FPGA Implementation of a Modular and Pipelined WF Scheduler for High Speed OC192 Networks

Abdallah Merhebi
a_merheb@ece.concordia.ca

Otmane Ait Mohamed
ait@ece.concordia.ca

Department of Electrical Engineering and Computer Science
University of Concordia,Montréal
Québec, Canada H3G 1M8

ABSTRACT

In this paper we propose an FPGA implementation of a multi protocol Weighted Fair (WF) queuing algorithm able to handle variable length packets targeted for Packet Over Sonet (POS) interfaces and ideal for the design of hybrid IP/ATM switches. Our contributions is an extension to an existing 4 channel scheduler architecture that combines the Highest Value First scheme and Round Robin scheme, to a modular multi channel scheduler design. The improvement we offer here compared to the previuous implementation is that we have used the existing 4 channel core module to build a higher order WF queuing system without decreasing its overall performance . As a result, our scheduler is general enough to accommodate ATM (UTOPIA Level3/4) , POS Phy Level3 (or PL3 for OC48) as well as POS Phy Level4 (or PL4 for OC192) interfaces.

Categories and Subject Descriptors: Hardware, RTL design, Control design.

General Terms: Algorithms, Design, Performance.

Keywords: FPGAs, WF scheduler, Pipeline, ATM,POS, PL4, OC192.

1. INTRODUCTION

The rise of the Internet has created the need of implementing high performance and scalable switches and IP routers able to handle different types of traffic streams (voice, data and video) over IP. The need for traffic management is essential in the Internet world since the nature of internet traffic is bursty in contrast to a fixed timed slot traffic used in TDM switches. In the mid 90's the implementation of ATM (Asynchronous Transfer Mode) switches was the first initiative to address network congestion by splitting different traffic streams into different classes of services with different priority levels. Furthermore ATM was the first medium used to converge Multimedia and Data traffic in fiber optic SONET/SDH networks. This is achieved by using ATM as an intermediate Layer 2 protocol. In the late 90's, new standards for carrying IP packets directly over SONET/SDH has been developed by the IETF (Internet Engineering Task Force) and OIF (Optical Internetworking Forum)(IP over SONET versus IP over ATM over SONET). This new standard became POS [3, 4] (Packet over SONET/SDH). The advantage of POS compared to ATM is its ability to map directly IP packets over SONET via PPP/HDLC framing and hence reduce the packet overhead. In addition the POS can map packet of variable lengths whereas ATM is fixed to 53 bytes. For instance, the multi protocol frames could be IP, Frame Relay or ATM itself. However traffic management on POS is still in its early stages. Many companies use POS with their own proprietary traffic management scheme in contrary to ATM where all the implementations are derived from the ATM Forum . The new trend in IP switches is the use of Multiprotocol interfaces (POS/ATM). Therefore, it is essential that the traffic management and queuing algorithms are transparent to these protocols, which is that they can handle ATM cells as well as POS frames transparently.In general the WF scheduler is based on:

1. An interface for transmitting and receiving packets between the external PL4 chip and the scheduler datapath (e.g, The Xilinx Source-Sink IP cores [5]).
2. The 4 channel scheduler core.
3. The datapath where packets are stored and then scheduled.
4. The Status interface that monitors the error control information of packets that are tramsmitted and received on the PL4 bus.
5. The calendar programming interface that sets the sequence at the scheduler initialization phase. This sequence will dictate the channel status reporting order on the transmit and receive sides of the scheduler.

In this paper, we propose an implementation of the scheduler core. Our new implemented Weighted Fair (WF) queuing algorithm is able to handle variable length packets and is targeted for POS interfaces. Several 4 channel scheduler cores can be combined together to form a higher order WF scheduler.

GLSVLSI'05, April 17–19, 2005, Chicago, Illinois, USA.

