit on a single die, which adverts Moore's law. The decrement in the dimensional size of the transistor has led to new views of on chip design. For every new design, the communication policy between different Intellectual Property (IP) plays an important role, as the communication methodology increases exponentially with decreasing dimensions of the transistor. In such kind of scenario, arbitration technique becomes complex, as the smaller feature increases the functionality which can be embedded on a single chip. In order to avoid scaling problem and to link different functional units of the system, a bus type structure is required for a chip. However the bus is designed in such a way that it can handle only finite functional units due to the limiting bandwidth factor and less processor support but an adept effort is being put in order to find numerous solutions to this problem by optimizing the bus bandwidth and increasing the CPU utilization using parallel programming.

An arbiter should be designed in such a way that it should be able to exploit the multiple cores of the processor because the earlier designed arbiters used sequential programming method in a single core environment. With an increasing consumer desire, the single core products have started showing decrement in the product performance. In order to tackle this problem, the feature of multi-core processing is recognized as a key component. In order to meet the users demanding requirements, the industry has taken the challenge to increase the number of cores. For several years, the technology of dual core, quad and octa core had been established, which is considered to be the beginning of a massive technological era which is about to arrive. However this has become a challenge not only for the semiconductor industry, but also for the software and the system designers who made them work. Writing applications, that are compatible to the system of parallel execution is quite arduous but it is the only solution to make the arbitration technique effective. If the system needs to obtain an optimal performance, then there is a requirement for a complex changes to be made in both system and the software. Before the arrival of multi-core technology, numerous efforts have been made in enhancing the performance for which the approach used were brute force on a single core system. For instance, the designers used to crank up the frequency, but on the other hand, with the increase in frequency, the whole system used to become incompatible, as the improvement in the frequency penalizes power consumption, which in turn generates heat that needs cooling of an advanced stage due to which the reliability decreases, and shortens the device life. These additional problems increases the overhead cost of the whole system. Various techniques such as pipelining and speculative execution does not generally scale up with the frequency of the processor. On another instance, at certain levels an instruction pipeline possess an internal clock requirements that cannot become compatible if the frequency of the processor clock is increased. Therefore, this necessitates the requirement of additional stages of pipeline process which in turn increases the number of cycles, that is required to perform the execution. On the other hand, it has been proven from Amdahl's law that, if 90-95% of the task is parallelized, then increment in the number of processor cores shows no improvement in system speed *[Amdahl, 1967]*. Task parallelization in multi-core environment system is of great need because doubling the core could only benefit the users, if the designers come up with the program that is compatible to be executed in the multi-core environment by using its resources to the maximum using an advanced arbitration technique.

An arbiter ensures compatibility between on-core speed with the off-core speed because a kind of memory wall starts building up if an enhancement in the on-core speed is not compatible with the off-core and I/O subsystem [*Bajrovic and Mehofer*, 2009]. A lower frequency bus matched with the higher frequency core will stall the system frequently as the core waits for the data. These mismatches have been compensated till some extent by implementing large and fast, on chip caches, but this cannot be a permanent solution to the problem as by enhancing the size and increasing the on-chip caches, increases both power consumption and the silicon size. The multi-core designs are used as a standard design across the computing spectrum that consists of high-end systems, such as huge servers, telecom infrastructure and supercomputers. Multi-core devices have been in use for many years but in different forms, for instance in the form of uni or the dual RISC cores inside QUICC Engine communication unit [*Dumitrescu et al.*, 2006]. A device that has a multiple cores with various types of instruction sets is known as heterogeneous device, where as homogenous multi-core devices has multiple identical cores in it. In today's scenario, the main focus is to create multi-core homogenous devices, but a significant amount of advantage can only be gained by using accelerators and specialized cores to shed the load from the main cores [*Ebrahimi et al.*, 2012]. The central challenge considered for multi-core environment is the task of parallelization [*Singh and Rattan*, 2003].In order to attain high degree of task parallelization, an arbitration technique plays a major role by synchronizing the execution of multiple cores. However the concept of parallel computation is not new for the industry but in order to implement a system which becomes compatible to run in a parallel computing environment is quite arduous task. Multi-core emphasizes more on data and task parallelism using fine arbitration technique as it focuses on the sector where the software and the system design matters a lot.

