VLSI Architecture and Implementation of Statistical Multiplexer

Abhilasha Rani Goel
Department of ECE,
Shobhit University, Gangoh, India
abhi.goel1989@gmail.com

Akhil Ranjan
Department of ECE,
JUIT, Solan, India
akhil.ranjan@juit.ac.in

Mohd Wajid
Department of ECE,
JUIT, Solan, India
mohd.wajid@juit.ac.in

***Abstract*—In networking data rate varies from application to application and usually ratio of peak data rate is much higher than average data rate i.e. bursty data transfer. Hence, service provider/Network cannot use normal multiplexing as it requires huge bandwidth with very small utilization factor, so there is a requirement of statistical multiplexer, which is based on incoming data rate statistics and efficiently utilizes the available total bandwidth. There are many applications which use this technique like asynchronous transfer mode, UDP/TCP protocol, and digital TV transmission, digital broadcasting. Generally hardware implementation is faster than software implementation, so authors have proposed VLSI hardware architecture of statistical multiplexer and implemented on FPGA using Xilinx ISE. Various modules are simulated, synthesized and implemented on FPGA. Digital operating clock frequency is also estimated for individual sub-module and integrated main module.**

Keywords—Data multiplexing; reconfigurable hardware; VLSI; FPGA; quality of service (QoS);

I. INTRODUCTION

Next Generation technology has the need of a very high speed information networks. When these networking services are built for end users, minimization of available resources without compromising on quality is an important aspect of the technological era. In this paper, statistical Multiplexer to control the user admission and QoS in terms of packet loss probability is proposed and implemented using hardware description language on Xilinx Spartan Technology. The most common form of multiplexing is that of time division multiplexing (TDM). In this multiplexing technique, each source is allocated a small time period to transmit information of fixed length. The packets available on the channel result from different sources, which send the packets simultaneously. The packets of the different sources are mixed on the channel e.g.- a packet is transmitted for first source, then a packet for a third source, next for the first, then two packets in a row for second source, and so on. This technique of mixing is referred to as "statistical multiplexing" [1].

A statistical multiplexer has the advantage that all channels will not have peak demand of bit rate at the same time. Channel which has more need gets more bandwidth and less needed channels will be allocated less bandwidth, which provides the better quality of service (QoS) [2].

In this technological era, QoS is an important parameter, which a user has to consider for each and every application. Implementing QoS issue is much more complicated because quality requirements differ from user to user. One critical problem in networking world is that how many users can be allowed in the network without diminishing the QoS levels. It is usually specified by packet loss probability parameter.

QoS networks are either deterministic or statistical. In a deterministic service, no packets are dropped in the network. This service provides highest level of quality but wastes a lot of network resources. A statistical network is based on probabilistic service such as packet loss probability< threshold. With this assumption, a statistical service has certain advantages over deterministic service by having knowledge about the statistics of traffic sources [3].

This paper presents the technical details, implementation and synthesis of statistical multiplexer. Section II provides brief review about Statistical Multiplexer. In section III, proposed VLSI architecture and its working is discussed. Section IV deals with synthesis results of integrated module and different sub-modules used in the architecture.

II. STATICTICAL MULTIPLEXER

Computer applications generate the data in a busty manner for transmission. These bursts of information are separated by long idle periods and formatted into packets that contain header which identify the source and destination. The packets are then transmitted over a communication line. If we assign a dedicated line to each terminal, it becomes highly inefficient.

