CHAPTER 4

SIMULATION MECHANISMS AND SPACE DIVISION SWITCHES

4

SIMULATION MECHANISMS AND SPACE DIVISION SWITCHES

4.1 INTRODUCTION

This chapter introduces detail of the simulation mechanism and discusses data generation using different traffic models. It discusses the MATLAB simulation of two types of space division switches:
1. Knockout switch and 2. Multistage switches like Banyan/Batcher-Banyan/Batcher-Banyan-Trap. At last it discusses VHDL implementation of 8x8 Knockout switch for different concentrator output and 8X8 Banyan/Batcher-Banyan/Batcher-Banyan-Trap in VHDL, using Maxplus II/Quartus II tool of Altera.

4.2 SIMULATION MECHANISM

The performance of switches is measured by simulation. To compare different switches, we generate four different data patterns (A, B, C, D). Same data are applied to different switches including crossbar with different scheduling algorithms and results are plotted. We get two data sets from the user: 'n' to decide number of time slots and 'bern' for Bernoulli probability for packets non arrival. After that we generate the packets depends on *unirandom* function and compare with the Bernoulli probability for packets not arrival. If the generated probability is higher than the probability given by the user then packet from that input is generated and number is inserted in odd column of "in" otherwise "0" is inserted in odd column of "in" to indicate no data is generated in that time slot . Destination number is generated depends on output is uniformly distributed or normally distributed.

A. With i.i.d. Bernoulli arrivals and uniformly distributed destinations:

In this traffic model, cell arrives as i.i.d. (independent and identically distributed) Bernoulli arrivals and cell destinations distributed uniformly over all outputs and generated by using unirandom function like;

D(1..N) = unidrnd (N, 1, n);

Above function generates 1 x n array, with data element (Destination) uniformly distributed from 1 to N and destination number generated is inserted in output location (even column of "in"). If input data (odd column of "in") is not generated for that slot, then in output location (even column of "in") "0" is inserted.

B. With i.i.d. Bernoulli arrivals and non-uniformly distributed destinations

Real traffic is not usually uniformly distributed over all outputs, but is asymmetric. In this traffic model cell arrives as i.i.d. (independent and identically distributed) Bernoulli arrivals and cell destinations distributed normally over some outputs with variance .An example of this traffic model would be a client-server work load in which a large number of clients communicate with a small number of servers. Cell destinations distributed normally over some outputs, generated by following code (1--N):

d(1--N) = normrnd(dia(1--N),varience,1,n);

It generates 1 x n array, with data element (Destination) normally distributed from 1 to N depends on mean {dia(1--N)} and standard deviation {variance} given by the user. Destination number generated is inserted in output location (even column of "in"). If input data (odd column of "in") is not generated for that time slot then in output location (even column of "in") "0" is inserted.

d(1--N) = abs(d(1--N)); % to make the number in d1 to absolute value.

d(1--N) = int8(d(1--N));% To convert destination number into integer.

C. With bursty arrivals and uniformly distributed destinations:

Real traffic is not only asymmetric it is highly correlated from cell to cell and so it is also bursty. We illustrate the effect of burstiness on PIM, RRM, iSlip, RPA, DPA, m-DPA,DSA scheduling algorithms using an on-off arrival process modulated by a two state Markov chain. The source alternatively produces a burst of full cells (on period) followed by an idle period of empty cells (off period). We get three data sets from the user: 'onp' to decide on period, 'ofp' to decide off period, and 'times' to decide number of time this cycle repeats. Destination is generated as in traffic pattern A.

D. With bursty arrivals and non-uniformly distributed destinations:

In this traffic model input generated using an on-off arrival process modulated by a two state Markov chain. The source alternatively produces a burst of full cells followed by an idle period of empty cells. The bursts and idle periods contain a geometrically distributed number of cells [65]. Destination is generated as in traffic pattern B.

Details of Bernoulli arrival process and Markov bursty traffic model are appended as Appendix A.