An efficient implementation and design of an arbiter is considered critical for various multi-core SoC designs. Usage of an arbiter not only includes the common multi-port memory modules, communication routers, shared address and data buses but also the applications which are considered to be less obvious such as schedulers, semaphores, SoC core allocators and instruction dispatchers. The significant and an essential problem for the design of an arbiter resides in its efficiency. The difficulty arises to find a feasible architecture that could optimize the bus bandwidth by the means of fair bandwidth allocation to the requesting modules and by maximizing the CPU utilization, as the processor spends more than 90% of its time in satisfying the cache misses [*Mishra et al.*, 2004]. Therefore it is necessary to choose a parallel implementation that is able to arbitrate the requests issued by different modules abruptly using the feature of maximum CPU utilization and moderate bandwidth allocation. The design of an arbiter should be kept as simple as possible but it is most important for an arbiter to ensure that it handles the critical path in an efficient manner.

1.2 Research Objective

Increasing the number of cores on the processor is of no use to gain the system speed. An arbiter is required which can exploit the multiple cores of the processor with a moderate bandwidth allocation. Therefore a new arbitration technique is proposed which is called an intelligent adaptive arbitration technique.

The performance of the new arbitration design is compared with other well known arbitration policies for a bus based environment. Various arbitration techniques implemented earlier lacks efficiency in terms of CPU utilization and moderate bus bandwidth allocation, therefore it is essential to come up with an advanced arbitration technique. The final design is modeled using SystemC and OpenMP tools which makes this arbitration technique different.

A research study designed to implement the proposed arbitration technique has the following research objective:

- To develop an arbitration technique which can exploit the multiple cores of the processor using the method of multi-threading and parallel programming
- To ensure a moderate bus bandwidth allocation and low latency for the proposed arbitration technique

1.3 Thesis contribution

The research builds upon previous and ongoing work on the recent arbitration techniques. Earlier implemented arbitration techniques were unable to exploit the multiple cores of the processor. However some of the arbitration techniques were fair in terms of bandwidth allocation. The proposed approach differs with other arbitration techniques in terms of CPU utilization with the use of parallel programming. In order to implement a smooth arbitration, each master core has been synchronized with other cores. Since there is a high degree of task parallelization, therefore there is a drastic decrement in the latency too. To simulate the whole arbitration technique, tools like SystemC and OpenMP has been used in a single environment.

The dissertation contributes to develop an advanced arbitration technique which enables parallel programming using multiple cores of the processor by allocating each core with moderate bandwidth. This arbitration technique also maintains low latency. The proposed arbitration technique is known as intelligent adaptive arbitration.

1.4 Thesis organization

Chapter 2 discusses related works, which elaborates earlier implemented arbitration techniques in detail. Chapter 3 introduces the proposed model, which describes the implementation of multiple master cores with each other by attaining a high degree of synchronization. It also discusses the way systemC and OpenMP threads interact with each other. Chapter 4 contains a detailed description of the obtained results, which consists of CPU utilization rate, bandwidth allocation values and latency rate. Finally in Chapter 5, the conclusions are given and some recommendations are made for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, levels of arbitration techniques are discussed and a comparison is made between a bus based arbiter and a network-on-chip (NoC) arbiter. General background of arbitration techniques used by other researchers have been discussed. Performance comparison of arbiters and earlier implemented arbitration techniques are discussed in more detail. Granularity in multiprocessor scheduling environment is elaborated so as to get an overview of synchronization between multiple master core with each other, so that they can be arbitrated parallely. Allotment of process to processors are discussed. This ensures each master/parent process to keep a local copy of its address before execution to enable smooth arbitration without getting stalled. Finally, the need and importance of real-time computing and thread scheduling for an arbiter is discussed in which its characteristics and methods are elaborated.

Section 2.2 discusses related works of various arbitration techniques. Section 2.3 describes performance comparison of arbiters. Section 2.4 shows earlier implemented arbitration techniques. Section 2.5 discusses granularity in multiprocessor scheduling. Section 2.6 discusses design issues concerned with assignment of processes to processors. Section 2.7 discusses some of the characteristics of real-time computing system and section 2.8 discusses thread scheduling for parallel implementation of task.

2.2 Related works

There are several arbitration schemes that are commonly used in bus based communication architectures. To reduce the time complexity of the whole operation numerous researches has been done in the area of arbitration algorithm. The static priority algorithm, round robin algorithm and time division multiplexing algorithm can be called as the base algorithm for any arbitration scheme. A two-level arbitration scheme can be created by combining two arbitration schemes. In the recent years, arbitration algorithm with varying degree of sophistication have been developed and applied to the on chip communication protocols. Sonics Smart Interconnect [*Pasricha and Dutt*, 2008] makes use of a two-level Time Division Multiple Access/Round Robin arbitration scheme. In this scheme, a Time Division Multiple Access (TDMA) arbiter allocates time slots to various masters. If a master does not have any data to transfer during its time slot, a second level round robin scheme selects another master to grant the bus access.