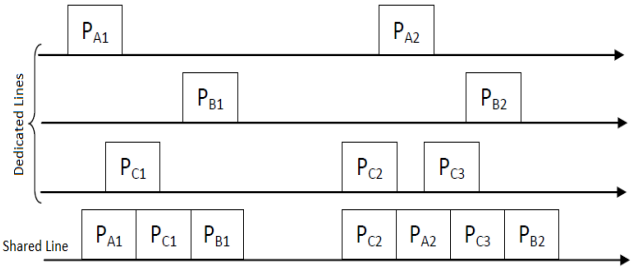

Fig. 1. Dedicated and shared lines

In "fig. 1" the first three lines show the times when the packets would have been transmitted, if each terminal had its own dedicated line at speed *R* bps. Last line shows the time when packets are multiplexed. Because the terminal generates the packets in busty manner, it is possible to combine the packet streams into a single line of speed *R* bps. Because packet generation time overlaps, the multiplexer must buffer and introduce delay in some of the packets. Thus by aggregating the packet flows into a single transmission line, the

multiplexer reduces the system cost by reducing the number of lines [4].

In general, statistical multiplexer is used in two situations: packets arrive from multiple lines for transmission to a remote site; packets arrive to a packet switch network then route some of the packets for transmission over a specific data link. In both situations the packets are multiplexed onto the given link [5].

In synchronous time division multiplexing, the dedicated slots of time are assigned to different users to transmit their packets. If a user does not have a packet to transmit in its dedicated slot, then the slot will go wasted because no other user can transmit in this time slot whether they have packets ready to transmit or not. But in asynchronous time division multiplexing no dedicated time slot is assigned to users, it allocates the time slot dynamically based on demand. The packets are generated and if the transmission line is busy, they are stored in buffer. As the transmission line becomes available, packets are then transmitted according to their position in the buffer. Typically, packets are transmitted in FIFO fashion but some applications use priority and scheduling mechanism of various types in order to determine the order of transmission. Statistical multiplexer as shown in "fig. 2" uses the asynchronous multiplexing to transmit the packets over a shared communication line [6].

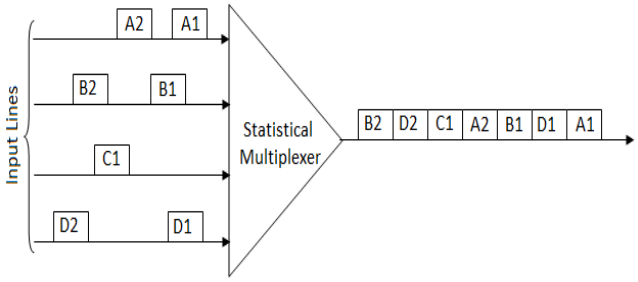

Fig. 2. Simple structure of Statistical Multiplexer

Now, suppose we have a system as shown in "fig. 3". There are M no. of sources, which have data that is statistical in nature, means they don't transmit data with constant periodic bit rate. It is variable in nature.

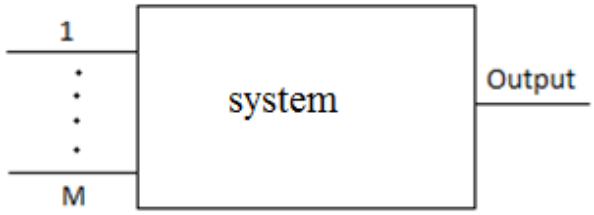

Fig. 3. An example of a system

Let the bit rate of source one is R_1, bit rate of source two is R_2 and so on, bit rate of source M is R_M. L is the maximum bit rate of output and Y is the combined bit rate of inputs, means

$$Y = R_1 + R_2 + \ldots\ldots\ldots\ldots\ldots + R_M$$

Suppose new user wants to enter in the system. The network problem is to determine whether new user can be admitted or not. In other way the network problem is to determine that how many numbers of users can be admitted in the system so that the packet loss probability is less than some threshold.

We consider that the PDF of bit rate of all users is known and PDF of i^{th} source data rate is represented by $f_{Ri}(r_i)$. Because, the bit rate of users is a random variable and independent to each other, so:

$$Y = R_1 + R_2 + \ldots\ldots\ldots\ldots\ldots + R_M$$

$$f_Y(y) = f_{R1}(r_1) * f_{R2}(r_2) * \ldots\ldots\ldots\ldots\ldots * f_{RM}(r_M)$$

$$Probability[Y > L] \Rightarrow packet\ loss\ probability$$

If $packet\ loss\ probability < \delta_{threshold}$, only then new user can enter in the system otherwise not.