Data generated in traffic pattern A, for 4x4 switch, up to 10 timeslots are shown below: Two columns (one for input and second for output) are there for each time slot.

```
in =  Columns 1 through 20
  1  4  0  0  1  3  1  4  1  1  1  4  1  2  1  2  1  4  1  3
  2  1  2  1  2  4  2  1  2  2  2  3  2  2  2  2  2  1  2  4
  3  3  3  2  3  3  3  2  3  2  3  1  3  3  3  4  3  3  3  3
  4  1  4  2  4  4  4  3  4  2  4  3  4  4  4  1  4  3  4  1
```

IN is variable name given to the data generation. There are 20 column in "IN". In first two column and in first row number "1" and "4"(with bold case), indicates that input port 1 to output port 4 connection is requested for data transfer. Same way second row interprets that input 2 to output port 1 connection is requested. Same way combination of odd column and even column give the request of port connection for the data transfer. In first row, 3rd and 4th column (with bold case),"0" and "0" interpreted as there is no data request. Table 4.1 shows the general simulation setting.

Table 4.1 General simulation setting

Simulation setting	
Simulation tool	MATLAB 7.0
Computer	Pentium IV CPU 3.20GHz ,0.99GB memory
Operating system	Windows XP
Traffic pattern	A,B,C,D
Offered load	Range from 0 to 100% in steps of 10%
Buffer size	Range from 1 to 8
Switches being simulated	Knockout switch, Batcher Banyan Trap switch, Crossbar switch along with PIM ,RRM, iSLIP, RPA, DPA, m-DPA, and DSA scheduling algorithms
Simulation duration for each simulation case	10,000 cell times for the 4x4 and 8x8 switches 1000 cell times for 16x16 and 32x32 switches
Averaging	100 times for 4x4 and 8x8 switches 10 times for 16x16 switches 4 times for 32x32 switches

$$\text{Throughput in \%} = \frac{\text{Number of successfully transmitted packets} + \text{Number of packets in buffer}}{\text{Total number of packets generated}}$$

The packet latency is defined as the interval time slots from the arrival time of a packet to the time slot in which it is successfully transmitted to output port.

Average latency is defined as: Average latency = mean (packet latency)

Delay variance is defined as: Delay variance = var(packet latency)

We keep same data environment. Parameters like throughput, packet latency, average latency, and delay variance are obtained by running MATLAB program of each switch/scheduling algorithms and graphs are plotted.

4.3 KNOCKOUT SWITCH

The knockout switch architecture was proposed by a group of AT&T Bell Laboratories. This fully connected architecture, combines the implementation simplicity of input queuing with the throughput performance of output queuing. In normal multistage space switches, output queuing cannot be used unless switch speedup is utilized. In this architecture every input has a non-overlapping direct path to every output so that blocking does not occur internally. It also employs output queuing to resolve the output port contention. In knockout switch it is not really necessary to buffer up to N cells at each output. Because it is generally highly unlikely that all N inputs will have a cell destined for a given output. In each cell interval, the number of cells accepted at each output port can be limited to a relatively small fixed number L, independent of N.($L \leq N$). This intended packet loss is known as *buffer blocking*. The rate of loss from buffer blocking can be readily controlled and kept low, to reduce the complexity of a switch.

4.3.1 Knockout Switch Architecture

The knockout switch uses one broadcast input bus from every input port to all output ports. Each cell transmitted onto its broadcast bus in each cell length time slot, carries the desired output port address. An output module recognizes its output port address and taps the cells intended for it. Multicast and broadcast cells are supported by this architecture.

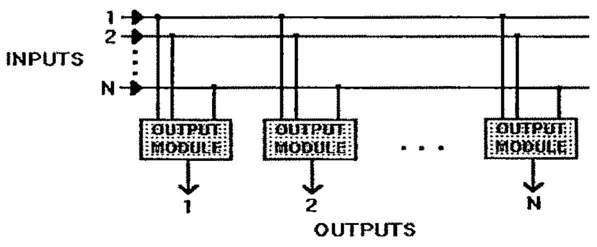

Figure 4.1 NxN Knockout switch architecture [45]

4.3.1.1 Output module

Figure 4.2 shows the internal architecture of output module.