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 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 [*Mishra and Dehuri*, 2011]. As, the technology is shrinking below 45nm, therefore there is a great need for the multi-core applications to be redesigned in such a way that it utilizes most of the cores to take optimal advantage of high performance processing. Through numerous challenges and a great effort, the performance race for embedded system and desktop computing has tackled the issues upto some extent that arises with the increasing frequency of the processor [*Marongiu et al.*, 2011]. However on the other hand, some of the most innovative workarounds are about to come to an end. Therefore in order to continue delivering system with the higher performance and increased efficiency, a new route should be taken, so that increment in the number of CPU cores does not comes to be a great challenge for the system designers, the trend which is generally followed in supercomputers and various other high end sophisticated systems.

In a bus based environment, multiple processors can be combined on the same IC, which leads to the development of Multi Processor SoC (MPSoC) system. MPSoC becomes compatible to accommodate sophisticated parallel computing applications, as it combines embedded systems, operating system and even analog circuits [*Lin et al.*, 2011]. In order to design such a system, certain level of challenges has to be faced by the designers. The speed of the processor does not play a major role in the performance of the multiprocessor systems as it depends more on efficient communication architecture among multiple processors and on the computation using balanced distribution among them [*Bowen and Buhr*, 1980]. There are multiple communication architectures in which shared bus architecture is quite popular for system which involves small number of processors due to its simplicity and area efficiency. Moreover, An arbiter should have an ability to adjust the bus bandwidth proportion assigned to multiple processors automatically [*Ebrahimi et al.*, 2012].

Apart from all these techniques, the concept of NoC arbiter has also been introduced. The core aim of a NoC is to divide the design in different functional units which can be called either a resource, an intellectual property or a system element, and connects these functional units through a universal communication network. There are N number of ways to create any kind of network, but it is quite easier for the designers to handle the system if they keep the network relatively simple, as the network interface are extremely complex, or the amount of design effort is being applied more into linking different nodes using an arbiter [*Zitouni and Tourki*, 2008]. NoC arbitration architecture also seems to be quite efficient as it has to be designed once for each new technology. However the communication bottleneck can be handled using the scalability of the NoC and theoretically it has got an ability to be extended to infinity. Network-on-Chip has got its own pros and cons. For smaller structures the bus architecture out weighs NoC in terms of performance. NoC architecture can overcome the demerits of traditional bus based architecture as there is tremendous research going on NoC based system architecture [*Zitouni and Tourki*, 2008].

On the other hand, there are several disadvantage of NoC architecture too. 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 . Second disadvantage is because of its fixed grid, the communication latency between two functional units becomes bigger due to non-linear data paths [*Zitouni and Tourki*, 2008]. 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 [*Zitouni and Tourki*, 2008]. It is quite difficult to quantify the exact or the actual brake even points between bus and a NoC [*Wang et al.*, 2009]. In terms of design, the NoC has got endless possibilities.

Any SoC's main design concern is to partition the system into two modules i.e. hardware and software based on the performance constraints. When the topology of the architecture consists of numerous channels, then suitable bridges are deployed to interlink the necessary channels. In a bus based system, the buses are shared by several masters which in turn necessitates the bus architecture to require certain arbitration protocols to manage the bus access, as the growing complexity of sophisticated chip's necessitates a fine grained scalable communication infrastructure [*Trahay et al.*, 2009]. Arbitration protocols generally includes various techniques such as static fixed, round robin, time division multiplexing and various other dynamic algorithms. It is not possible to understand a complex design unless or until its underlying information provided in a concise manner [*Antonopoulos et al.*, 2003]. Communication topologies which are based on shared bus architectures have got several advantages if compared with NoC topologies in which the various modules performs a remote communication via network nodes because a bus based architecture can be manipulated by the software or the hardware designers. This is due to the fact that, the bus system has got an ability to be presented as a simple medium, which eases the communication between different system modules [*Bhuyan*, 1987]. More over in a bus based system the deadlines of data transfer can easily be foreseeable, as the current systems operates using sequential programming method rather than parallel. There are various factors which can affect the performance of SoC architectures. For instance an uncontrolled allocation of the communication bandwidth to different modules of the system may lead to the starvation of lower priority components which may bungle up the whole system in terms of performance [*Bourgade et al.*, 2010]. Latencies in the system also plays an important roles the variations in the time-profile of the requesting modules may lead to large latencies for higher priority modules.