III. Proposed VLSI Architecture and Working

Based on the principle described in section II, VLSI hardware architecture is proposed as shown in "fig.4" to control the new user admission in statistical multiplexing and to provide better QoS in terms of packet loss probability.

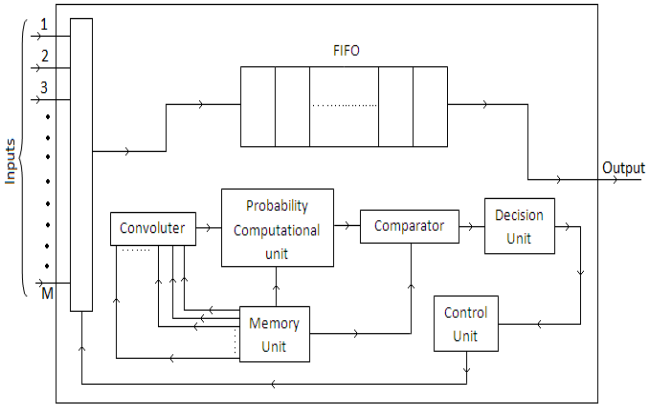

Fig. 4. Proposed Architecture

This reconfigurable hardware architecture has been implemented on FPGA. The FPGA hardware implementation is faster than the software implementation because it takes less time for computation, also it is reconfigurable [7], [8]. The working of this architecture is based on the flow chart as shown in "fig. 5".

If new user wants to enter in the system then it will send its packet to the system. Packet has the source address, by which system will know that it is not the packet from existing user. The convolution unit in the system computes the convolution of all PDF of source data rate, which is already stored in the memory. Its output is fed to the probability computation unit. It computes the probability of combining data rate Y is greater than the maximum data rate of the output, i.e.

$$P(Y > L) = \int_L^{\infty} f_Y(y)\, dy$$

The comparator unit compares the output of probability computation unit with a threshold value which is again stored in the memory. Based on the output of comparator, decision

unit decides whether the packet of new user will be stored in the buffer or it will be discarded. If the output of comparator is 1, it means that the combined bit rate of inputs is less than the maximum bit rate of the output so the packet of new user can be stored in the buffer means new user can enter in the system. Decision unit gives the input to control unit to store the packet in the buffer. If the output of comparator unit is 0, then decision unit gives the input to control unit to discard the packet.