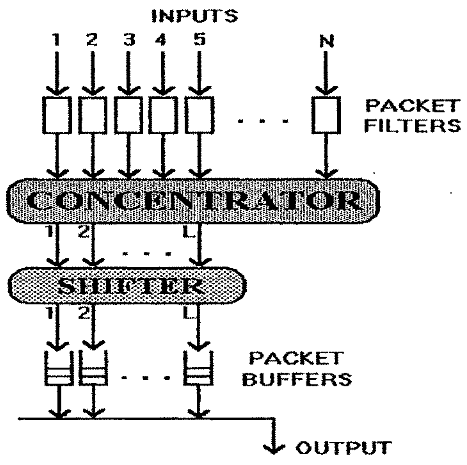

Figure 4.2 Detail Architecture of output module [104]

Each input to an output module receives the fixed-length cells broadcasted on the corresponding input bus.

4.3.1.1.1 Packet Filter

The job of each packet filter is simply to pass the cell to the concentrator if the cell is destined for that output. There are set of N packet filters, each interfacing a line in each output module.

4.3.1.1.2 Concentrator

The role of the concentrator is to identify among its inputs those cells that are active and route them to its leftmost outputs, one cell per output line. Here, at most, $L \leq N$ cells are retained and queued at each output port during each cell time slot. When L+X cells arrive simultaneously, only L of them will be processed via the concentrator; all others will be lost. By properly choosing L, (independent of the number of inputs) the loss rate induced by the concentrator can be controlled and maintained at a reasonably low level. Analysis shows that, L = 8-10 is sufficient to maintain the packet loss rate in the concentrator at one packet per million, for large N and full input load, and it only grows by one per every order of magnitude reduced in the loss rate (i.e. L = 11 is enough for a loss rate of one packet per billion) [55]. This effect is the key to maintain linear complexity of the knockout switch, as the numbers of buffers are proportional to $L \times N$ rather than N^2.

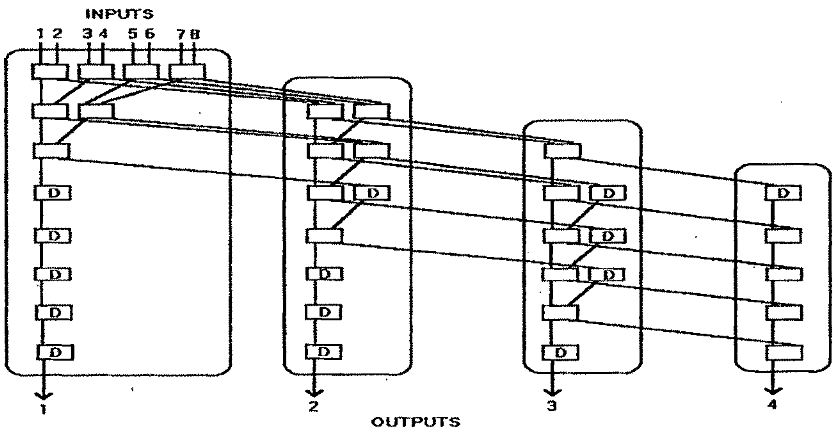

Figure 4.3 Detail architecture of 8x4 Concentrator [33]

As shown in figure 4.3, concentrator carries out pair wise knockout competition, for each time slot, giving uniform treatment to cells from different outputs. In 8 X 4 concentrator there are four (generally, *L*) stages as shown in the figure 4.3. Each stage is designed to operate like an elimination tournament. Contention is programmed to set itself to the *"bar"* state if there is an active cell on its left input, and to *"cross"* otherwise as shown in figure 4.4(b) and 4.4(c) respectively. Whenever there is only one active cell at the inputs of a contention, it is allowed to pass downward. If both cells are active, the right-hand one is "knocked out" to the next stage and contends there. Each stage produces one "winner" among the active cells that enter it, and each subsequent stage receives one less active cell than the previous one. Therefore, when there are *k* active cells, they are guaranteed to come out on the outputs of the first *k* stages.