Thus, an arbiter is required to decide which master should be granted the bus access. Hence an arbiter should be designed in such a way that it suits the system by maximizing the CPU utilization, keeping high throughput and low starvation among the different master cores [*Dong and Rojas-Cessa*, 2012]. 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 [*Poletti et al.*, 2003]. 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 in the system by fair means.

In recent years many researchers focused on developing multi-level arbitration scheme in order to reduce the system latency and to achieve fair bandwidth allocation. Xu et al. [*Xu et al.*, 2007] proposed an arbiter called an adaptive dynamic arbiter in which they proposed 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 and neither it maximizes the CPU utilization as it does not gets implemented using parallel programming method . An arbiter should take a decision in such a way that it suits the systems specification and requirements like the defense advanced research projects (DARPA) arbiter developed by team Caltech in 2004 selects the best vote among the different masters according to the system need in automated vehicle [*K.Henrik*, 2004]. Aravind [*Aravind*, 2005] presented an algorithm which is a fully distributed software solution to the arbitration problem in multi-port 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. Moreover, their arbitration technique follows sequential style of programming and therefore does not make use of the multiple CPU cores. Enrico and Massimo [*Macii and Poncino*, 1998] 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. Their arbitration technique was fair in terms of bandwidth and latency but were least concerned regarding CPU utilization as it did not got implemented using parallel programming method. The major disadvantage of common-bus multiprocessor system is the reduction of throughput caused by conflict between processors requiring access to the shared memory. Ideally, throughput should increase directly with the number of processors but the bus contention diminishes this increase [*Lopez et al.*, 2011]. 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 [*Bhuyan*, 1987]. Abdelkrim and Rached [*Zitouni and Tourki*, 2008] proposed an arbiter synthesis approach that allows a high performance Multi Processor System-on-Chip (MPSoC) communication using Asynchronous Transfer Mode (ATM) switch for multi-bus and NoC architecture. Their result demonstrates that MPSoC offers an attractive alternative to conventional communication architectures by providing low communication latency using sequential style of programming. On the other hand, there are several disadvantage of NoC which have been discussed earlier.

Chen et al. [*Chen et al.*, 2006] 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 as its bandwidth allocation is quite diverse. Their work demonstrated a two level arbitration scheme which comprised of time division multiple access algorithm and lottery based algorithm. They developed master cores according to the traffic behavior of the data flow which consists of both heavy traffic masters and light traffic masters. On the other hand, their masters did not show synchronization among them and were implemented using sequential programming method. Therefore the masters were unable to maximize the CPU utilization. However, in terms of diverse bandwidth allocation, their arbitration technique was superior and were able to handle hard real-time bandwidth requirement. A unique algorithm was proposed by Li et al. [*Li et al.*, 2007], 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. In their case, an 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 [*Li et al.*, 2007]. Their arbiter worked well in terms of fair bus bandwidth allocation but on the other hand it did not exploit the multi-core parallelism.

2.3 Performance comparison of arbiters

According to Pasricha and Dutt [*Pasricha and Dutt*, 2008], arbiters are compared on the basis of user oriented performance and system oriented performance. In order to analyze the user-oriented issues, the following parameters are taken into consideration.

- Turnaround time
- Response time
- Deadlines
- Predictability

In order to analyze the system oriented issues, the following parameters are taken into account.

- Processor utilization
- Fairness
- Average bandwidth utilization

2.3.1 Turnaround time

This is considered as the interval of time between the process submission and its completion which includes the exact execution time and the time spent to get access to the other resources.

2.3.2 Response time

This time is considered as the submission time of requests until the response is received. It is quite possible for some process to produce some of the output parallel as the request is processed. Thus, from user's point of view, it is considered to be a better measure than turnaround time.

2.3.3 Deadlines

There is a specific deadline which is set for the process completion. The scheduling process is supposed to subordinate other goals, so as to maximize the deadlines which is to be met.

2.3.4 Predictability

The amount of running time of the process should be the same as defined and at the same cost regardless of the system load.

2.3.5 Processor utilization

The main aim of the scheduling policy should be to maximize the number of process completion per unit of time. It is a measure of the amount of work done, which clearly depends on the length of the process. There is a high influence of scheduling policy to maximize the number of process completion, as it effects the processor utilization.

2.3.6 Fairness

If the does not specify the priority of the process, then each process should be treated the same by the scheduler and there should not be starvation among the different process.

2.3.7 Average bandwidth utilization

It is the share of bandwidth utilized by different cores of the master or threads. The bandwidth allotted to different master should be moderate and fair.