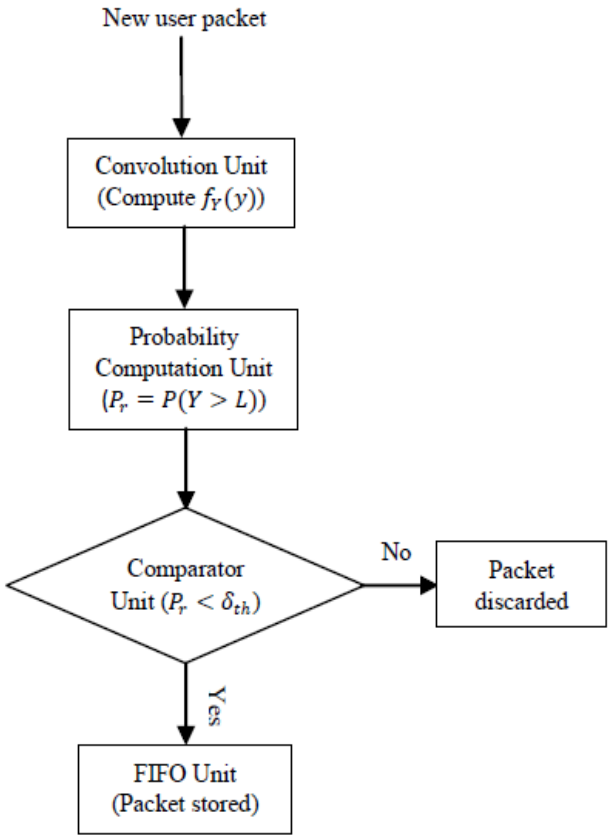

Fig. 5. Flow Chart

IV. Synthesis Results

The proposed architecture is implemented at register transfer level using Verilog HDL and then synthesized using Xilinx ISE 8.2i with Xilinx Virtex technology. The Synthesis of architecture has been performed for Virtex4 XC4VFX140 (speed grade -11, package FF1760). In this paper some assumptions have been taken which are given below:

- Three users are already in the network and fourth user wants to enter in the network.
- The PDF of bit rate of all the users is already known to us and is stored in the memory.

The architecture is divided into four sub-modules and then calls these sub-modules into the final integrated unit. The sub-modules of this project are: convolution unit, FIFO, probability computation unit, and comparator unit. Then interface them with each other to perform the operation of statistical multiplexer. The synthesis results of all the modules are given as follows:

The convolution unit is used to convolve the PDF of bit rate of users. When fourth user comes to the network with its packet which contains the address through which network gets to know that it is not the packet of the existing user, and then it gives the control signal to the convolution unit. Convolution unit take the PDF of bit rate of all the users from the memory and start its function. For the simulation of convolution unit, we have taken two inputs, each have 9 samples of 16 bits. The output of convolution unit has 17 samples of 16 bits.

The hardware implementation of convolution unit takes the 17 clock cycles to perform the convolution of two PDF of order 9. It will always take the 17 clock cycles to calculate the convolution of two inputs of order 9, irrespective of the number of bits to represent the one sample of PDF of data rate. Table I gives the synthesis results of the convolution unit. Here, the samples of PDF are represented in 16-bit binary number. After synthesis minimum time period of convolution unit is 5.34 ns.

TABLE I. Synthesis Results of Convolution Unit

Logic Utilization	Used	Available	Utilization
Number of Slices	201	63168	0.318%
Number of Slice Flip Flops	256	126336	0.203%
Number of 4 input LUTs	260	126336	0.206%
Number of bonded IOBs	50	896	5.580%
Number of GCLKs	1	32	3.125%
Number of DSP48s	9	192	4.688%

The probability computational unit takes the input from the convolution unit and it calculates the probability of combined bit rate of all users is less than the maximum output bit rate. Basically it calculates the sum of samples of convolution output up to the no. of maximum output bit rate.

For the demonstration purpose, we add only 5 samples of the output of convolution unit. The output of probability computational unit gives the addition of 5 samples. Table II gives the synthesis results of the probability computation unit. The minimum clock period of probability computational unit is 3.61 ns.

TABLE II. Synthesis Results of Probability Computation Unit

Logic Utilization	Used	Available	Utilization
Number of Slices	39	63168	0.062%
Number of Slice Flip Flops	40	126336	0.032%
Number of 4 input LUTs	57	126336	0.045%
Number of bonded IOBs	42	896	4.688%
Number of GCLKs	1	32	3.125%

The comparator unit takes one input from the probability computational unit and other input from the memory. Other input indicates the threshold value. If the output of probability computational unit is less than the threshold value, only then

the user can admit in the network. Table III gives the synthesis results of comparator unit.

TABLE III. SYNTHESIS RESULTS OF COMPARATOR UNIT

Logic Utilization	Used	Available	Utilization
Number of Slices	9	63168	0.014%
Number of 4 input LUTs	17	126336	0.013%
Number of bonded IOBs	34	896	3.795%
Number of GCLKs	1	32	3.125%

Based on the output of the comparator, packet is stored in the buffer. In this architecture, the function of buffer is carried out by the FIFO unit. FIFO means the first input first output, so the packets are transmitted in the order in which they are stored in the buffer. Table IV gives the synthesis results of the FIFO unit. The architecture of the FIFO unit has some control input and output like 'read' and 'write' as its input to indicate that which operation will be performed and 'full' and 'empty' as its output to indicate whether the FIFO is full or empty.

