Indian Institute of Technology, Madras.
Department of Electrical Engineering.

Internship Report Document

Optimized Multicast Scheduling Designs For Input Queued Switches

Practical and hardware efficient multicast scheduling for the OSMOSIS crossbar scheduler

Mohammed Shoaib

Supervisor : François Abel.
Manager : Ronald.P.Luijten.

Based on the work performed during summer 2006 at:

IBM Research GmbH.
Zurich Research Laboratory.
Sämerstrasse 4, Rüsclikon,

Zurich - 8808, Switzerland.

Acknowledgements

I thank the Almighty for His compassionate mercy and benevolence that I could complete this work in its true solidity. I would also like to express my deep sense of gratitude towards all those who have been helpful and co-operative in making this work a grand success. I am especially indebted to the enormous help and assistance of my advisor François Abel and colleagues Cyriel Minkenberg and Mark Verhappen. Their guidance was phenomenal for the completion of this work in its true flavor. I would also take the *honor* of thanking Ronald.P.Luijten, my manager at IBM, for everything which I cherished here. Finally, this thanksgiving would be incomplete without mentioning the names of all those who were participative in some way or the other in this work and with whom I spent the most memorable moments of my stay at Zurich – Daniel Hottinger, Wolfgang Denzel, Mitch Gussat, Richard Grzybowski, Henry, Markus Schneider and the whole OSMOSIS team - Thank you very much. I would also like to thank my colleagues and good friends, Andrea Bornaghi (UBS AG,Zurich), Kristijan, Marco, Anita and Ville for their help and warm company. My greatest thanks to my parents and my dearest brother for all the strength I gained from their encouragement, guidance and sacrifices. Last but not the least, my whole hearted thanks for all IBM Zurich Research Lab employees and those whom I might have missed mentioning here for all the goodness, warmth and bliss I experienced with your company. Thank you everyone again.

Optimized Multicast Scheduling Designs For Input Queued Switches

Mohammed Shoaib

August 7, 2006

Abstract

Dense multicast traffic is a natural result of increased demand for network services like audio and video distributions in todays high density networks. With such consistent demands, it seems inevitable that the volume of such traffic will continue to grow for some time to come. Today, if ATM switches are to be more popular in internet circles, either as standalone switches or as cores of high performance routers, it is necessary that they be able to handle multicast traffic with minimum hardware complexity and good efficiency. From previous works, it has been concluded that, the *concentrate algorithm* for multicast traffic, proposed in [1] personifies near perfection in performance and fairness but is really hard to implement. Schemes like *WBA* [1] provide relaxed hardware complexities for a little conceded performance. Very simple hardware schemes like *Round Robin Matching (RRM)* lead to poor latencies at higher throughputs.

In this project, we analyze the performance and hardware behavior of the popular multicast scheduling scheme, the Weight Based Arbiter (WBA) [1], as part of our research for the OSMOSIS [2, 3] project with strict hardware specifications[2]. An exploration to simplify the WBA with small compromises on fairness and performance led us to propose schemes which were more practical, simple and balanced although not totally perfect in performance. We propose multicast scheduling schemes with acceptable levels of performance close to the concentrate algorithm. Variations of the WBA design and novel schemes like WRRM, which are hybrids of WBA and RRM, in multi-cycle or multi-stage iterations, proposed here are hardware simplifying candidates for high performance switches. The design implementation of the WBA algorithm with behavioral hardware synthesis is shown to saturate the state-of-the-art OSMOSIS Multicast chip [1] with just a 40X40 Switch. Improving this misbehavior was the major motivation for this work. After several optimizations, a structural design for the WBA algorithm is proposed as an alternative, which incorporates a Binary Tree Comparator (BTC) and a Balanced Delay Adder (BDA) as the core design simplifiers. But, even this is seen to hog up the chip area soon. Novel modifications to the WBA are then proposed, where weight calculation methodologies itself are modified with simplified hardware complexity for a small bargain in performance. To make future explorations more organized, a Centralized Function Block (CFB) scheme with a Multicast Weighted Round Robin Matching(mWRRM) algorithm is proposed which is a hybrid implementation of the simple Round Robin scheme and the WBA. On the performance front, multiple iterations of the WRRM scheme are shown to produce results close to that of WBA altough keeping implementation complexities manageable. We conclude the work with a comparitive overview of the pros and cons of each of the proposed schemes.

[1] Xilinx Family : Virtex II Pro, Device : XC2VP100, package : FF1704, Speed Grade 6

Contents

1 Introduction **5**
- 1.1 The Network Switch Design . . . 5
- 1.2 More about switch architectures . . . 7
  - 1.2.1 Switch fabrics . . . 7
  - 1.2.2 Buffering strategy . . . 9
- 1.3 OSMOSIS . . . 10
  - 1.3.1 High Performance Computing Systems *(HPCS)*: . . . 10
  - 1.3.2 OSMOSIS System Overview . . . 11
  - 1.3.3 Design operatives . . . 11
  - 1.3.4 OSMOSIS summarized specifics and features . . . 12

2 The Multicast Scheduling Problem **13**
- 2.1 Preamble . . . 13
  - 2.1.1 Scheduling MC Cells . . . 13
  - 2.1.2 Buffer choices . . . 14
- 2.2 Past work on MC scheduling . . . 14
  - 2.2.1 Scheduling schemes . . . 15
  - 2.2.2 Analysis of MC Scheduling schemes . . . 16
- 2.3 The Weight Based Algorithm . . . 16
  - 2.3.1 Motivation and objectives . . . 17
  - 2.3.2 Working of the WBA . . . 17
  - 2.3.3 Analysis of the WBA scheme . . . 18

3 Tweaking the WBA **20**
- 3.1 The behavioral WBA design . . . 20
  - 3.1.1 Issues with behavioral design . . . 20
  - 3.1.2 Structural OB WBA design . . . 22
  - 3.1.3 FPGA implementation results . . . 22
  - 3.1.4 Limitations and possible improvements . . . 22
- 3.2 Optimized IOB WBA - The structural design . . . 23
  - 3.2.1 Adder types . . . 23
  - 3.2.2 The Balanced Delay Adder (BDA) . . . 24
  - 3.2.3 Hardware implementation results . . . 24
  - 3.2.4 Further tweaking - Distributed WBA . . . 25

3.2.5 The Multicycle WBA implementation 25
3.3 Conclusions and remarks on the WBA design 27

4 Alternate MC Scheduling Schemes 29
4.1 The CFB framework 29
4.2 The Round Robin Matching (RRM) arbiter design 29
4.2.1 The mWRRM perspective 31
4.3 Latency vs Throughput performance of the WRRM design . . . 32
4.4 Conclusions and remarks 34

5 WBA variations. 35
5.0.1 The OCF scheme 35
5.0.2 The LFF scheme 35
5.0.3 Mixed AF scheme 36
5.1 Latency vs Throughput results 37
5.2 HW synthesis results 37
5.3 Conclusions and remarks 38

6 Comments and closure 40
6.1 Summary of the design exploration methodology 41
6.2 Trade-off : The key to practical design 41

List of Figures

1.1 Network switches 6
1.2 General network switch architecture 7
1.3 Switching fabric classification of network switches 8
1.4 Shared Memory Switch Design 8
1.5 Shared Medium Switch Design 8
1.6 Classification of network switches based on buffering strategies . 9
1.7 An example of an HPCS 10
1.8 OSMOSIS system overview 11

2.1 FIFO buffer Input Port 14
2.2 NXN WBA scheduler connection details. 17
2.3 The WBA operation schematic 18
2.4 The WBA Input Block 19
2.5 The WBA Output Block 19

3.1 FPGA area occupancy. 21
3.2 Minimum clock period required. 21
3.3 Exemplary eight input comparator tree 21
3.4 FPGA area occupancy. 22
3.5 Minimum clock period required. 22
3.6 Sixteen Input Balanced Delay Adder (BDA) schematic. 24
3.7 FPGA area occupancy. 25
3.8 Minimum clock period required. 25
3.9 The distributed WBA. 26
3.10 The Multi-Cycle WBA Design Schematic. 26
3.11 The Multi-Cycle WBA synthesis results. 27

4.1 The Centralized Function Block (CFB) design framework. 30
4.2 CFB schemes summarized design. 31
4.3 FPGA area occupancy. 32
4.4 Minimum clock period required. 32
4.5 Latency vs Throughput performance of the mWRRM, CFB Scheme. 33

5.1 The signed subtractor is the hardware facing the axe. 36

5.2 The subtractor is replaced with a simple multiplexer. 36
5.3 Latency vs Throughput performance of the Age-Fanout WBA Schemes. 37
5.4 FPGA area occupancy. 38
5.5 Minimum clock period required. 38
5.6 OCF,LFF and AF scheme summary. 39

6.1 Summary of the design exploration development. 41

Chapter 1

Introduction

Increasing use of the internet and demands for networking of computers have led to the enormous growth of network bandwidth. As a result of this increasing demand, fast,cell-based,switched networks like ATMs have grown to the be really popular. As the networks grow, even WDM and IP applications have developed deepened demands for high speed switching. The extensive distributions of multimedia applications together with increasing demands have propelled the need for efficient and quick scheduling and switching of such huge number of cells. Multicast traffic is another manifestation of the great demand for such network services and applications. The crave for speed within the limitations of available bandwidths, has thrust the need for efficient high speed switching of such traffic. Addressing this switching problem for the growing traffic density is paramount if the Quality of Service (QoS) in such bandwidth intensive demands has to be upheld despite the soaring demands. Hardware simplicity and scalability are the watchwords for practical crossbar schedulers in any practical approach toward this issue.

1.1 The Network Switch Design

With advancements in optical technologies, heavy traffic flow these days is on the optical network. Although the all optical datapath gives a quick and speedy gateway for high-speed communication, the switching speeds of the fast network are enormously high and require an electonic controller. This constitutes the main device of focus in this work. The fast traffic flow over the optical data network is such that, there needs to be communication links established on the fly between the requesting clients and the requested hosts. The hosts may have requests from multiple clients and the clients may have requests for multiple hosts at any given point of time. This leads to contention for the seizure of the data path among various contending requests. To resolve this contention issue and grant access to a particular host to listen to a paticular client at a particular point of time, we need strategies to grant tramission(reception) access to the

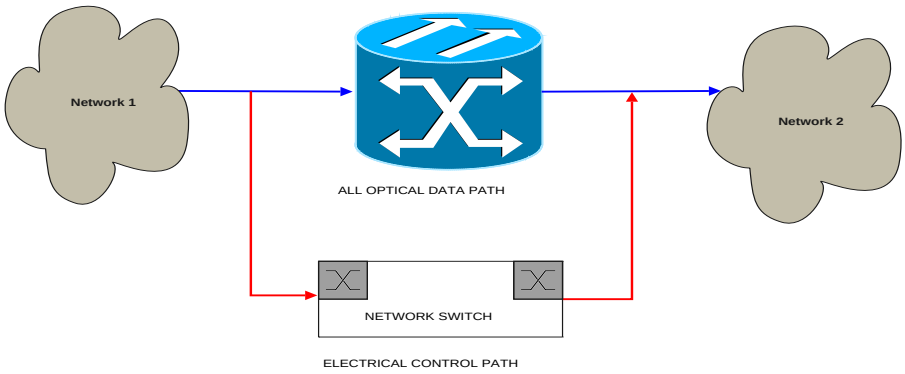

Figure 1.1: Network switches

inputs(outputs). This resolution of contending requests and grants is the main purpose and objective of the electronic controller switch. The electronic switch for any network comprises of three parts :

Input queues to buffer the cells arriving on input links.

Output queues to buffer the cells going out on output links.

The Switch Fabric to transfer cells from the inputs to the desired outputs.

Input Queues: The requests from a particular client to communicate with a host at any point of time leads to contending requests as explained above. These multiple requests arrive at the electronic controller at the same time. The electronic controller requires some time to make a connection decision. In the meantime, more requests are generated as the datapath is of very-high speed. These second phase requests have to be saved in a memory to be used in future contention resolution cycles after the first phase of requests are satisfied and no more requests remain. The input queue is essentially constituted of the waiting packets of requests, waiting to be scheduled and decided upon for accessing the datapath.

Output Queues: The scheduling decision by the electronic controller could be completed before hand and they need to be then buffered in some memory. This is the output buffer and the queued-up grants at the output buffer constitute the output queue. These are the signal packets waiting to tell the outputs in what order they need to accept data trasfer from the inputs. Only early scheduling schemes, lead to buffering of the grants queued up to be transmitted at the start of a cell-cycle.