2.4 Arbitration techniques

According to Buttazo *[Buttazo, 2011]*, the decision mode of an arbitration algorithm can be generally categorized in two manner.

- Nonpreemptive
- Preemptive

Nonpreemptive

In this case, if the process is in the running state, then it continues to execute until it terminates or it blocks itself for input/output operation or it requests for some other operating system service.

Preemptive

In this case, the process which is in the current running state can be interrupted and moved to the ready state. The decision to preempt any process is made if any new process arrives with a higher priority. When an interrupt occurs which puts a blocked process in the ready state. This kind of policy incurs greater overhead if compared with non preemptive policy. This policy prevents monopolization of processor for a very long time.

There are several arbitration algorithms which have been developed and is mentioned as follows:

2.4.1 Static Fixed Priority (SFP) scheme

The most commonly used arbitration scheme is the static fixed priority scheme, in which masters on a bus are assigned fixed priority values. The master with highest priority always gets access to the bus with a fixed bandwidth. This scheme can be implemented in a non-preemptive manner. In a preemptive implementation, an ongoing lower priority data transfer from a master is terminated immediately without being completed if a request for bus access is received from a higher priority master. In a non-preemptive implementation, the ongoing lower priority data transfer from a master is allowed to complete before the bus is handed over to the higher priority master. Static Priority scheme is simple to implement and can provide high performance by ensuring critical data transfers, such as between processor but this scheme should be implemented carefully as it can lead to starvation of lower priority masters, which might never be able to get access to the bus if there are frequent bus accesses by higher priority masters. Wiseman and Feitelson [*Wiseman and Feitelson*, 2003] implemented SFP scheme in terms of strict gang scheduling in a parpar cluster of 8 computing nodes in parallel and measured CPU utilization rate which came out to be 20%. However on a single node, SFP scheme cannot be implemented parallely. Warathe et al. [*Warathe et al.*, 2009] proposed a static lottery bus communication architecture which showed efficiency over traditional arbitration algorithms but the only limitation of this implementation is that the distributions of their resulting random numbers were not uniform. SFP has also been implemented by Li et al. [*Li et al.*, 2007] using four masters for fair bandwidth allocation. However, this technique did not show any sort of fair bandwidth allocation among the masters. This algorithm is simple to implement as it requires a small area and cost. In terms of hefty communication traffic, this algorithm does not hold good as the masters bearing the lower priority suffers starvation.

2.4.2 Time Division Multiple Access (TDMA) arbitration

Time Division Multiple Access (TDMA) arbitration scheme can guarantee a fixed, higher bus bandwidth to masters with higher data transfer requirements and also ensures that lower priority masters do not starve. In this scheme, each master is assigned time slots of varying lengths, depending on the bandwidth requirements of the master. The choice of number of time slots to assign to each master is extremely important. The slots allocated to higher priority masters should not be in such a way that the master with lower priority starts starving. The length of each time frame should be long enough to complete at least a single data transfer. TDMA scheduling diagram mentioned by Xu et al. [*Xu et al.*, 2007] divides execution time on the bus into time slots and allocates time slots to masters in a specific way using preemption.

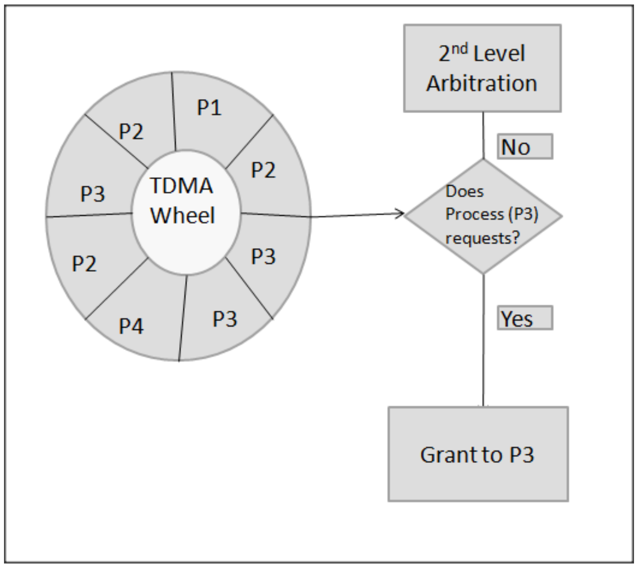

Figure 2.1 Time Division Multiple Access

In figure 2.1 the first level uses a time wheel where each slot is statically reserved for a unique master. If a master possessing the current time slot does not issue request, the time slot would be wasted. To overcome this inefficiency the second level arbiter scheme has to issue the bus to the other master contending for the bus. Moreover, this arbitration technique cannot implement the requesting masters in parallel as the degree of granularity between them is very less. Therefore, it is unable to exploit the multiple cores of the processor.