We have implemented, simulated and synthesized the proposed scheduler core targeting Xilinx Virtex-II pro series FPGA's. The rest of our paper is organized as follows: In Section 2, we introduce our proposed algorithm. Section 3 describes the detailed implementation of the 4 channel weighted fair scheduler which is the building block that can be used to implement higher order queuing systems. In Section 4, we present our synthesis and simulation results for the higher order scheduler based on a 4 channel building block, and finally in Section 5 we conclude.

2. PROPOSED ALGORITHM

The common aspect of the majority of the known algorithms is the credit computation at each clock cycle [2, 1]. The Data from the channel or service class with highest priority will be scheduled for transmission upon completion of the required credit computation. Three main Scheduling algorithms are presented in [2]:

1. *Highest Value First (HVF).* The counter with highest value is selected. This policy selects the counter that has accumulated more "tokens" than anyone else, and thus deserves to be serviced.

2. *Round Robin (RR).* The next class is selected in a round-robin fashion. This policy attempts to achieve some fairness among all competing service classes.

3. *Highest-Priority-First (HPF).* The class with highest priority channel is selected. Here the value of the channel counter is irrelevant since the importance is given to the value or the ranking of the class. As an example if class 1 has more priority than class 4 and both classes have non empty queues with class 4 having more credits than class 1, then the packet from class 1 will be chosen even if it has less credits.

Our algorithm uses a combination of Round Robin (RR) and Highest Value First (HVF). The scheduling is performed in two stages. In the first stage, the channels sharing the same weight will be serviced in a round robin scheme. In the second stage, the channel selected for scheduling will be among the channels that have been serviced in the first stage and possess the highest weight. The weight for a particular class i such that $0 \leq i \leq N$, where N is a total number of classes, is computed as a ratio between the maximal weight W_{max} and the weight w_i of class i. This ratio, say c_i, represents the number of clock cycles that a packet of service class i and weight w_i needs to wait before being delivered.

The Round Robin stage monitors all the counts c_i representing the count for channels sharing the same weight, and services the channels in each group in a round robin scheme each time the count c_i is achieved and one of the channels has a packet ready to be delivered. Than the second stage follows the highest value first scheme and will perform a final selection based on the channels that have been serviced in the first stage and have the maximal weight (or smaller count $Ci = W_{max}/w_i$ of class i). The advantage of our implementation is that once c_i has reached its limit, it will not increase and stay unchanged until the packet is delivered where again it will be reset to 0. In the implementations described in [2, 1] the credit will still increase until the packet is delivered and may result in an overflow.

Our implemntation is not protocol specific because our algrorithm is generic. For instance it can support several interfaces(IP, ATM and Frame relay). The packets are encapsulated within a common PL4 control header that contains control information on each packet (Start of packet (SOP), End Of Packet (EOP) and Error control) [5]. The algorithm stores and schedules variable length packets from its internal RAM by decoding the information in the PL4 header. The Receive and transmit interfaces really work at the lowest level and do not need to decode any higher level headers (IP, ATM or Frame Relay).

3. DETAILED IMPLEMENTATION OF THE 4 CHANNEL SCHEDULER CORE

The 4 channel scheduler core contains two main parts:

3.1 The Weight Configuration interface

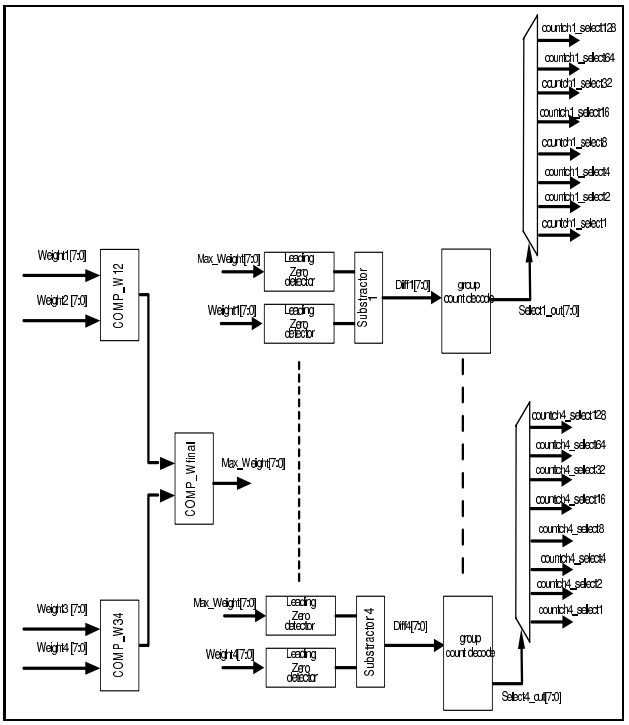

Figure 1: Overall block diagram of the Weight configuration interface

There are three major steps undertaken by the Weight configuration interface. The first is to determine the maximal weight among the 4 channels that have been configured. The maximal weight is determined using cascaded comparator stages. Once the maximal weight has been determined the second step would be to determine the ratio (W_{max}/W_i) which will activate the appropriate group counter for the channel i. For the ratio we need to determine the number of leading zeros in the weight of each channel starting from the most significant bit, then we compute the difference between the number of leading zeros of the maximal weight and the weight of the channel i. Since all weights are one of the eight possible shifts of a single 1 in an eight bit string, then it is easy to determine the weight ratio between the highest weight among the four channels W_{max} and the current channel W_i. In the last step, the ratio (W_{max}/W_i)

will lead to the selection of the appropriate group counter for each channel. Another important point to outline here is that channels having the same weight ratio (W_{max}/W_i) will be sharing the same group count. The result is that at each time the count is completed for a specific group counter the scheduler will switch the selection between the channels sharing the same count in a round robin fashion to split evenly the bandwidth between them.

3.2 4 channel Scheduler main control block

Once the appropriate group counter selection has been set for each channel by the weight configuration interface, the core scheduler control block (see Fig. 2) will enable the scheduling scheme dictated by the group counters. The core Scheduler is divided into three major functional stages:

a) The first stage is constituted of the eight group counters corresponding to all the possible ratios already discussed previously and the glue logic that detects if there is at least one packet in the buffer that is ready to be transmitted. At each time the required count is completed, the group counter enables the round robin priority encoder which will schedule evenly the active channels that share the same count and have at least one packet in their buffer that is ready to be transmitted.

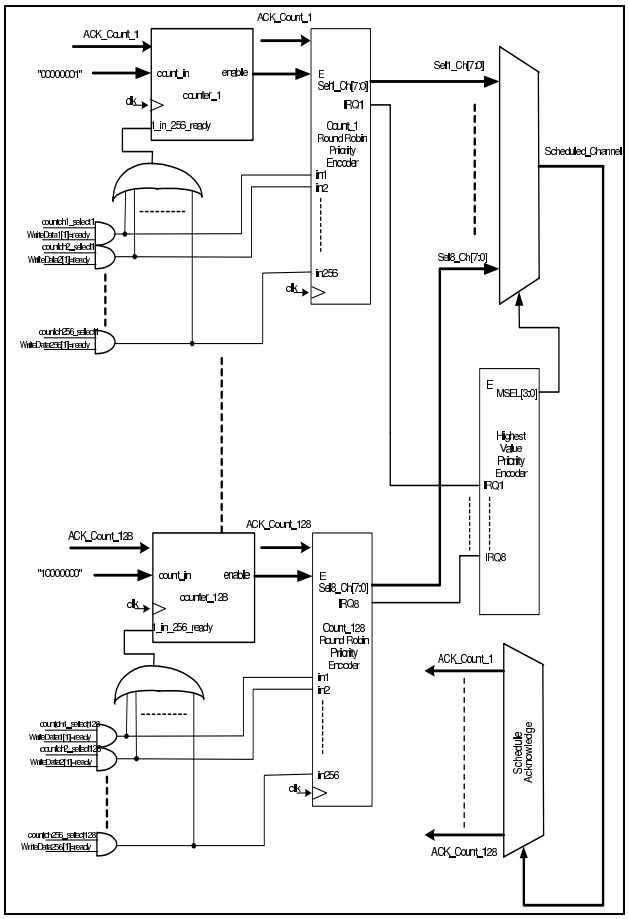

Figure 2: Block diagram of the scheduler control block

b) The second stage is the set of eight round robin priority encoders associated to each ratio counters. The round robin priority encoder will switch the selection between the channels once it receives the active enable signal from the group counter and the count acknowledgement from the scheduler Highest Value First (HVF) stage. The count acknowledgement is initiated to confirm that the packet corresponding to the channel has been transmitted.