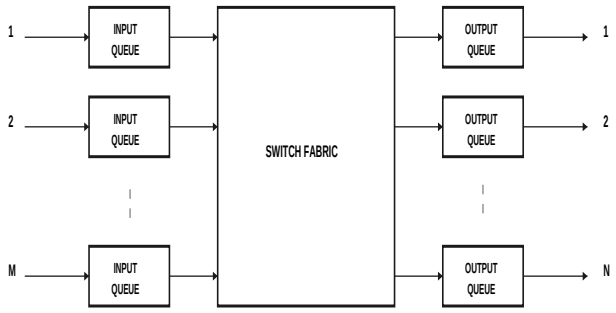

Figure 1.2: General network switch architecture

Switch Fabric: The waiting queue of request packets in the input buffers are taken into the electronic controller which then decides the connection pathway for data transfer among the various inputs and outputs (clients and hosts). The electronic controller sends the decided scheme of dataflow arrangement and then sends it to the output buffer to be recorded there and then sent forward to the outputs at every cell cycle. The electroinic controller together with the running algorithm which decides the connection methodology of the input and output ports constitutes the *switch fabric*

1.2 More about switch architectures

1.2.1 Switch fabrics

Switches (esp ATM) are classified as time division switches or space division switches based on their switch fabrics. The time division switch is further divided into shared medium type and shared memory type. Single path and multiple path switches are sub-classes of space division switches. crossbar,fully-interconnected and banyan types are categorized as single path and augmented banyan, clos, parallel banyan and recirculating type are classified under multiple path switches. Each of the switch fabric architectures have their own advantages and disadvantages.

Time division switches

Its believed that switch structure dominates more on the implementation of the switch, however the performance is governed more by the buffering strategy – their location and usage by the input and output ports. The shared memory switch uses a common memory for establishing paths between an input and an output pair.The shared medium switch comprises of a ring or a bus as the interconnecting medium. *Time-division switches are generally simpler and are*

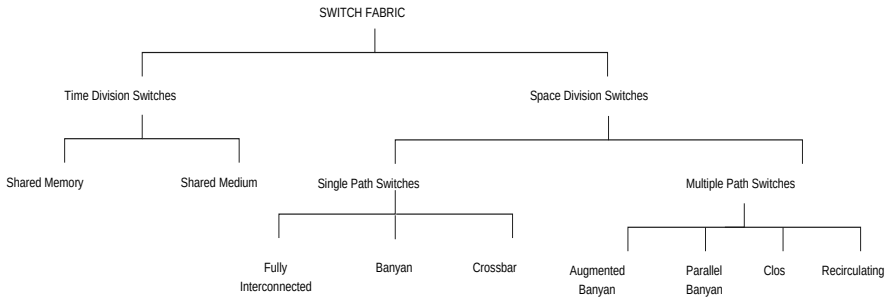

Figure 1.3: Switching fabric classification of network switches

easily extended to support multicast operation as the shared resource structure has a broadcasting nature. In a *shared memory switch*, the shared memory is logically or physically partitioned to cater to each output. Incoming cells are multiplexed into a single data stream and sequentially written to the appropriate locations of the common memory depending on their destination addresses. The routing is employed on the stored cells to produce an output data stream, demultiplexed to the outputs. Memory sharing requires less memory than dedicated memory architectures and hence they form an important design concept[6] *Shared medium switches* are preferred in the development of bus-type and ring type networks. The main drawback of shared medium switches lies in their bandwidth limitations in large-scale switches. This arises from the fact that all cells are transmitted though a single path and the network bandwidth must aggregate upto the total bandwith of all input ports. This issue is typically adressed by a bit-slice organization using multiple rings or busses. However the ultimate limitation comes from memory access speeds. Examples of shared medium swithces are NEC's ATOM (ATM Output buffer modular switch) [[7], IBM's PARIS(Packetized Automated Routing Integrated Systems) [8] and Fore Systems's ForeRunner ASX-100 switch [32]

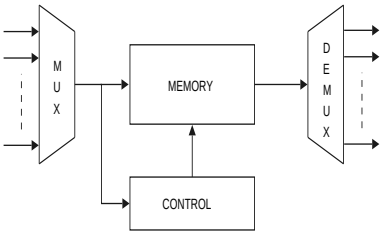

Figure 1.4: Shared Memory Switch Design

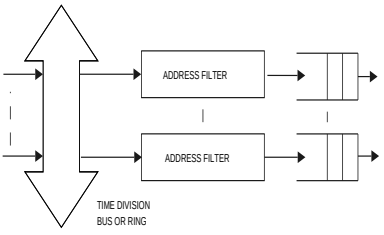

Figure 1.5: Shared Medium Switch Design

Space division switches

Multiple path space division switches were focussed on improving the performance of single packet switches but, they tend to be more bulkier than their single path counterparts and find more interest in performance dictated networks. Hardware implementation complexities dictate the use of single path switches. The *fully interconnected* switches are again redundant hardware consuming and *banyan switches* form competitors to traditional crossbar swiches. Crossbar based switches stand staggered from the banyan type in the fact that despite their large number of nodal interconnects, they are simpler for hardware implementation and acceptable in performance levels for most practical applications.

1.2.2 Buffering strategy

Various buffering strategies exist in modern switch architectures, each with its own pros and cons. The common buffering methodologies are summarized in the figure below. Externally buffers are more advantageous for the simple fact that they help to keep the switch fabric, less cluttered and simple in design. Thereby reducing the bulk of the fabric for easy portability and scaling. Shared memory buffers again stand out from their dedicated counterparts for the simple reason of effective resource utilization. Recirculating buffers, where-in, routed information within the switch fabric is temporarily stored to coerse the decision capability of the switch fabric in multiple iterations. These also have a good resource utilization profile and performance. A popular Input buffered switche is the FIFO (First In First Out) buffered switch.

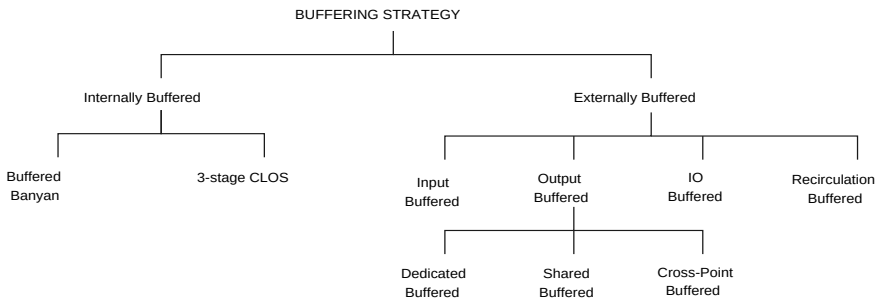

Figure 1.6: Classification of network switches based on buffering strategies

Input-buffered switches have traditionally been pitted for poor performance. It has been shown that FIFO queued switches lead to Head of Line (HOL) blocking and the throughput for unicast traffic is limited to just about 58% under relatively benign conditions [4]. With correlated arrivals, the throughput is further limited [5]. Output queued(buffered) switches have been pursued

by various researchers in the past, but memory bandwidth required in such an approach is many multiples of the line-rate. But, with limited memories, researchers had to fall back on input-queued switches. Numerous papers have exemplarily shown that higher throughputs are possible by using non-FIFO input-queuing policies [10, 9, 11, 13, 14]. The demand for network services has outgrown the increase in commercially available memory bandwidths, making input-queued switches more important for practicality and more pressing for performance. It has been researched and proposed in the past that they are good when it comes to fairness and work-conservation issues in bandwidth intensive services [1].

1.3 OSMOSIS

OSMOSIS is an optical packet switching interconnection network for *high-performance computing systems*. It aims at delivering sustained high bandwidth, very low latency and high cost-effective scalability.

1.3.1 High Performance Computing Systems *(HPCS)*:

High Performance Computing Systems are large distributed systems with several interconnected processor and/or memory nodes. Increasing processor performance and hardware concurrency in todays networks require the HPCS to perform equally efficiently, if not better. There is a grave need for sustained high bandwidth and low latency interconnection networks for all inputs to arbitrary outputs. Presently, the HPCS are implemented in the electronic domain as packet switching networks. Increased density and demands promise to peak the performance limits of the electronic switching. An all optical packet switching seems inadvertent in the foreseeable future although, technological limitations forbid this dramatic performance transgression.

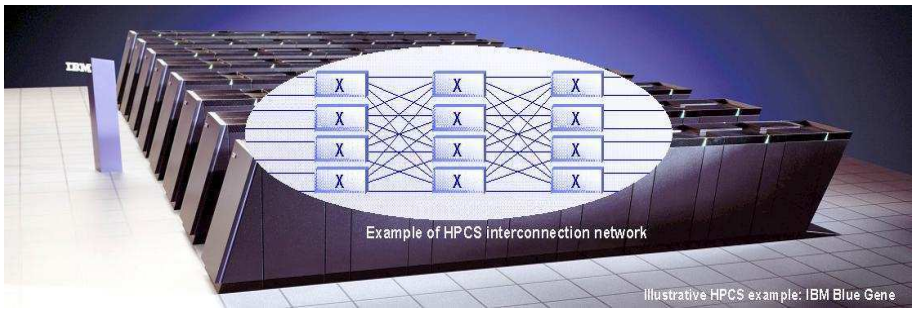

Figure 1.7: An example of an HPCS

1.3.2 OSMOSIS System Overview

The **O**ptical **S**hared **M**em**O**ry **S**upercomputer **I**nterconnect **S**ystem (OSMOSIS) project aims to address the technical challenges of HPCS and accelerate the cost reduction of an all optical packet switch. The project is a joint development 64 port HPC interconnect demonstrator of *Corning Inc* and *IBM* with all-optical datapaths operated at 40Gb/s. Based on an optical broadcast-and-select architecture that employs a combination of wavelength and space division multiplexing, *OSMOSIS* achieves a balance between cost,throughput and port count. Fast optical switching is accomplished with modern Silicon Optical Amplifiers (SOA). Electronic packet buffers at the switch input resolve temporary switch contention. A low-latency scheduler co-ordinates the transmission of packets across the optical data path and the gate timing of the SOAs. The architecture is amenable to eventual multistage scalability by means of electronic packet buffers between the stages.[1]

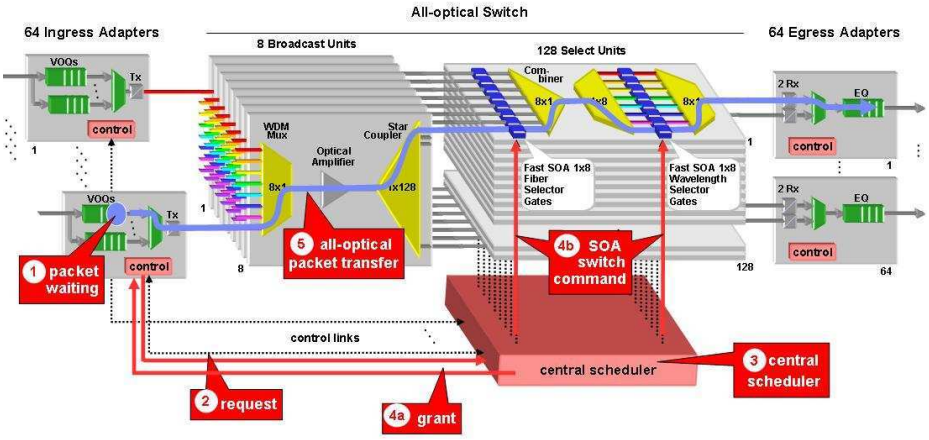

Figure 1.8: OSMOSIS system overview

1.3.3 Design operatives

Shown above is the OSMOSIS system overview. HPC nodes can send data across the all-optical switch via the ingress adapters and receive data through the corresponding egress adapters. The centralized scheduler co-ordinates contention-free data packet transfer through the all-optical data path. The all-optical switch is realized as a 64X128 fabric and each egress adapter has two receivers (Rx) for optimized performance. The optical broadcast and select switch comprises broadcast units associated with the inputs interconnected by a perfect shuffle

[1] Adopted from "Optical Interconnection Networks : The OSMOSIS Project" Ronald.P.Luijten, Wolfgang.E.Denzel,Richard.R.Grzybowski,Roe Hemenway

to select units associated with the outputs. In order to avoid packet collisions in the optical switch, it is necessary to store packets temporarily in the ingress adapters. The simplest way to do this is with a FIFO. As discussed in the previous section, the FIFO queue has its own choice reasons. The queued packets are scheduled by the contoller in a way that has a contention free routing.