2.4.3 Round Robin (RR) scheme

Round-robin arbitration scheme ensures there is no starvation in the system. In this scheme, access to the bus is granted in a circular manner, to every master on the bus and makes it certain that every master will eventually get access to the bus. A master abandons control over the bus when it no longer has any data to send and passes the ownership to the next master in queue. The round robin scheme is simple to implement, and can ensure equal bandwidth distribution on a bus, but suffers from a drawback compared to the static priority scheme, that critical data transfers may have to wait a long time before they can proceed. Round robin scheme can be implemented using both preemptive (only applicable for clusters) and non-preemptive manner [*Wiseman and Feitelson*, 2003]. A modified version of round robin algorithm has been proposed by Yaashuwanth and Ramesh [*C.Yaashuwanth and Ramesh*, 2010] which modifies all the drawbacks of a simple round robin algorithm by reducing the high context switch rate, large waiting time and larger response time using non-preemptive method. Shin et al. [*Shin et al.*, 2006] came up with a new algorithm known as round-robin arbiter generator (RAG) tool. This RAG tool can generate a design for a bus arbiter which is able to handle the exact number of bus master for both on-chip and off-chip buses using non-preemptive method. Dong and Rojas [*Dong and Rojas-Cessa*, 2012] introduced two arbitration schemes based on round robin arbitration for combined input cross point buffered packet switches. It has been shown that combined input-cross point buffered switches provide high-performance context switching and relax arbitration timing for packet switches with high-speed port but their architecture does not overcome the disadvantage of fixed framework layout. Ramasubramanian et al. [*Ramasubramanian et al.*, 2009] worked upon the existing algorithms in order to reduce the latency caused by contention among the processors preemptive method but was unable to allocate fair bandwidth among processors. Li et al. [*Li et al.*, 2007] implemented the round robin scheme using non-preemptive method with four masters and achieved fair bandwidth allocation for each master using sequential programming. According to Wiseman and Feitelson [*Wiseman and Feitelson*, 2003] the CPU utilization rate for round robin scheme was 45% in the parpar cluster of 8 computing nodes. Figure 2.2 shows a simple round robin architecture.

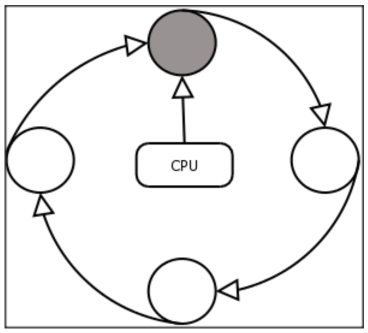

Figure 2.2 Simple round robin architecture

2.4.4 Dynamic priority scheme

This is another complex, but highly efficient arbitration scheme that can dynamically vary the priority of the master during the run time [*Pasricha and Dutt*, 2008]. To analyze the data traffic at the run time additional logic is used, and the priorities are dynamically adapted to the changing traffic profiles of an application. This type of scheme is very useful when the master needs to send large amount of data with low latency. The cost of implementing such type of scheme can be high as it requires several registers to keep track of priorities and data traffic profiles at various point of execution.

2.4.5 Programmable priority scheme

This is a simpler variant of the dynamic priority scheme, which allows application to write into the arbiters programmable registers and set the priority for masters on the bus dynamically [*Pasricha and Dutt*, 2008]. Vilas and Shyam [*V.Nitnaware*, 2010] used Programmable Priority Encoder in their arbiter and it supposed to be the most time-critical component of the scheduler design. The programmable priority encoder chosen for their design is the hybrid design which combines two simple priority encoders. Thermometer encoding is used to mask the input of one priority encoder based on the programmed priority level. Their architecture is a combination of round robin and 8X8 switch. As the context switching of the processors increases using round robin policy, the latency of the system will increase. Moreover, round robin scheme does not uses parallel method of programming.

The above discussed algorithms comes out to be quite inadequate due to the starvation of low priority components and latency of high priority components which results in overall system degradation. Following are some efficient arbitration techniques which keeps user oriented as well as system oriented requirements into consideration.

- Static lottery bus architecture
- Dynamic lottery bus architecture

2.4.6 Static lottery bus architecture

Gist of this architecture is the probability based arbitration algorithm which gets implemented in the central lottery manager. This system does not possess a fixed topology, as a result various on-chip components can be interconnected in a network of shared channels. Multiple master requests for the ownership of the bus to the lottery manager. Each master is statically allotted a number of lottery tickets, as shown in figure 2.3.