c) The third functional block of the core scheduler is the Highest Value First stage. After each of the round robin priority encoders has made its own channel selection, the HVF stage makes the selection among the channels from the round robin stage with the highest weight.

We have implemented and tested the scheduling algorithm on a virtex II pro FPGA. The size of a 4 channel implementation is about 1994 CLB's. The design is pipelined and the delay of the maximal pipeline stage is 17ns that can run at a speed of 58.82 Mhz. Using an internal 64 bit datapath this allows the scheduler to have a total bandwidth of 3.76 Gb/s. Our next objective is to improve the design by adding more pipelining in the internal stages. This consists in breaking the RR and HVF stages into smaller pipelined stages and optimizing the internal logic such that each pilpeline stage does not exceed 7ns. This will result in increasing the speed up to 9/10 Gbps.

4. 16 CHANNEL SCHEDULER IMPLEMENTATION

The 16 channel scheduler is obtained by cascading several 4 channel WF scheduler building blocks as illustrated in Figure 3. The size of a 16 channel modular design is 10200 CLB's. Table 1 provides the theoretical and experimental results for the 4 and 16 channel schedulers.For that table we define the following parameters:

1. w_i : weight of channel i
2. Nb insts: Number of instances for each channel
3. B_{iex} : experimental bandwitdth measured
4. B_{ith} : theorethical bandwidth derived from the weight configuration

For the 4 channel scheduler we have the following weights: $(ch_1 = 8, ch_2 = 4, ch_3 = 2, ch_4 = 1)$. As for the 16 channel scheduler, we assign the same weight, say 8, for the 4 channels in each scheduler of the first stage, and we assign the wieght: 8, 4, 2,and 1 for channels (ch_1, ch_2, ch_3, ch_4) respectively.

5. CONCLUSION

In this paper we presented a scalable and pipelined 4 channel weighted fair queuing algorithm that can be used in Multi protocol IP/ATM/Frame Relay switches and handle different sized packets as apposed to the implementations in [2, 1]. In our algorithm we combine two scheduling schemes, round robin (RR) and Highest Value First (HVF). This adds more flexibility to the algorithm and can make use of both scheduling schemes or one of them depending on the type of application. Another improvement we are providing compared to the architectures proposed in the two papers [2, 1] is the ability of our scheduler to be pipelined for enhanced performance.

w_i	Nb insts	% B_{iex}	B_{ith}
Ch1: $w_1 = 8$	4281	42.86%	43.6 %
Ch2: $w_2 = 4$	2570	25.73%	29.12 %
Ch3: $w_3 = 2$	1996	19.98%	17.47 %
Ch4: $w_4 = 1$	1141	11.41%	9.7 %

w_i	Nb insts	% B_{iex}	B_{ith}
Chgroup1: $weight\_gr1 = 8$	4283	42.87 %	43.6%
Ch1: $weight\_ch1 = 8$	1093	10.94 %	11.5%
Ch2: $weight\_ch2 = 8$	1067	10.68%	11.5%
Ch3: $weight\_ch3 = 8$	1063	10.64%	11.5%
Ch4: $weight\_ch4 = 8$	1060	10.61%	11.5%
Chgroup2: $weight\_gr2 = 4$	2569	25.72 %	29.12%
Ch5: $weight\_ch5 = 8$	657	6.57 %	7.8%
Ch6: $weight\_ch6 = 8$	637	6.37%	7.8%
Ch7: $weight\_ch7 = 8$	637	6.37%	7.8%
Ch8: $weight\_ch8 = 8$	638	6.38%	7.8%
Chgroup3: $weight_gr3 = 2$	1996	19.98 %	17.47%
Ch9: $weight\_ch9 = 8$	510	5.10 %	4.36%
Ch10: $weight\_ch10 = 8$	498	4.98%	4.36%
Ch11: $weight\_ch11 = 8$	495	4.95%	4.36%
Ch12: $weight\_ch12 = 8$	493	4.93 %	4.36%
Chgroup4: $weight\_gr4 = 1$	1141	11.42 %	9.7%
Ch13: $weight\_ch13 = 8$	288	2.88 %	2.42%
Ch14: $weight\_ch14 = 8$	285	2.85%	2.42%
Ch15: $weight\_ch15 = 8$	285	2.85%	2.42%
Ch16: $weight\_ch16 = 8$	283	2.83 %	2.42%