1.3.4 OSMOSIS summarized specifics and features

- ▷ **Low switching overhead ($< 25\%$).**
  - Dead time for SOA switching.
  - Preamble for synchronization.
  - Packet header.
  - Forward error correction (FEC) bits.
- ▷ **Low bit error rate (10^{-21}), reliable delivery.**
  - Raw error rate target 10^{-10}
  - With single-error FEC on header and data – 10^{-17}
  - with multiple-error detection and retransmission – 10^{-21}
- ▷ **Low latency/high throughput.**
  - Optical-path delay, fast SOA switching.
  - Fast encoding/decoding – code block size compromise.
  - Fast central scheduling through pipelined implementation.
  - Virtual output queues (VOQ)
- ▷ **Scalability to 2048 nodes.**
  - 3-stage,2-level Fat Tree Topology.
- ▷ **Multicast support.**
  - Fair integrating with unicast scheduling, with no control channel overhead.
  - Independant schedulers for UC and MC traffic with filter,merge and feedback scheme.
- ▷ **Predefine scheduling rate/cell rate.**
  - Produce one high quality matching every 51.2ns.
  - Use deeply pipelined matching with parallel sub-schedulers (FLPPR).
- ▷ **FPGA only implementation.**
  - Have an FPGA only implementation of a 64X64 scheduler with an acceptable performance level.

Chapter 2

The Multicast Scheduling Problem

2.1 Preamble

The growing number of newly emerging applications such as teleservices, distance learning and IPTV on the internet has resulting in an increasing proportion of Multicast *(Abbreviated as MC)* traffic. As a result, current IP routers and ATM switches need to handle point-to-multipoint(Multicast) traffic besides point-to-point (Unicast) in current network topologies. Scheduling algoritms form critical blocks in any high speed switching system in modern times. The scheduling algoritm finds a conflict free match between input-output pairs and generates the grant signals. Designing schedulers capable of keeping up with the scalability of the switch in line speed and/or port count is a challenging and important task.

2.1.1 Scheduling MC Cells

The input-queueing structure has been a combination of the MC and UC queueing structure. The widely used unicast queueing structure has been the VOQ structure [11] since it avoids the issue of HOL blocking to a large extent [4] Maintaining such queues for multicast traffic is impractical because this requires $2^N - 1$ [24]. The FIFO queue for the MC traffic is more practical, despite their other minor drawbacks. Most of the scheduling history has been based on FIFO queues for multicast traffic [27, 15]. Other algorithms have used k queues for the multicast traffic where $1 < k \ll 2^N - 1$ [26, 25]. The major drawback of these algorithms lies in their inability to achieve high performance or run at high speeds.

2.1.2 Buffer choices

Adding small buffers inside the crossbar fabric chip of an input-queued switch has also been proposed in the literature[16]. The claim is that the presence of internal buffers simplifies the scheduling and makes it distributed. We do explore, further in our discussion architectures, which adopt internal buffers for the farbric and show their advantages in the multicast scheduling context and with an eye on practical implementation ease of the design. Although there have been attempts to design such fabrics for multicast switches [20, 22], they are more of a theoretical nature and generalized, they lack implementation results of the WBA scheme or the variations we propose hereunder.

The design exploration choice in this work was the FIFO buffer. This was because our aim was to keep simplicity as the prime objective as long as we are not paying a heavy price for it.

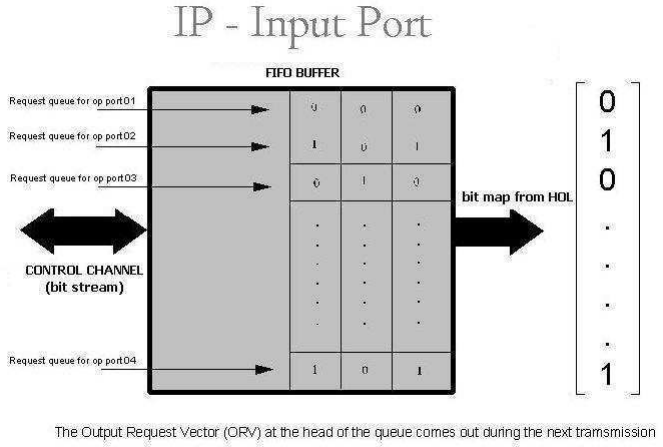

Figure 2.1: FIFO buffer Input Port

The figure shows incoming multicast cells, which are queued up in the FIFO buffer. These form the Input Port (IP) to be considered in the design architectures to follow. At the start of each iteration, the queue at the HOL pops out of the buffer, which has its fanout as the destination requests.

2.2 Past work on MC scheduling

Much of the previous work has been on independant handling of MC and UC traffic. Integrating the MC and UC scheduling under the FILM integration scheme was proposed in [28], where the MC and UC traffic was separated with a FILM(Filter and Merger) architecture. The traffic was isolated as MC and UC and the scheduling of the cells was completed independantly and then the traffic was merged again to be arbitrated upon to access the switch fabric to

produce an integrated design for the arbitration. It was shown that, this scheme outperforms its unintegrated counterparts. We stick to this operative and work with the MC scheduling part of the FILM design. Integrated scheduling of MC and UC traffic was also been pursued under works like [20, 22]

2.2.1 Scheduling schemes

FIFO queues have been notorious for HOL blocking. VOQs[11] solve the this issue to a large extent, but are more practical only for unicast traffic. practical iterative algorithms have been proposed for VOQ efficiency [23, 12]. Many unicast schemes exist under the tag of weight based schemes [19, 21] and round robin schemes [17, 18]. We draw motivation from this unicast classification of scheduling schemes to try and mix these schemes for the multicast traffic. The advantages being clearly that the round robin schemes have low hardware utilization and the weight based schemes have better performance. Thus a multicast operative with a mixture of the good aspects from each of these schemes giving a practical algorithm could be considered "The Find" of the design exploration considered here. Following are a few popular multicast scheduling algorithms.

The Concentrate Algorithm

The residue concentration algorithm [1] is the best known algorithm providing nearly ideal latency vs throughput performance. The concentrate algorithm always concentrates the residue onto as few inputs as possible. The summary of operation is captured here. [1] :

1. Determine the residue.
2. Find the input with the most in common with the residue. If there is a choice of inputs, select the one with the input cell that has been at the HOL for the shortest time. This ensures some fariness. Since an input cell can remain at the HOL indefinitely, this algorithm does not meet the fainess constraint in the strict sense.
3. Concentrate as much residue onto this input as possible.
4. Remove the input from further consideration.
5. Repeat steps (2)-(4) until no residue remains.

The TETRIS model

The TETRIS model of multicast scheduling is inspired by the popular block packing game TETRIS. The major model specifics of the design are : Each new output cell may occupy any position in its appropriate output slot as long as (i) it does not alter the Departure date stamps [1] of any other cell and (ii) It does

[1] adopted from McKeown et.al Multicast scheduling for input queued switches.

not leave any slots beneath it unoccupied. The discharge at any time is the set of output cells in the bottom-most layer and the residue is everything that is left behind.

THE TATRA algorithm

Motivated by the tetris model, the TATRA algorithm was first proposed in [29]. This was an immediate approximation of the concentrate algorithm. But as concluded in [1], the results yet seem to make it impractical for FPGA implementation.

The Weight Based Algorithm

The search for a more straight forward multicast scheduling algorithm with more implementation ease, led to the Weight Based Arbiter or the WBA. There were numerous simplifying features included in the WBA. To reduce implementation complexity, an input cell must wait in line until all thecells ahead of it have gained access to all of the outputs that they have requested. There are two popular schemes of scheduling the multicast cells [30] – the fanout splitting (or cell-splitting) and the fanout no-fanout (or cell) splitting. Because fanout-splitting is work conserving, it enables a higher switch throughput [31] for little increase in implementation complexity. Hence the WBA employs cell splitting. The WBA has an input block and an output block whose specifics are discussed in the following chapters. The connectivity of the scheduler is shown below as a quick overview of the broadcast algorithm.

2.2.2 Analysis of MC Scheduling schemes

The concentrate algorithm is the best in terms of latency vs thoughput, but its not practical. TATRA also was sacked for similar reasons. The round robin arbiter for multicast traffic is known to have poorer performance. The candidate of natural implementation choice for the OSMOSIS objectives was the WBA and its possible modifications.

2.3 The Weight Based Algorithm

An alogorithm that maximizes residue concentration with conceded fairness can starve some inputs even though it may achieve a high throughput. If an algorithm aims to be fair, it may not achieve the best possible residue concentration and thereby slumps the throughput. In order to draw a line between the choice for fainess and throughput, we need to decide upon their relative importance. The **W**eight **B**ased **A**rbiter (WBA), proposed by Mc.Keown *et.al* aims to achieve this objective.

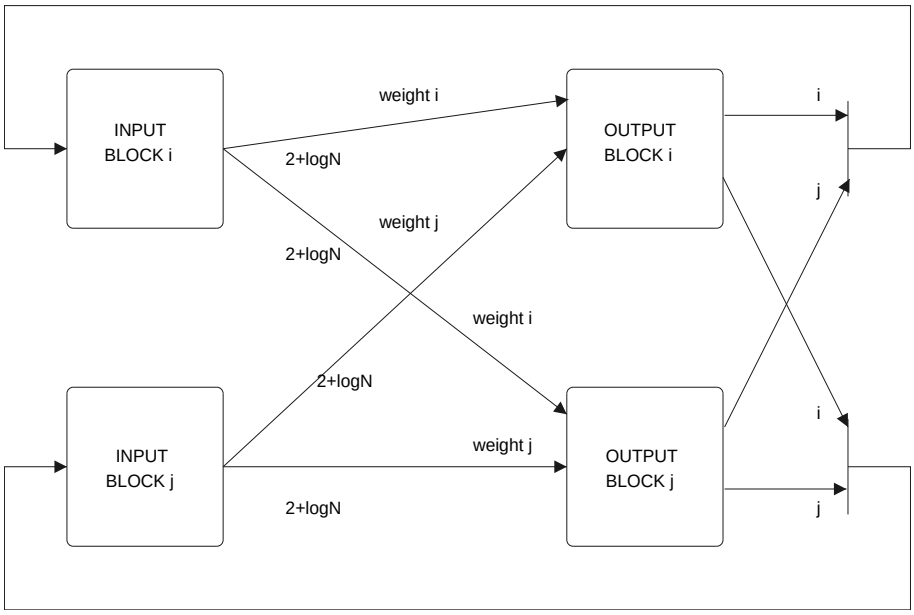

Figure 2.2: NXN WBA scheduler connection details.

2.3.1 Motivation and objectives

The motivation for the WBA was the search for a simple enough algorithm which eclipses the hardware implementation difficulties of the other algorithms like TATRA[1], which requires a collective effort for the organization of the queued packets and provides a more broader perspective of parallelism by distribution. The definition of fairness was very rigid and uniformly same for all the inputs. This kind of rule-setting does not help esp when the inputs are non-uniformly loaded or when we have a prioritized assignment objective.
The main objectives of the WBA were:

1. Simple to implement in hardware.
2. Fair to a large extent, achieving a high throughput.
3. Ablity to cope with non-uniform loading or support prioritized matching.

2.3.2 Working of the WBA

The operation of the WBA is based on assigning weights to the input cells based on their age and fanout at the beginning of every cell time. Once the weights are assigned, each output chooses the heaviest cell among the subscribing inputs. In case of multiple requests with the same weight, the scheduler grants the cells randomly. Dictated by fainess objectives, a positive weight should be given to age and to maximize throughput, fanout should be weighed negetively. Thus, older the cell, the heavier it is and larger the cell, lighter it is. Basing the choice of grant allocation on age and fanout compromizes between the extremes of pure residue concentration and of strict

fairness. If "f" is the weight assigned to fan-out and "a" for the age, then, for an MXN switch, no cell waits at the input port for more than $M + f * N/a - 1$ cell times. We can even scale the weight calculation based on the prioritizing the age/fanout in weight calculation. In particular, if we give equal weight to age and to fanout, no cell waits at the Head Of Line (HOL) for more than $M + N - 1$ cell times.

2.3.3 Analysis of the WBA scheme

Since the weight computation for each input cell doesnt depend on any other parameters, this computation can be done at each input separately and in parallel. Also, the weight comparisons at the outputs could be done in parallel. These lead to two sections or parts of importance in the WBA design *viz.*

- ▷ The Input Block (IB) which does the weight computation.
- ▷ The Output Block (OB) which does the weight comparison and selection.