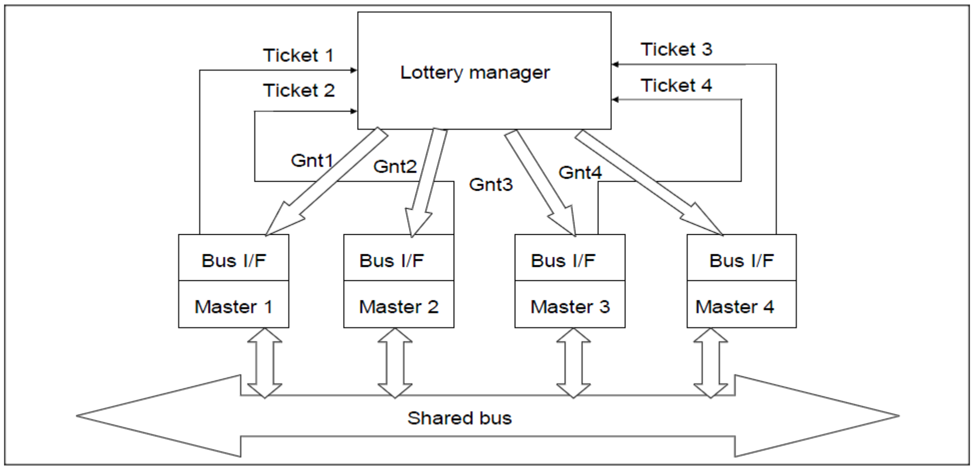

Figure 2.3 Lottery bus based architecture

The lottery manager chooses the winning master by calculating the number of lottery tickets of each master. The master which possess highest number of lottery tickets gets access to the bus with a fair bandwidth for certain number of bus cycles. In order to avoid monopolizing of bus from a master a maximum transfer size has been allocated to each master to limit the amount of bus cycles which each granted master can utilize [*Chang HP*, 2003]. In order to minimize the idle bus cycles, static lottery bus architecture pipelines lottery manager processes with actual data transfers, thus the latency is reduced. The lottery manager takes the input in the form of set of requests and the number of lottery tickets each master has got. The output is in the form of a set of grant lines which indicates that the selected master is permitted to transfer data across the bus. In order to make sure regarding any pending request the lottery manager polls the incoming request lines after every bus cycle.

2.4.7 Lottery based arbitration algorithm

Let M1,M2,M3 and M4 be the set of masters and t1,t2,t3 and t4 be the number of tickets held by each master. Suppose during any bus cycle, the group of pending bus access requests be represented by set of boolean variables r_i(i=1,2...,n) where r_i =1 if component M_i has got a pending request, otherwise r_i =0. Master associated with the component which bears the largest number of tickets finally gets access to the bus with a fair bandwidth. Probability of granting component M_i is given by

$$\text{P}(M_i)= \frac{r_i{*}t_i}{\sum_{j-1}^{n} r_j{*}t_j} \qquad (2.1)$$

Figure 2.4 shows an architecture of lottery based arbitration algorithm.