For the demonstration purpose, we have assumed that the FIFO can only store the 8 samples of 16 bit. After synthesis, minimum clock period for the FIFO unit is 3.33 ns.

TABLE IV. SYNTHESIS RESULTS OF FIFO

Logic Utilization	Used	Available	Utilization
Number of Slices	129	63168	0.204%
Number of Slice Flip Flops	162	126336	0.128%
Number of 4 input LUTs	241	126336	0.191%
Number of bonded IOBs	42	896	4.688%
Number of GCLKs	1	32	3.125%

TABLE V. SYNTHESIS RESULTS OF INTEGRATED MODULE

Logic Utilization	Used	Available	Utilization
Number of Slices	1394	63168	2.207%
Number of Slice Flip Flops	1763	126336	1.395%
Number of 4 input LUTs	1355	126336	1.073%
Number of bonded IOBs	219	896	24.442%
Number of GCLKs	1	32	3.125%
Number of DSP48s	81	192	42.188%

After the simulation and synthesis of individual block, authors have interfaced all the modules with each other to perform the operation of statistical multiplexer. In this paper, it is assumed that three users are already in the network but fourth user wants to admit in the network. In the integrated main module, convolution unit has been called three times to perform the convolution of PDF of data rate for four users. All the modules work in parallel, means the probability computation unit does not wait for the whole output of convolution unit. It starts its working when it gets the first sample of the output from convolution unit. Table V gives the synthesis result of the statistical multiplexer. After synthesis minimum clock period for the integrated unit is 9.297ns.

V. CONCLUSION

This paper has presented a VLSI Reconfigurable hardware architecture for statistical multiplexer on FPGA platform. FPGA implementation will be faster than the software implementation. We have synthesized the integrated module and the different sub-modules of statistical multiplexer. All the modules are performing their operation in very less time only in the range of 'ns'. In this paper, modules are implemented using the academic FPGA, if commercial FPGA is used then the speed will further increase.

REFERENCES

[1] Jin Cao, William S. Cleveland, Dong Lin, and Don X. Sun, " The Effect of Statistical Multiplexing on Internet Packet Traffic: Theory and Empirical Study," Research Bell Labs, Murray, Hill, NJ.

[2] Amr Rizk and Markus Fidler, "On Multiplexing Models for Independent Traffic Flows in Single and Multi-Node Networks", IEEE Transactions on Network and Service Management, Vol. 10, No. 1, March 2013.

[3] Robert Boorstyn, Almut Burchard, Jorg Liebeherr, Chaiwat Oottamakorn, "Statistical Multiplexing gain of Link Scheduling Algorithms in QOS networks," Technical Report: University Of Virginia, CS-99-21, July 1999.

[4] C. D'Apice, R. Manzo, S. Salerno and N. Likhanov, "Network Traffic Modelling and Packet-Loss Probability Approximation", Journal of Mathematical science, Vol. 132, No. 5, 2006.

[5] Data and Computer Communications by William Stallings.

[6] Jianwei Huang, Chee Wei Tan, Mung Chiang, Raphael Cendrillon, "Statistical Multiplexing over DSL Networks".

[7] Wajid, Mohd, and S. B. Shashank. "Architecture for Faster RAM Controller Design with Inbuilt Memory." Computational Intelligence, Communication Systems and Networks (CICSyN), 2010 Second International Conference on. IEEE, 2010.

[8] Mirza, Khushboo, et al. "Reconfigurable Vlsi Architecture for Graph Color Allocation with Its Chromatic Number." Computer Networks and Information Technologies. Springer Berlin Heidelberg, 2011. 531-534.