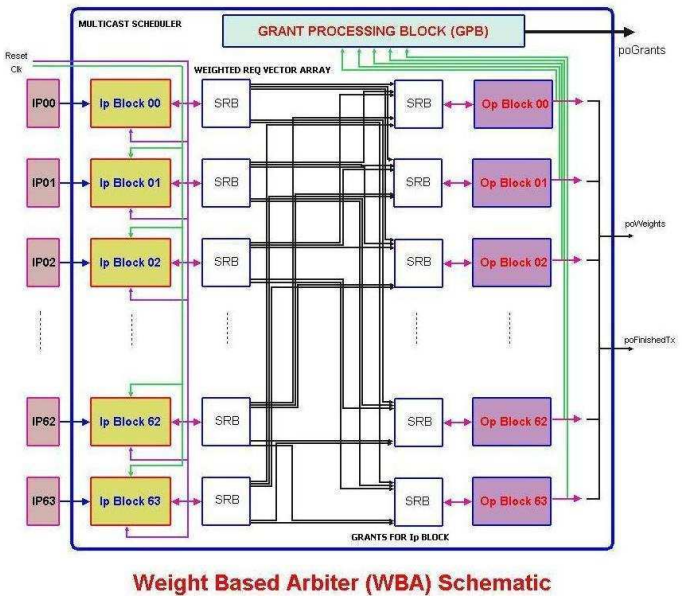

Figure 2.3: The WBA operation schematic

Hence, the implementation complexity of the WBA is only of order 1 or O(1). The hardware implementation is also straight forward and simple. The figure shows the general WBA architecture. The Signal Resolution Block (SRB) resolves the signals and arranges the weights scaled by the fanout to be sent to the output ports.

The Input Block (IB)

The Input Block or IB has its architecure as described in the figure below. The IB has to calculate the weight for each of the input cells based on their age and fanout. The

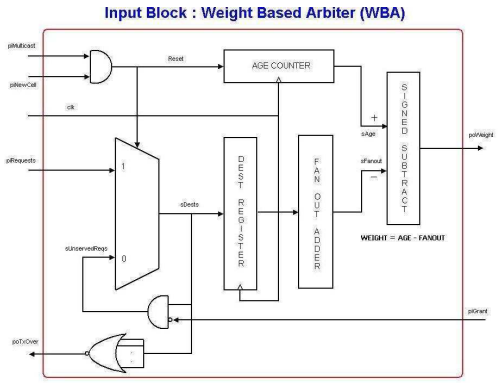

Figure 2.4: The WBA Input Block

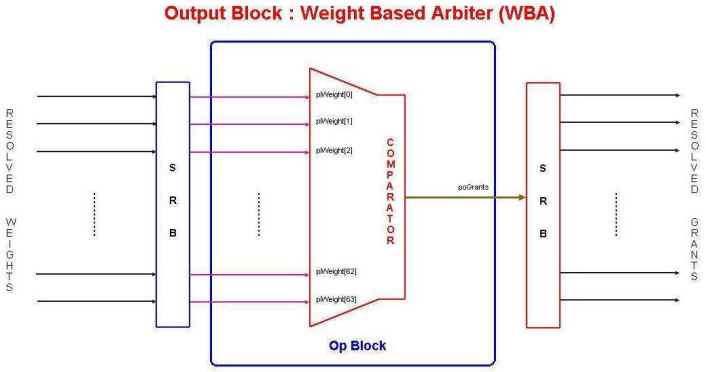

Figure 2.5: The WBA Output Block

age counter in the IB increments the age of a cell which is not served in a particular iteration. The fanout adder determines the fanout of a particular input port. The grants coming from the OB are fed back to the IB's to update their age and fanout values for the next iteration. The age counter resets, after all the requests are granted for the particular cell. Cell splitting is employed here.

The Output Block (OB)

The ouput block is a simple M-Input comparator which just takes an array of length M with each element of length $N+2$ bits. The comparator generates the output as the grant array granting the highest weight among the requesting weights in a particular iteration.

Chapter 3

Tweaking the WBA

The WBA is architecturally simple. It has the IB and OB as the basic building blocks. The WBA works with a large amount of parallelism and implementation simplicity. We implement the Weight Based Arbiter with a behavioral VHDL design. The rest of the chapter is organized as follows : we first discuss the implementation results of the behavioral design of the WBA. We then look into ways of optimizing the WBA design. We look into the OB and IB one after the other and optimize the design with a structural description of the blocks, using popular constructs for the comparator and the fan-out adder. The motivation for such a development is the poor performance of the behavioral design. Finally towards the end of the chapter, we look into alternate strategies of WBA implementation where we propose the distributed architecture which completely exploits the parallel, distributed nature of the OSMOSIS architecture and the multi-cycle implementation design. The multicycle design forms a part of the class of scheduler designs with an internal buffer.

3.1 The behavioral WBA design

The behavioral WBA simply defines the operating characteristics of the WBA in a manner specifying its behavior rather than the structure. The design is implemented on the Chip [1]. The results of the clock period and the hardware utilization are shown below.

The implemented designs were switch sizes of 2X2,4X4 upto 64X64. The resulting area occupancy and clock periods are tabulated below the graphs. The resulting performance was questionable in the sense that, the design nearly saturates the chip area only after implementing a switch of size 40X40. The clock period performance is also poor, with a minimum clock period requirement of 232.306 ns for the 64X64 switch which clearly is not acceptable, because we aim to do the WBA matching at one go in the minimum multiple of 51.2 ns.

3.1.1 Issues with behavioral design

There are two main blocks in the WBA, the IB and the OB. We looked into the OB for optimization possibilities. The main performance bottleneck was found to be the

[1] Xilinx Family : Virtex II Pro, Device : XC2VP100, package : FF1704, Speed Grade 6

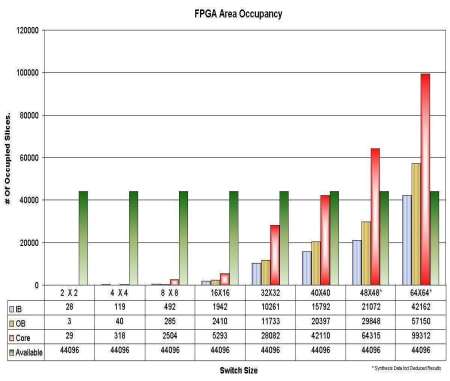

Figure 3.1: FPGA area occupancy.

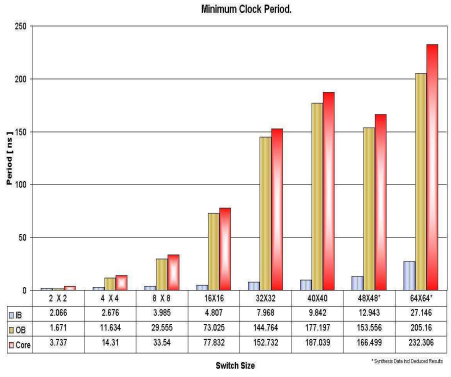

Figure 3.2: Minimum clock period required.

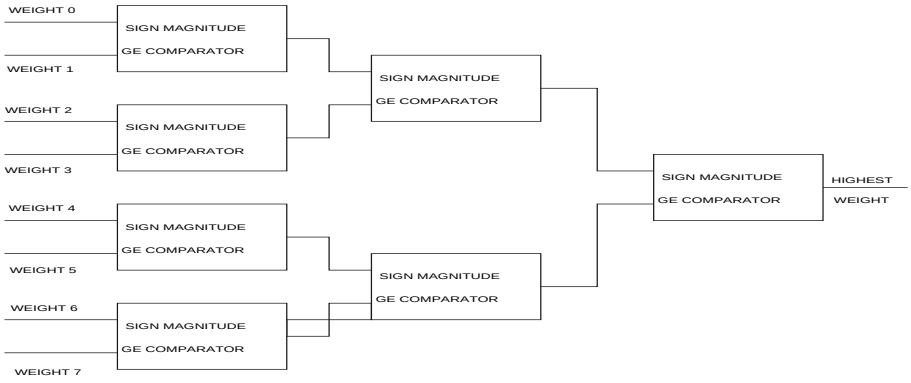

Figure 3.3: Exemplary eight input comparator tree

comparator in the output block which needed to compare the incoming weights to determine the largest among them. The behavioral design implemented the comparator as a eight bit serial comparator with 63 stages for a 64X64 switch. This was clearly an inefficient way of implementation. The immediate result of this was to explore possible alternatives for the comparator which could be implemented in a more simplistic and efficient manner.

The Binary Tree Comparator (BTC)

The 64 bit Binary Tree Comparator (BTC) was the natural choice for the implementation of the weight comparator in the OB for the simple reason that it was very efficient and also very simple with minimum design complexity. The structural design of the tree was regular with no cluttering overhead of complicated signal handling was needed. The Binary tree comparator was implemented as shown in the figure. Each of the comparator blocks were just eight bit greater-equal sign-magnitude comparators.

Shown is an illustration of the binary tree comparator. Which has eight weights as inputs and each comparator block outputs the highest weight which is routed to the next level of the tree. Finally, the tree root, outputs the highest weight among the array of the input weights.

3.1.2 Structural OB WBA design

The result of including the Binary Tree in the OB of the behavioral WBA was that, the design was more controlled. The structural description of the OB in the WBA was instrumental in obtaining a drastic improvement in the performance of the WBA design. The FPGA implementation results are shown in the plots below.

3.1.3 FPGA implementation results

Clearly, we can observe the drastic improvement in the hardware performance of the structural OB WBA. Shown are the results for varying switch sizes. The 64X64 switch fits in the FPGA though barely. The clock speed also seems to be faster now. This is a ray of hope for achieving the target objectives of the OSMOSIS specifications. The clock period of 56.245 ns for the 64X64 switch is fairly close to the 51.2 ns target, though not lesser.

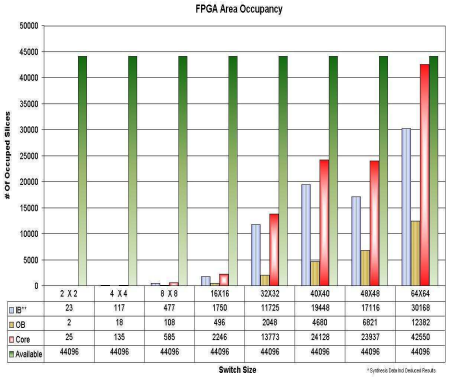

Figure 3.4: FPGA area occupancy.

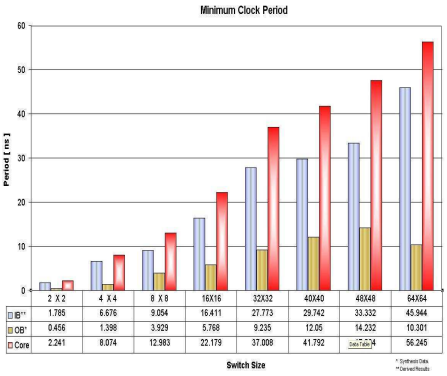

Figure 3.5: Minimum clock period required.

3.1.4 Limitations and possible improvements

The structural OB works pretty well compared to the poorer behavioral synthesis. But still there are unexploited areas of investigation in the IB of the WBA design. We look into the possibilities of optimizing the input block in the design to (hopefully) achieve more acceptable performance levels.

3.2 Optimized IOB WBA - The structural design

When we look at the IB of the WBA, there seems to be no complex hardware. But there's more to it than what actually meets the eye. The fan-out adder in the IB which adds the requests from the input port to determine the fanout value of the cell was the major focus of our investigation. This seemed to be an explicit candidate in that, this was implemented again as a serial, 5 stage adder for a 64X64 switch in the RTL(Register Transfer Level) of the design. We look at ways of implementing the adder in a more structural and efficient manner.

3.2.1 Adder types

Hardware algorithms for multi-operand adders

Array : Array is a straightforward way to accumulate partial products using a number of adders. A n-operand array consists of $n-2$ carry-save adders (CSA). Its the most bulkiest of the designs.

Wallace tree : A Wallace Tree or Wallace Adder is known for its optimal computation time when adding multiple operands to two outputs using carry-save adders. The Wallace tree guarantees the lowest overall delay but requires the largest number of wiring tracks (vertical feedthroughs between adjacent bit-slices). The number of wiring tracks is a measure of wiring complexity.

Balanced delay tree : A Balanced Delay Tree or Balanced Delay Adder (BDA) requires the smallest number of wiring tracks but has the highest overall delay compared to the Wallace tree and the Overturned-Stairs Tree. Figure 3 shows an 18-operand balanced delay tree, where CSA indicates a carry-save adder having three multi-bit inputs and two multi-bit outputs. The greatest advantage of the BDA is its implementation simplicity and comparable delay performance with the other structures.

Overturned Stairs Tree : Overturned Stairs Tree or OST Adder requires smaller number of wiring tracks compared to the Wallace tree and has lower overall delay compared to the Balanced Delay Tree. Still the implementation complexity of the design is relatively higher than the BDA.

Compressor Tree : A Compressor Tree has a more regular structure than an ordinary CSA tree made of (3,2) counters because the partial products are added up in the form of a binary tree. Yet again the tree requires a large number of odd fan-out splits to be bypass wired with the counters. The design is sequential.