Table 1: Experimental results measured for 10,000 clock cycles for 4 and 16 channel scheduler implementations

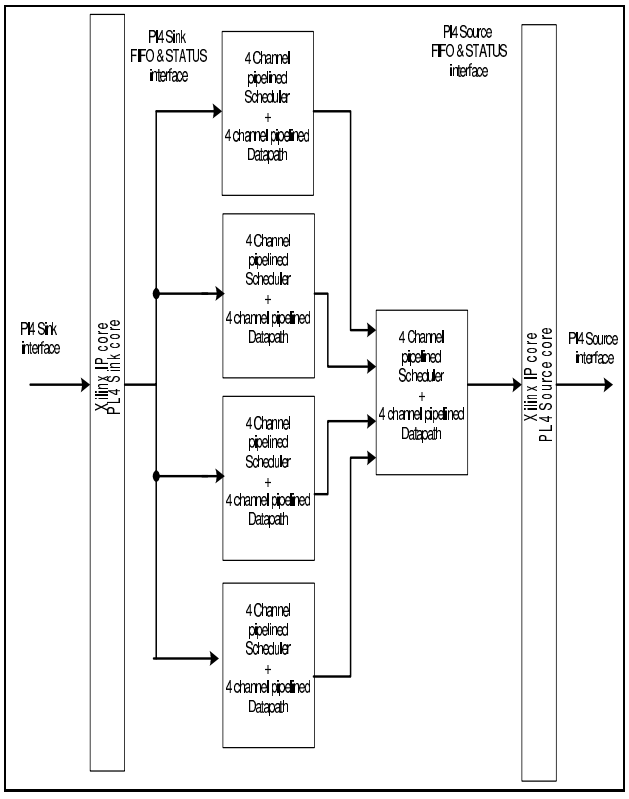

Figure 3: Scalable and pipelined 16 channel scheduler implementation

The use of pipelining opens the door to scalability to an unlimited number of channels based on a 4 channel building block and a flexible multilevel weight configuration interface.

6. REFERENCES

[1] Eddie Law K., L. Multi queue management and scheduling for improved QoS communication network. Technical report, Nortel Networks, Ottawa, Canada, 1997. http://www.cnaf.infn.it/~ferrari/papers/sched/TheBandwidt.pdf

[2] Katevenis M., Serpanos D., and Markatos E. Multi-queue management and scheduling for improved qos in communication networks. Technical report, Institute of Computer Science (ICS) and Foundation for Research & Technology - Hellas (FORTH), 1997. http://www.ics.forth.gr/proj/avg/asiccom.html

[3] Cam R. and T. Pluris. *System Packet Interface Level 4 (SPI-4) Phase2 : OC192 System Interface for Physical and Link Layer Devices.* PMC Sierra and Terabit Network systems, June 2000. http://www.oiforum.com/public/documents/OIF-SPI4-02.1.pdf

[4] T. Pluris R., Cam. *System Packet Interface Level 3 (SPI-3): OC48 System Interface for Physical and Link Layer Devices.* PMC Sierra and Terabit Network systems, June 2000. http://www.oiforum.com/public/documents/OIF-SPI3-01.0.pdf

[5] XILINX INC. *SPI-4.2 (PL4) IP core Product Specification*, February 2004.