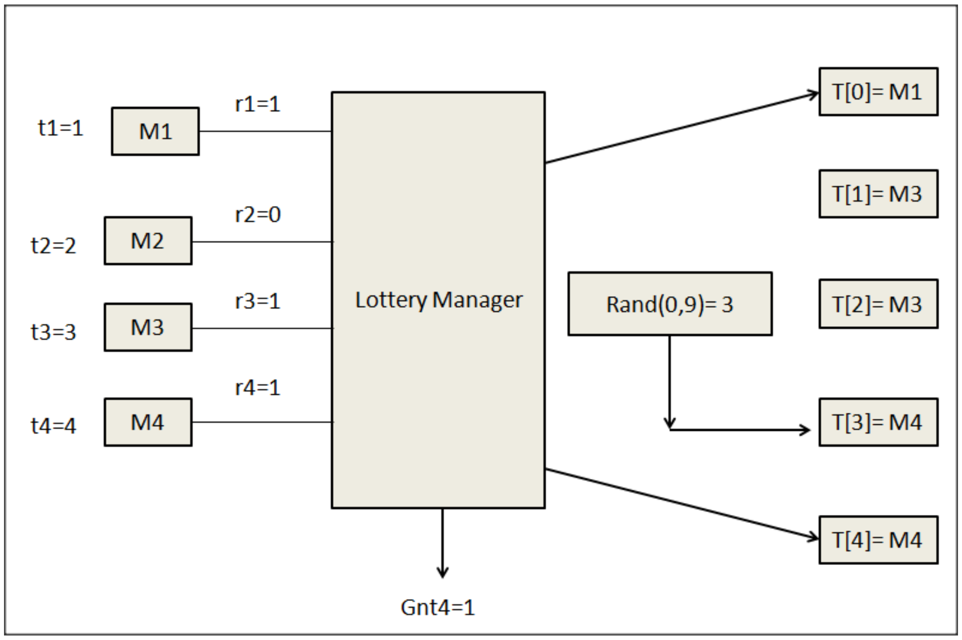

Figure 2.4 Lottery based arbitration algorithm

In order to make a decision, the lottery manager analysis the number of active tickets which the lottery system has got [*Jou et al.*, 2010]. This is given by the formula

$$\sum_{j=1}^{n} r_j * t_j \qquad (2.2)$$

These tickets are generated sequentially to each master. A pseudo-random number from the range $[0,\sum_{j=1}^{n} r_j * t_j\ ]$ is used to determine the component which is to be granted the bus. If the number is in the range $[0,r_1 * t_1]$ then the bus is granted to the component M_1, or if it is in the range $[r_1 * t_1, r_1 * t_1 + r_2 * t_2\ ]$ then the component M_2 is granted the bus. In general, if the number lies in the range $[\sum_{k=1}^{i} r_k * t_k\ \sum_{k=1}^{i+1} r_k * t_k]$, then the component $M_{(i+1)}$ is granted the bus. Since there is a sequential process involved in generating tickets to the masters, therefore this arbitration technique cannot be implemented parallely.

2.4.8 Dynamic lottery bus architecture

In this architecture, the inputs are in the form of group of request lines i.e r0, r1, r2, r3 as shown in the figure 2.5. There is a specific ticket generator module, which generates tickets to the master.

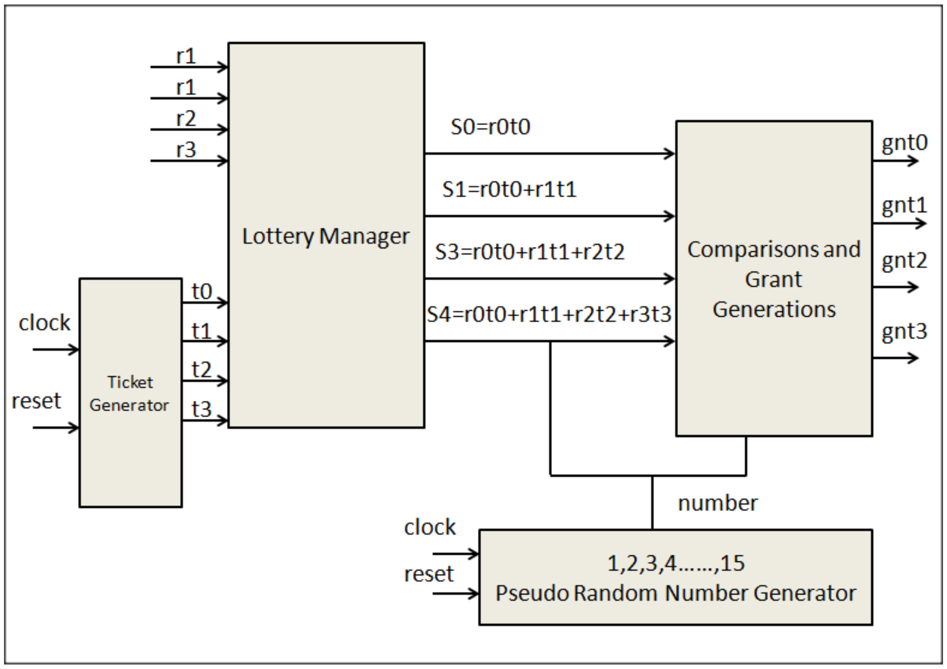

Figure 2.5 Lottery bus architecture

As the range of the ticket values are dynamic, therefore a partial sum needs to be calculated for each component at every lottery which is given by

$$\sum_{j=1}^{n} r_j * t_j \qquad (2.3)$$

The above equation is implemented using bit-wise AND and tree of adder. The final result is obtained using the range (T=r0t0+r1t1+r2t2+r3t3) in which the random number lies. This design follows the architecture of static lottery manager only with slight modifications. The distribution of random numbers in this architecture is non-uniform which is a slight limitation of this architecture [*Lahiri et al.*, 2001]. It is advantageous in a way that