Dadda Tree : A Dadda Tree is based on (3,2) counters. To reduce the hardware complexity, we allow the use of (2,2) counters in addition to (3,2) counters. Given the matrix of partial product bits, the number of bits in each column is reduced to minimize the number of (3,2) and (2,2) counters. The design is again sequential.

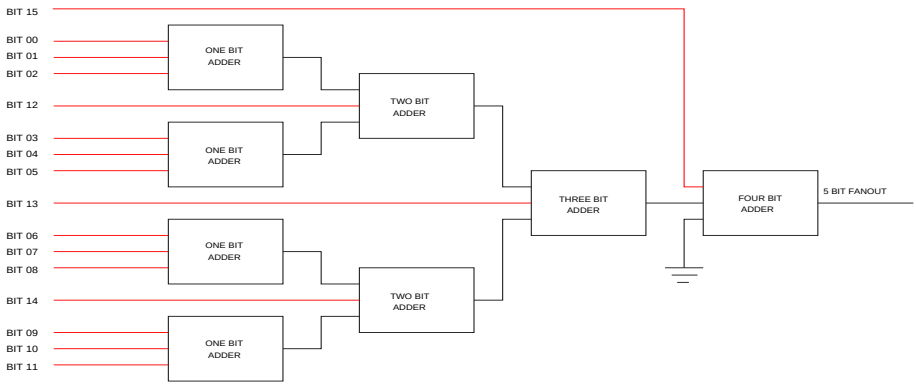

Figure 3.6: Sixteen Input Balanced Delay Adder (BDA) schematic.

(7,3) counter tree : A (7,3) Counter Tree is based on (7,3) counters. To reduce the hardware complexity, the use of (6,3), (5,3), (4,3), (3,2), and (2,2) counters is allowed in addition to (7,3) counters. Dadda Tree's strategy is employed for constructing the (7,3) counter trees.

Redundant Binary Addition Tree : A Redundant Binary (RB) Addition Tree has a more regular structure than an ordinary CSA tree made of (3,2) counters because the RB partial products are added up in the binary tree form by RB adders. The RB addition tree is closely related to (4;2) compressor tree. The RB number should be encoded into a vector of binary digit in the standard binary-logic implementation. In this generator, a minimum length encoding is employed, based on positive-negative representation.

3.2.2 The Balanced Delay Adder (BDA)

The Balanced Delay Adder shown exemplarily in the figure below was the design choice because of its low routing complexity, minimum distortion and delays and more regular structure, just perfect for easy hardware synthesis. The more fanout splitting was done, the more symmetric the design became and the choice of the adder was more justified.

Shown in the figure is a balanced delay adder with sixteen inputs. The fanout computed is five bits long. Each stage is a binary adder with an incremented number of bits. The result of each stage is passed on to the next stage to be adder to the fresh bits. The remaining bits are bypassed though the stages is a regular fashion to serve as carry-in for the forthcoming stages of the adder.

3.2.3 Hardware implementation results

The FPGA implementation results of the optimized wba are shown in the following figure. The results are really pleasing. The 64X64 switch fits comfortably in the chip area and also the timing score is outstanding compared to the initial behavioral estimate of 232 ns.

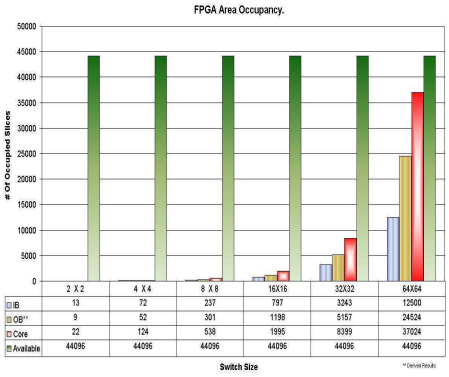

Figure 3.7: FPGA area occupancy.

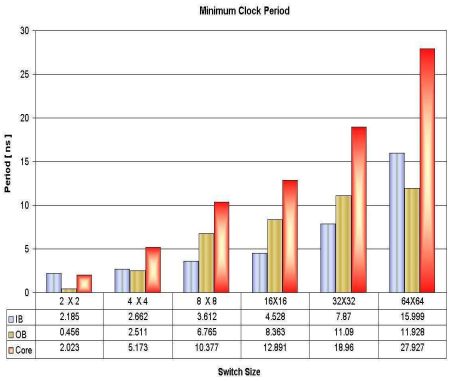

Figure 3.8: Minimum clock period required.

The design of the WBA in a structural way reduces so much of hardware implementation overheads. The design is more compact and performs at an acceptable level of performance on the scales of the OSMOSIS objectives.

3.2.4 Further tweaking - Distributed WBA

The structural implementation of the WBA speeds up the operation of the WBA design to a very large extent. The results obtained from hardware synthesis are really close to acceptable levels of hardware performance. Other alternative ways of implementing the WBA algorithm could be thought of. The distributed architecture is one such alternative, wherein, the distributed, multichip nature of the OSMOSIS controller board is exploited. The performance characteristics of the distributed WBA are expected to be the same as the structural WBA, but there would be a chip area saving due to the partitioned nature of the design. Although this is not completely correct — the reasons being that, the partitioning of the IB and OB places them on different chips and intoduces additional routing delays as overheads. This apart, the other characteristics remain the same. The traffic coming in from the LCI's over channel A and channel B are efficiently exploited in the sense that, the processing is done at the LCI itself, before passing on the control to the OB on the multicast chip.

3.2.5 The Multicycle WBA implementation

The Multicycle implementation of the WBA is another alternative. This is a variation of the WBA design with an input buffer in the switch. The auxiliary clock is an additional clock in the design. The design as such is pretty simple, but requires some heuristics to understand its performance. The input blocks calculate the weights for the requesting cells based on their age and fanout as usual in the structural IB design. Then the weights are passed on to the intermediate block, which has a comparator.

The comparator in the central block compares the weights and identifies the highest among the subscribing ones and generates a grant signal for the particular input. Then, it passes an updated weight array to be registered and reused by the same comparator block but with the weight chosen in the previous cycle being masked out. Then there

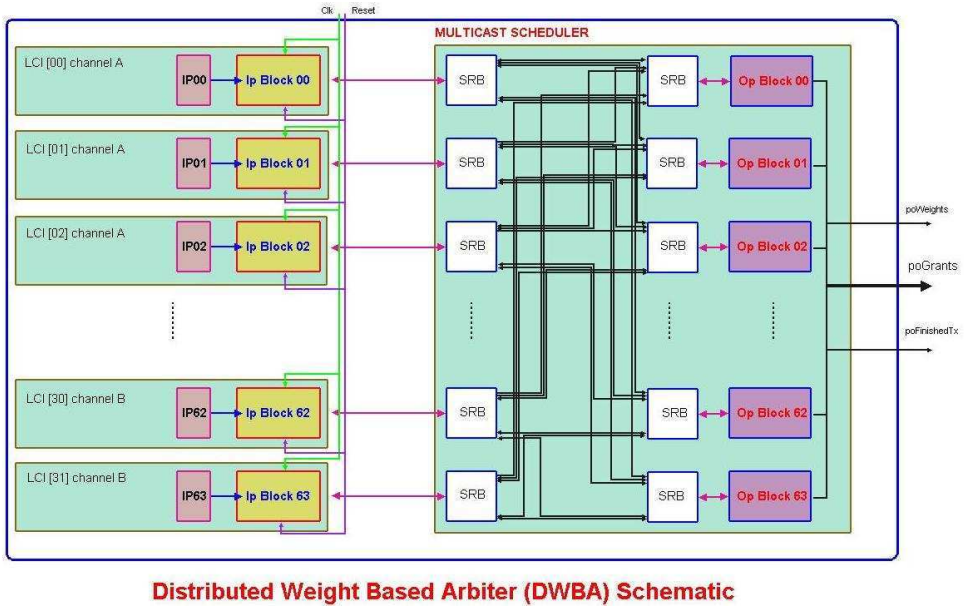

Figure 3.9: The distributed WBA.

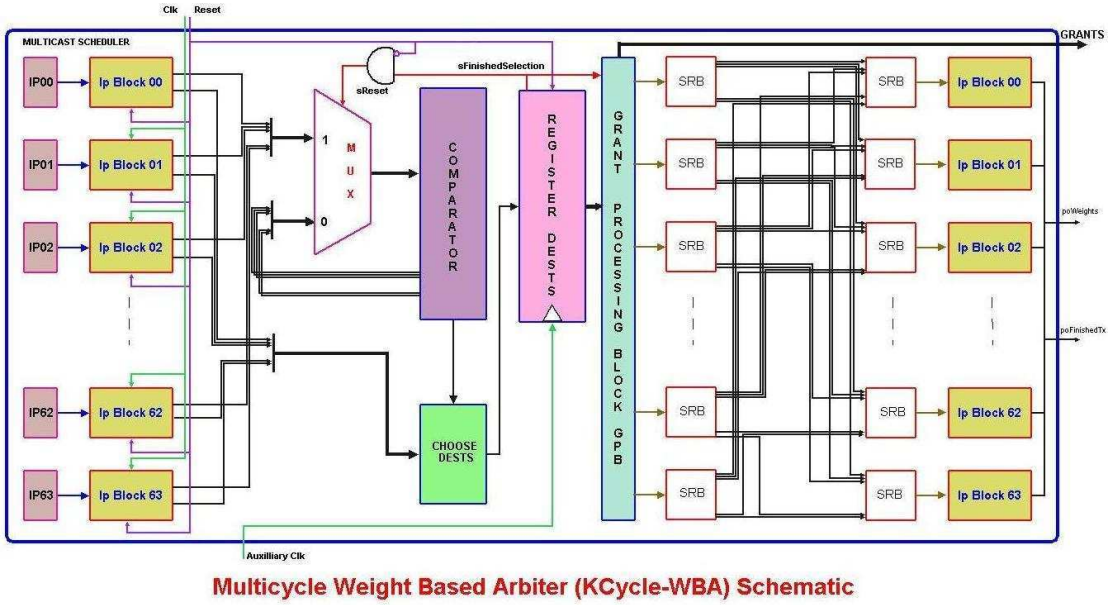

Figure 3.10: The Multi-Cycle WBA Design Schematic.

is a second iteration identifying the second highest weight, serving it, registering the grants and updating the weight array to be iterated upon in the next auxiliary iteration cycle. This cycle could be done until the matching graph is complete, in that, all the outputs have some input subscribing to it, whence a flag is passed to terminate the iteration cycle of the auxiliary clock and continue with the data transfer. This main flag, also acts in updating the age/fanout of the input cells for the next set of iteration cycles. The clear advantage of such an approach is that, the 64 comparators in the OB are avoided and are replaced by a centralized scheduler with just one comparator working through multiple iterations to generate the matching graph. However, there's no free lunch in this world. The price paid for such a convinience is an additional internal switch memory or a set of internal registers to store the intermediate grant arrays and weights to be used in the following iterations.

HW synthesis results

The FPGA implementation results of the Multicycle scheme is shown in the figure below in clear comparison with the structural/OB structural and behavioral WBA. It seems clearly an attractive choice for efficient WBA design. Although, the additional memory module in the multi-cycle design is a overhead. But this is clearly eclipsed by the huge FPGA area savings, we accomplish by avoiding the implementation of 64 OBs for just a single OB.

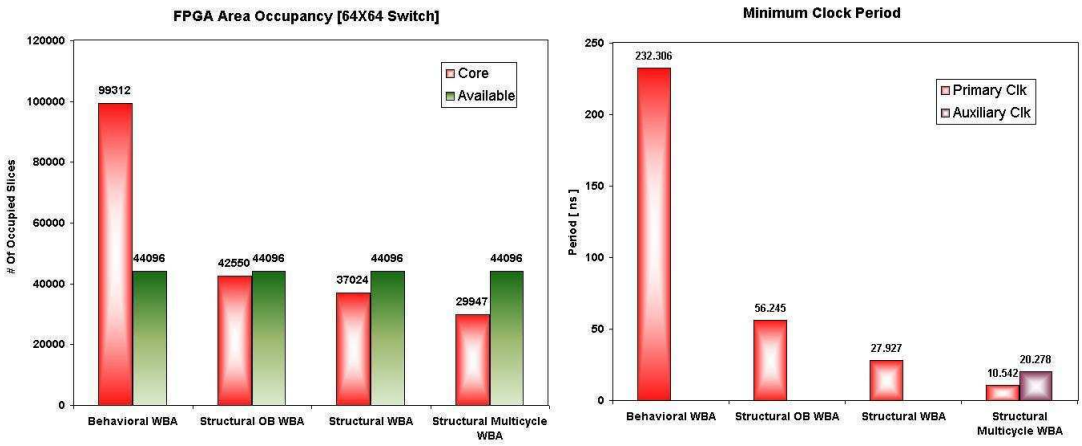

Figure 3.11: The Multi-Cycle WBA synthesis results.

3.3 Conclusions and remarks on the WBA design

The behavioral implementation of the weight based arbiter is clearly a point out of question, when it comes to the HW performance objectives of OSMOSIS. The structural OB design, showed significant promise for further enhancements, which were

completed by having a complete structural design of the WBA by including the Balanced Delay Adder in the IB. The FPGA area occupancy and clock period performance of the structural design were appealing and well within the limits of the OSMOSIS specifications. But much desired to be done, on the fact that, the design was still occupying a substantial amount of the FPGA. The distributed WBA design was then proposed which was a simple chip partitioning process, wherein, the IB's were placed on the LCIs to let the OBs have more space in the MC chip. The Multi-cycle WBA design was finally looked into. This design was really novel in that it avoided the use of 64 OBs as in the traditional design of the WBA and incorporated only a centralized comparator block with multiple iteration strategies to observe dramatic clock period reduction as well as area reduction.

Chapter 4

Alternate MC Scheduling Schemes

The previous investigation on WBA implementation and optimization led us to want for an ingerated framework of operation wherein, we could vary the scheduling scheme in a centralized way to compare its performance with the other schemes. This was the driving motivation to develop the Centralized Function Block (CFB) framework.

4.1 The CFB framework

The Centralized Function Block or the CFB is a unified design frame for comparing the various arbitration schemes. This stands for the hybrid implementation of the WBA and its variations in combination with the simple multicast Round Robin Matching (mRRM) algorithm. The kind of a scheme for MC scheduling needs to be separately distinguished from UC and MC unified schedulers.

The figure for the CFB framework needs some explanation. The Input Blocks at the left side, act like the stimuli providers for the CFB, calculating the Input Parameter Vector (IPV) – could be the weight, age or fanout according to variations of the scheduling schemes in the CFB. The Output Blocks are replaced by Programmable Priority Encoders (PPE) [12]. The Centralized function block is just a comparator to determine the highest among the IPV's and then outputs the Output Pointer Index (OPI) which acts as the pointer index for the PPE. The PPE receives the OPI which tells the Round Robin Pointer in the PPE to prioritize a particular request and jump to it. Hence, there is the partial WBA operation in the CFB and programmable RR in the PPE. The grant vector is thus updated after multiple iterations and sent back to the IB's to update the IPV for the next iteration cycle.

4.2 The Round Robin Matching (RRM) arbiter design

The Round Robin Arbiter is the most simplistic design for arbitration. Although, it seems very simple in the VHDL code, the implementation results are surprising in

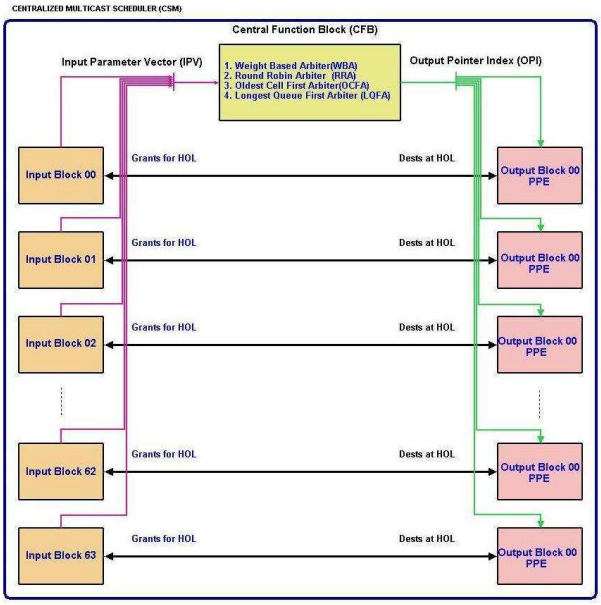

Figure 4.1: The Centralized Function Block (CFB) design framework.

that, the resultant hardware logic is comparable to the WBA, but still lesser. The scheme, is a very simple one, wherein, a pointer moves in a round robin fashion among the requesting MC destinations. It then grants the requests in a cyclic fashion. The grant vector may previously be completed partially as in the CFB scheme or maybe incomplete.

Implementation results of the RRM The Round Robin scheme implementation has been shown to perform the worst among all the proposed schemes so far when it comes to throughput versus latency performance. But the hardware implementation results as expected, yield a lower area occupancy than the WBA. The round robin selected weight in the CFB gets preferentially higher position and is passed on as the pointer to the PPE which runs a second round robin on the incoming destinations, keeping in preferential respect, the prior round robin selection done by the CFB. Thus the selection of the grant vector is completed.

Hybrid design methodologies The immediate objective was to keep the hardware advantages of the RR and the performance highlights of the WBA, doing away with their respective disadvantages in performane and hardware inefficiency. Thereby, leading us to an hybrid model incorporating the WBA and the RR in a mixed fashion. The CFB provides the perfect framework for the operational comparison of these policies. This was the prime motivation for the CFB and the hybrid design methodology.

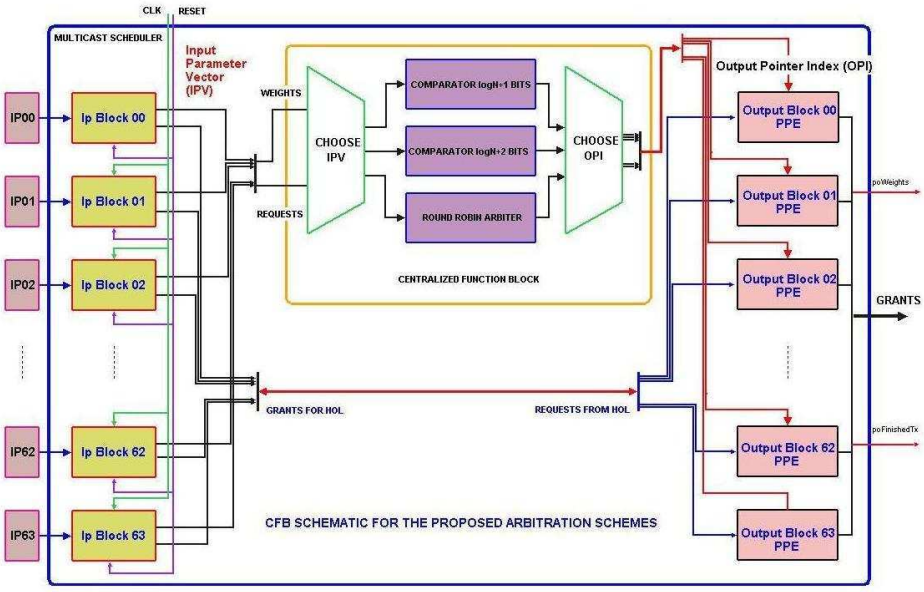

Figure 4.2: CFB schemes summarized design.

4.2.1 The mWRRM perspective

The operation of the MC Weighted Round Robin Matching (mWRRM) is very similar to the one explained above. Except that, the CFB now selects the highest weight among the IPV's which are calculated by the IB. Then the pointer is sent to the PPE to run a round robin on the incoming destinations, giving preference to the preselected weights by the CFB. Thus the operation is a mixture of the CFB and the WBA but only the highest weight is selected by the CFB and granted the OPI.

Multicycle implementation

This kind of an approach with the mWRRM brought us to the point of thought, where we questioned the operability of the CFB, which was selecting just one weight among the incoming weights and passing the OPI to the PPE. The main point of contention was that, we wanted to use the operation of the CFB more than what it was being used. The multicycle implementation of the mWRRM does precisely this kind of an operation. Here, the CFB selects the weights which are incoming at the OPI's from the IB in a multicycle fashion, more like the multi-cycle WBA, but not always until the grant array is complete. There are a preset number of iterations, upto which this scheme lets the CFB run with an auxiliary clock, masking out the highest weights every cycle before the new iteration. The multicycle implementation, clearly leads to more effective utilization of the CFB, but adds more logic as an internal buffer to the switch. Although, no hardware is reduced as in the multi-cycle WBA case, where the 64 OBs were replaced with just one OB.

HW implementation results Implementation of the WRRM in the FPGA had expected results. The chip occupancy was more than the WBA for reasons explained in the previous section. The lower clock speed, lower than the WBA was the most interesting aspect of this design. The hardware could be run faster than the WBA was a bonus for the small amount of invested hardware.

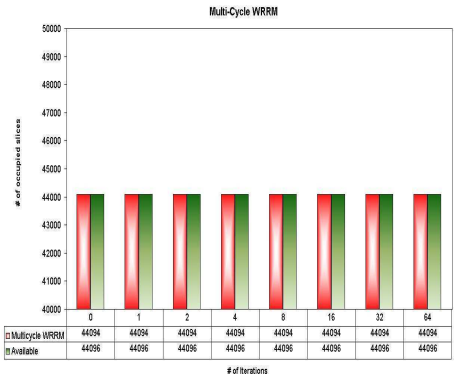

Figure 4.3: FPGA area occupancy.

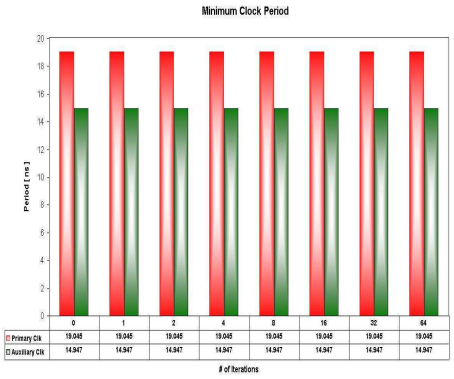

Figure 4.4: Minimum clock period required.

Multistage WRRM design

The major motivation of the multistage design of the WRRM instead of the Multi-cycle design was that, the multiple-cycle implementation of the WRRM needed an internal buffer run by an additional clock, which was giving better clock performance than the WBA, but had the drawback of higher chip area occupancy. To offset this disadvantage, the multistage architecture does away with the internal buffer *(large!)* for additional number of serial comparator introduced in the CFB chain. This design, although adds more logic, is theoretically lesser than the multi-stage WRRM, which requires an input buffer for a large number of weights and destination signals (64 vector of 8 bits lenght and 64 of 6 bits for a 64X64 switch). The operatives are heuristically estimated in this case. The development of the design, with lower amount of hardware but a slightly higher clock period again brings us to the point of our initial pursuit — A balanced trade-off between area,clock speed and performance. The natural door, which is opened next is the performance exploration of the proposed schemes.

4.3 Latency vs Throughput performance of the WRRM design

The latency versus throughput explorations Cyriel Minkenberg of IBM Zurich Research Lab, Switzerland for performing timely simulations of the designs and from which the results are adopted from of the CFB mWRRM scheme were an eye-opener. As the number of iterations/stages of the comparison in the CFB increased, the results were more and more satisfying and closer to the WBA. Of course, as seen from

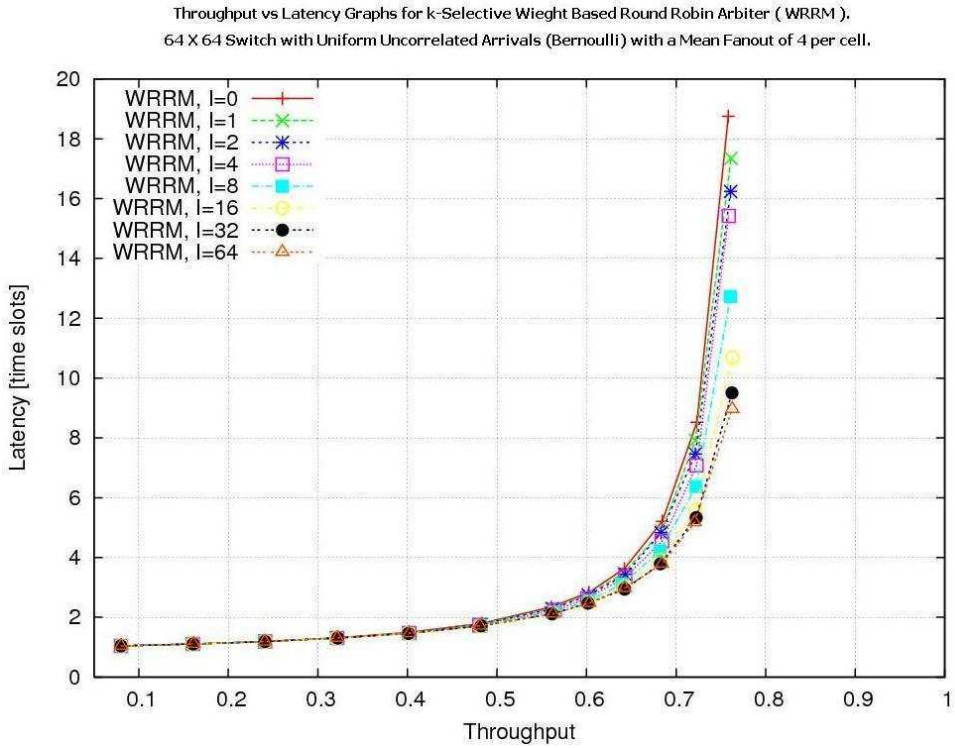

Figure 4.5: Latency vs Throughput performance of the mWRRM, CFB Scheme.

the graph, the Round Robin performs the worst on the performance front and the concentrate algorithm [1] the best. The WBA was a slight deviation from ideality for simplicity gains. The other designs of WRRM were a further trade off for simplicity and hardware performance at the expense of a digression from the WBA Latency vs Throughput. As seen from the graphs, the WRRM scheme works really well with an acceptable performance level and also with saved resources on the FPGA. After about 32 iterations (or 16 iterations – as an approximations), the WRRM heads really close to the WBA in performance. Thus instead of doing all the 64 iterations in the traditional WBA way, an alternative way could be to do just a few WBA iterations in the multicycle or the multistage way to head close on the performance score to the WBA. The rest of the selections could be round robin in the PPE.

4.4 Conclusions and remarks

The conclusions are pretty straight forward and simple from the design explorations in that, the crucial factor again is the trade-off parameter between performance, area, and clock speed. Its shown in the sections of this chapter, how a small compromise on the performance could lead us to better clock speeds and FPGA area reductions. This could be more practical in that we may not always need the same performance levels as the WBA at all times. The modulation could be significant for variable traffic arrivals, making the proposed designs more attractive and practical. The bottom line is that, the choice is ultimately based on the trade-off between the tree crucial parameters. The main driving force is the practicality of the design and the big question being, could it be implemented on the chip efficiently ?

Chapter 5

WBA variations.

We saw in the previous design explorations that, the WBA implementation was more productive and efficient when it was done in a structural manner. The structural IOB WBA saw that the area and the clock speed score were a huge improvement from the behavioral implementation. The design, of the WBA was further tuned with the distributed WBA design where the MC chip was partitioned and held just the OB and the IBs were all placed on the LCI's. The multicycle WBA implementation added an input buffer but simplified the design further. Finally in the previous chapter, we saw the hybrid implementation of the WBA when it was coupled with the round robin mathching in the CFB framework.

Alternative weight estimation schemes After exploring the design enhancements of the existing WBA scheme, its more interesting to explore scheduling schemes, wherein the weight estimation methodology itself is altered. The calculation of the WBA had in its IB, the hardware structure as described previously. We explored possibilities of structural optimizations of the IB Fanout adder and the comparator in the OB. The other evident candidate of out optimization objectives is the signed subtractor in the IB. There are 64 such subtractors at the head of the IBs. Our optimization mission was fired by the bulk of this hardware.

5.0.1 The OCF scheme

The Oldest Cell First or the OCF scheme, intends to simplify the subtractor at the head of the IBs. This scheme, doesnt calculate the weights for the input cells in the prior mentioned way at $Weight = age - fanout$, but on the contrary, sends only the age of the cell as the weight to the OBs for comparison and grant generation. This scheme clearly simplifies the architecture of the WBA design, but has other repurcussions on the scheduling, which are discussed in the following sections of the chapter.

5.0.2 The LFF scheme

The Longest Fanout First or the LFF scheme, also works similar to the OCF, the only differentiating character of the LCF is that the weight is calculated based on only the fanout of the cell. Hence the age-counter in the IB is done away with in this case. Another design plus in the OCF and the LFF is that, the weights are one bit lesser

now and require just a comparator and not a sign-magnitude comparator in the OB. The LFF design is simple as the OCF, but has other implications as the OCF.

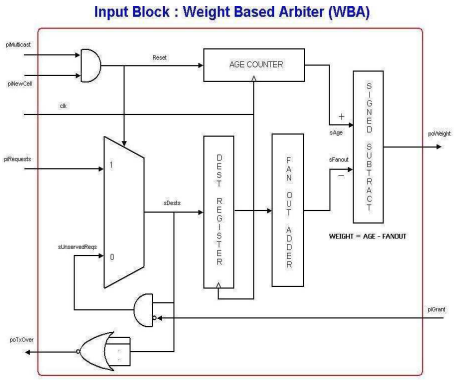

Figure 5.1: The signed subtractor is the hardware facing the axe.

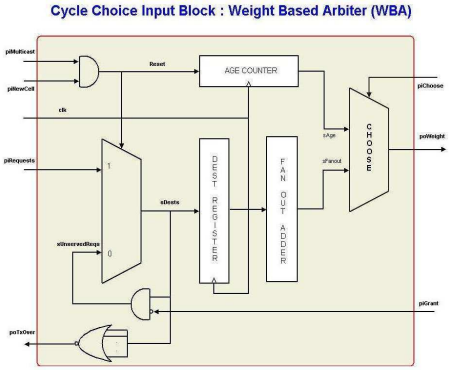

Figure 5.2: The subtractor is replaced with a simple multiplexer.

Fairness and performance issues

The main issue in the OCF and the LFF is the performance issue. Sure, there are substantial gains in the area and higher clock speeds, but the price paid is that the schemes are not fair and performing in thoughput exploitation. The OCF tends to be unfair in not serving all the queues with equal consideration and the LFF dents the throughput performance of the scheduler. Any cell with the lowest number of fanout is always served and the performance is not as good. As we already know and have seen all thorough this design exploration, there is always a price paid for any gains from the strategies proposed.

5.0.3 Mixed AF scheme

To balance the performance hampering and the hardware gains, we propose to mix the OCF and LFF schemes in multiple cycles to generate a mixed design operative Age-Fnaout (AF). The investment, compared to the OCF or LFF is that, we have to have an additional hardware to multiplex the age or fanout in specific cycles. The gains are an acceptable amount of fairness and throughput levels. When compared with the WBA, the gains are that the signed subtractor is replaced by a simpler multiplexer and the price is paid in reduced fariness. The latency vs throughput performance of the OCF is poorer than the LFF. The scheme alternative could be that we assign the weight to be the age for one cycle and to be fanout for the next three cycles. This k-cycle choice WBA could be used to exploit the good latency vs throughput performance of the LFF and include the OCF scheme for one of the cycles, thereby maintaining a balance between fairness. This kind of a mixed design is the philosophy behind the mixed AF schemes denoted by age(25%)—fanout(75%). The latency vs throughput performance is fairly good and also the hardware utilization and clock speed is acceptable.

5.1 Latency vs Throughput results

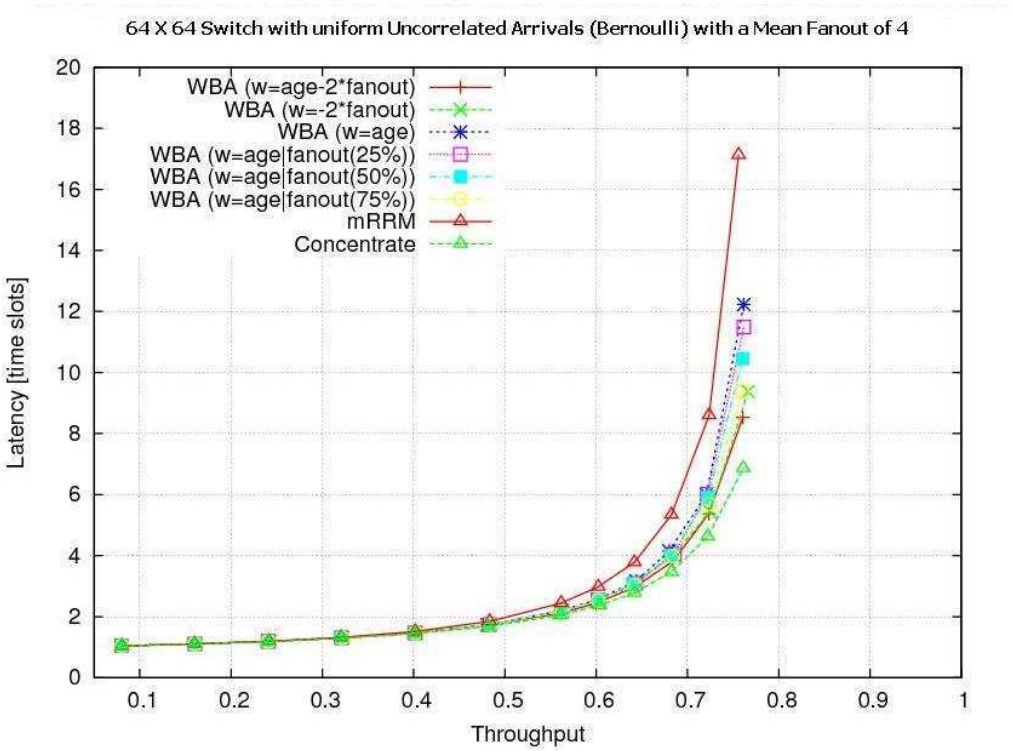

Figure 5.3: Latency vs Throughput performance of the Age-Fanout WBA Schemes.

As is clearly seen from the graphs that the age only parametrization of the weight is very poor in performance and the fanout only gives good latency vs throughput performance. Hence a balance between them is the objective. The mixed AF scheme achieves this, with a latency vs throughput curve in between the OCF and LFF curves, when the swing is uniform between the OCF and LFF. Making the switching ratio equal to 0.75 (i.e 3 cycles of LFF and 1 cycle of OCF) shift the curve more closer to the LFF curve which is actually pretty good in performance. Although, by doing this, we include the OCF occasionally and dont starve any of the contending input ports, beyond a certain minimum limitation of some multiple of 3 cycles.

5.2 HW synthesis results

The implementation of the WBA, where the weight is calculated using age and fanout in mixed cycles promises to show good hardware performance by occupying a smaller area on the FPGA chip and also having a lower clock period. The implementation

results of the above design schemes is shown below.

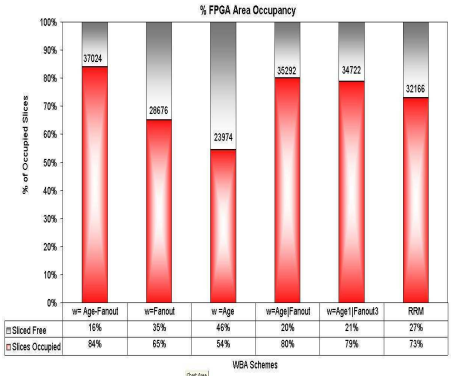

Figure 5.4: FPGA area occupancy.

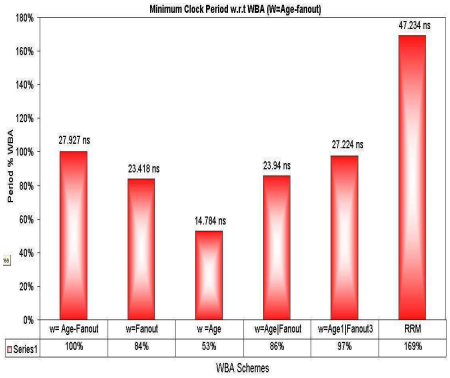

Figure 5.5: Minimum clock period required.

The age alone implementation of the weight calculation methodology gives the lowest FPGA area occupancy. The Fanout alone also occupies reasonably lesser FPGA area. But when we implement a mixed design of the LFF and OCF distributed over multiple cycles, the results are really encouraging. The FPGA area occupancy is only 79% and the clock speed is still higher than the pure WBA implementation. The synthesis behavior is not totally predictable in that, there is additional routing overheads and other signal resolution complications, which don't allow direct interpolation of the results from one scheme to another.

5.3 Conclusions and remarks

The results and resolutions from this chapter are that, instead of having a better performing but bulkier WBA, if we shift to schemes like the OCF (where we schedule the oldest cells first) or the LFF (where we schedule the largest fanout cells first), the hardware area gains are really enormous. But we pay in a way that, the OCF scheme is not fair and LFF does not achieve the maximum throughput. Thus, we resort to an intermediate approach, wherein, we assign the age to be the weight for one cycle and fanout for the other - the latency vs throughput performance of the schemes was astounding. It was really in between the good and bad curves of the LFF and OCF respectively. This gave us a compromising gateway for a hybrid implementation. This kind of an implementation was extended by reducing the switching ratio between fanout and age to be 0.75:0.25. The results were not really unpredictable. The latency versus the throughput curves of the scheme moved closer to the WBA, which was a good sign. Moreover, the FPGA area occupied and the clock period were also minimum in this case than the WBA in the ordinary sense of its application.

Summary of comparative results The results of the exploration of variations of weight calculation methodologies are summarized in the table below. They contain relative comparison estimates of area, performance, clock periods, fairness and

throughput among the various weight calculation methodologies of the modified WBA architecture.

WBA: **W = AGE – FANOUT**	FPGA Area Occupied : **Highest.** Min Clock Period : **Highest.** Throughput vs Latency : **Best.**	Highest in area/clk period but **best performance characteristics.**
WBA: **W = AGE**	FPGA Area Occupied : **Least.** Min Clock Period : **Least.** Throughput vs Latency : **Worst.**	Least in area/clk period but **Compromises Fairness** in scheduling.
WBA: **W = FANOUT**	FPGA Area Occupied : **Low. [< WBA]** Min Clock Period : **Low. [< WBA]** Throughput vs Latency : **Best.**	Low in area/clk period but is leads of **IP starvation**.
WBA: **W = AGE | FANOUT**	FPGA Area Occupied : **High. [< WBA]** Min Clock Period : **High. [< WBA]** Throughput vs Latency : **Good.**	**Moderate** in area/clk period and also in performance trade-off.
WBA: **W = AGE(1) | FANOUT(3)**	FPGA Area Occupied : **High. [< WBA]** Min Clock Period : **High. [< WBA]** Throughput vs Latency : **Worse.**	Moderate in area/clk period and is **close to WBA** in performance.
WBA: **W = AGE(3) | FANOUT(1)**	FPGA Area Occupied : **High. [< WBA]** Min Clock Period : **High. [< WBA]** Throughput vs Latency : **Better.**	Moderate in area/clk but **not as good in performance**.

Figure 5.6: OCF,LFF and AF scheme summary.

Chapter 6

Comments and closure

The main objectives of all the design strategies in this investigation are : Acceptable performance levels (without being very strict) and ease of practical hardware synthesis. The gains are higher clock speeds and reduced chip area occupancy. The watch-word all through is "Trade-off". Its sometimes a trade-off between latency vs throughput performance and FPGA area, and sometimes between higher clock speeds and reduced performance or fairness and throughput. All through this design exploration process, we try to optimize the WBA design, which we choose as our benchmark for comparisons. We started out with the behavioral WBA performing very poorly and optimized it to design the compact and controlled structural WBA. Then we looked at alternate strategies of multi-cycle WBA and the distributed WBA. With the design ideas all flowing in the direction of practical optimizations, we look into mixing the WBA with the Round Robin scheme, in the Centralized Function Block(CFB) scheme –which formed a unified investigation framework. This methodology gave us significant area reductions, for a small price in performance. We sank our teeth more deeply into the mWRRM (MC Weighted Round Robin Scheme) which is implemented using the CFB framework in a multi-cycle or a multistage way. This kind of implementation, gave us more insights into the performance characteristics of the mWRRM in comparison with the WBA. The mWRRM with iterations more than 16, tends to perform really well, although the hardware invested in such an alternative is slightly more than the WBA. Then we looked at the possibilities of weight determination using age only or fanout only or age and fanout both in different iteration cycles. This methodology simplified the IB in the WBA and was advantageous in that, it spared us some chip area by replacing the 64 signed subtractors at the head of the input cells by simple multiplexers. This approach was tempting, as the area occupied by OCF or LFF were substantially lower than the WBA, but they compromised fairness and throughput to a large extent making their puristic implementation questionable. This was the motivation for a mixed cycle implementation of OCF and LFF. This was an interesting methodology, wherein, the age was chosen in one cycle and fanout in the next three as the weights to be sent for comparison to the OBs. Also the Latency vs Throughput performance of this scheme was in very close proximity to the benchmarking WBA. This was another attractive alternative, which sure stands a considerable chance for a practical implementation scheme of the modified WBA.

6.1 Summary of the design exploration methodology

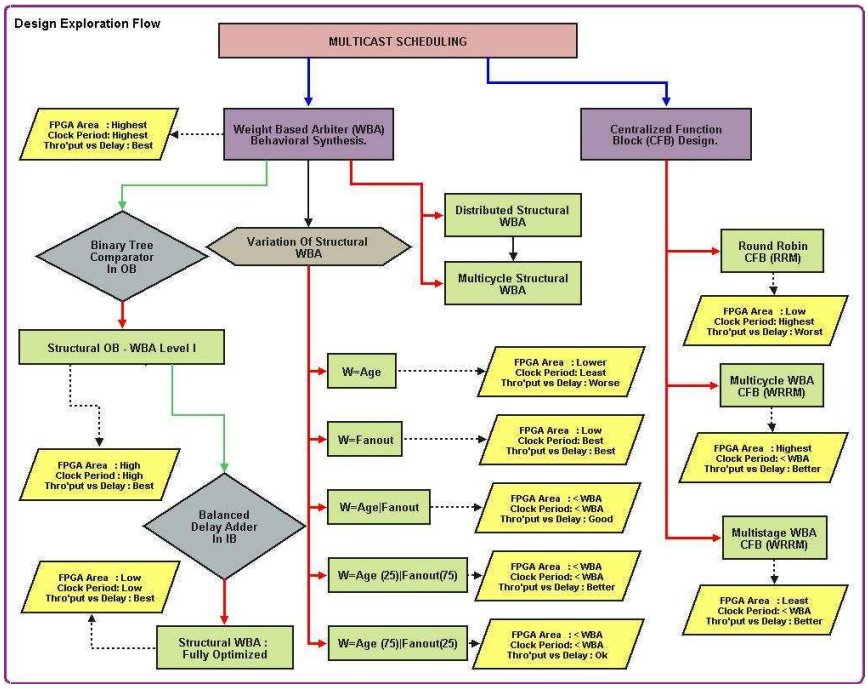

Figure 6.1: Summary of the design exploration development.

6.2 Trade-off : The key to practical design

In the complete analysis of this report, the implementation or hardware synthesis of the design was the most critical parameter. We analyzed means and methods of improvised scheduling of MC cells starting with the WBA and proposing many other schemes along the way. At the end of it all we conclude that, the structural WBA implementation is optimized enough to fit into the chosen FPGA of the OSMOSIS project – thats the solution for the project objectives. But, if we desire further optimizations or further chip area saving for more practical implementations, we conclude that, The Hybrid AF, The Multicycle WBA or The Multistage mWRRM could be potentate candidates. They exhibit good area savings and higher clock speeds than the WBA and also have comparable Latency vs Throughput performance. Thus, the key to modern design for applications with high scalablity, large switch sizes and variable incoming traffic types is implementation simplicity. The WBA is not always the only solution. Alternative solutions could do a lot more for crossbar based scheduling, without any additional hardware investment or architectural modifications for the

FIFO queue and buffering strategies. The directing pathway for design considerations is practicality of the proposed scheduling schemes, which is more often paramount than a very high performance, impractical solution. This is because that which works in practice is more useful than that which works only on paper and is intangibile.

Bibliography

[1] B.Prabhakar,N.McKeown,R.Ahuja,"Multicast scheduling for input queued switches",*IEEE J.Sel.Areas Commun.,vol.15,no.5,pp.855–866, June. 1997*

[2] Ronald.P.Luijten,Cyriel Minkenberg, Roe Hemenway, Michael Sauer and Richard Grzybowski,"Viable opto-electronic HPC interconnect fabrics",*Proc. CM/IEEE SC2005 Conference on High performance Networking and Computing, November 12–18 2005, Seattle, WA, USA, CD-Rom, page 18. IEEE Computer Society, Nov 2005.*

[3] Roe Hemenway,Richard R.Grzybowski, Cyriel Minkenberg and Ronald Luijten,"An optical packet-switched interconnect for supercomputer applications",*OSA J. Opt Netw, vol.3, no.12, pp.900-913,Oct 2004*

[4] M.Karol,M.Hluchyj and S.Morgan,"Input versus output queueing on a space division switch",*IEEE Trans. Comm, 35(12) pp.1347–1356*

[5] S.Q.Li,"Performance of a nonblocking space-division packet switch with correlated input traffic",*IEEE Trans. Comm, vol.40, (no.1) : pp.97–108. Jan 1992*

[6] F.Tobagi,"Fast packet switch architectures for broadband integrated services digital networks",*Proc.IEEE, vol.78, no.1: pp.133–167. Jan 1990*

[7] H.Suzuki,H.Nagano and T.Suzuki,"Output buffered switch architecture for Asynchronous Transfer Mode",*Proc. ICC, pp.99–103.4.1, Boston, MA, June 1989*

[8] I.Cidon *et.al*,"Real-Time packet switching : A performance analysis",*IEEE J.Sel.Areas Commun.,vol.6,no.9,pp.1576–1586, Dec, 1988*

[9] H.Obara,"Optimum architecture for input queueing ATM switches",*Elect.Letters, pp.555–557, 28 March 1991*

[10] N.McKeown, P.Varaiya and J.Walrand,"Scheduling cells in an input queued switch ",*IEE Electronic. Letters, Dec* 9^{th} *1993, pp.2174–5*

[11] N.McKeown,"Scheduling algorithms for input-queued cell switches",*Phd.Thesis, University of California at Berkeley, May 1995*

[12] N.McKeown,"iSLIP scheduling algorithm for input-queued switches ",*IEEE Trans. on Networking, vol7, no.2, pp:188–201, Apr 1999*

[13] H.Obara,S.Okamoto and Y.Hamazumi,"Input and output queueing ATM switch architecture with spatial and temporal slot reservation control" , *Electr.Letters,* 2^{nd} *Jan 1992, pp.22–24*

[14] M.Karol,K.Eng, H.Obara,"Improving the performance of ATM Queued packet switches",*INFOCOM '92, pp.110–115* :

[15] M.Andrews,S.Khanna and K.Kumaran,"Integrated scheduling of unicast and multicast traffic in an input-queued switch ",*IEEE INFOCOM, pp.1144–1151, 1999.*

[16] M.Nabeshima,"Performance evaluation of combined input and crosspoint queued switches ",*IEICE Trans. on Commun. vol.B83-B, no.3, March.2000*

[17] R.Rojas-Cessa, Z.Jing, E.Oki and H.J.Chao,"CIXB-1 : Combined Input One cell crosspoint buffered switch ",*IEEE HPSR, pp.324–329, 2001*

[18] K.Yoshigoe and K.J.Christensen,"A parallel-polled virtual output queued switch with a buffered crosspoint ",*IEEE workshop on High Performance Switching and Routing, pp.271–275, 2001.*

[19] T.Javadi, R.Magill and T.Hrabik,"A high-througput algorithm for buffered crossbar switch fabric ",*IEEE ICC pp.1581–1591, June 2001.*

[20] L.Mhamdi and M.Hamdi,"Scheduling multicast traffic in internally buffered crossbar switches ",*IEEE ICC pp.1103–1107, June 2004.*

[21] L.Mhamdi and M.Hamdi,"MCBF : A high-performance scheduling algorithm for buffered crossbar switches ",*IEEE Commcn. Letters, vol.07, no.09, pp.451–453, Sept 2003.*

[22] S.Sun,S.He,Y.Zheng and W.Gao,"Multicast scheduling in buffered crossbar switches with multiple input queues ",*IEEE HPSR. pp. 73–77, May 2005*

[23] T.Anderson, S.Owicki, J.Saxe and C.Thacker,"High speed switch scheduling for Local Area Networks ",*ACM Trans. on computer systems, pp.319–352, 1993.*

[24] M.A.Marsan, A. Bianco, P.Giaccone, E.Leonardi and F.Neri,"Optimal multicast scheduling in input queued switches ",*IEEE ICC, 2001.*

[25] S.Gupta and A.Aziz,"Multicast scheduling for switches with multiple input queues ",*Proc. of Hot Interconnects, pp.28-33, 2002.*

[26] A.Bianco, E.Leonardi, Neri.F, Piglione.C and P.Giaccone,"On the number of input queues to efficiently support multicast traffic in input queued switches ",*IEEE, HPSR, pp.111-116, June 2003.*

[27] N.McKeown,"A fast switched backplane for a gigabit switched router ",*Business Commn. Rev. vol.27, no.12, 1997.*

[28] E.Schiattarella and C.Minkenberg,"Fair integreated scheduling of unicast and multicast traffic in an input queued switch",*IEEE*

[29] Prabhakar.B and McKeown.N,"Designing a multicast switch scheduler ",*Proc. of the* 33^{rd} *Annual Allerton Conference, Urbana-Champaign, 1995.*

[30] J.F.Hayes,R.Breault and M.Mehmet-Ali,"Performance analysis of a multicast switch ",*IEEE Trans. communication, vol. 39, no.4, p.581–587, April 1991.*

[31] Joseph.Y.Hui, Thomas Renner,"Queueing analysis for multicast packet switching ",*IEEE Trans.communication,vol.42, no.2/3/4, pp.723–731, Feb 1994.*

[32] MPR Teltech Ltd, $AtmNet^{TM}$ User's Manual, Aug, 1992.