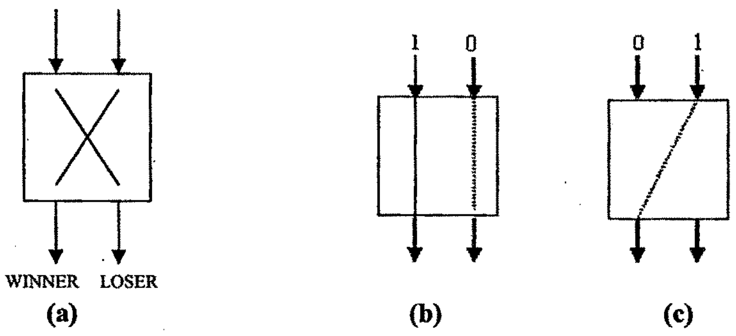

Figure 4.4 (a) The 2x2 contention switch (b) Bar States (c) Cross State [84]

4.3.1.1.3 Shifter

If the shifter is not used between concentrator and the packet buffer then the leftmost buffers would tend to fill up faster, and might even overflow despite the presence of empty buffer entries on the right. The shifter prevents that from happening by spreading each bulk of cells arriving at its input continuously to the right; i.e. if the last buffer to receive a packet happened to be n, then the next p cells arriving at the shifter's input will be directed to buffers n+1, ..., n+p (modulo *L*).

The L FIFO buffers are completely shared and thus are equivalent to a single FIFO queue with L inputs and one output. Because of the round-robin nature of the shifter, buffers are filled and emptied cyclically. At each time slot, the output line fetches a cell from a buffer just right (cyclically) of the buffer last fetched from, beginning with buffer 1. Due to round-robin buffer filling policy, if the output circuitry encounters an empty buffer, all buffers are empty at that point. The output pointer can then just stops there and wait for that buffer to receive a cell, after which the circular emptying of buffers can restart from that point, onwards.

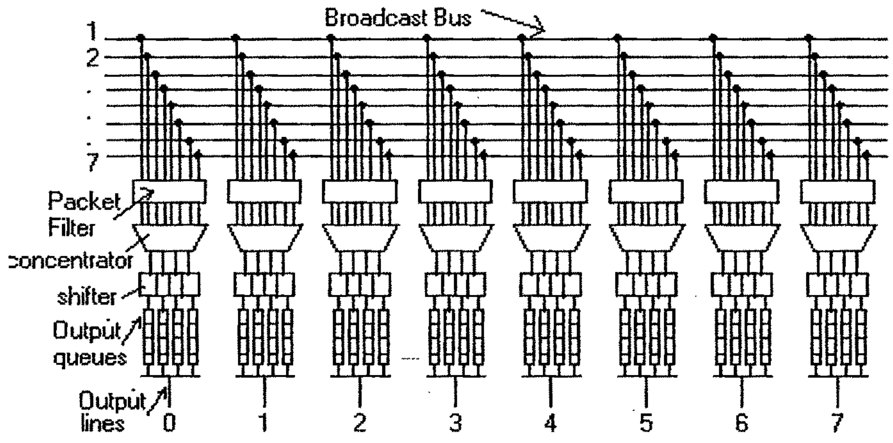

Figure 4.5 Detail architecture of 8x8 knockout switch [89]

4.3.2 Determination of packet loss probability of knockout switch

If probability of packet loss from output congestion is kept below the loss expected from the network sources like transmission line errors, network failure, and buffer overflows, then packets dropped within the N to L concentrator should not be so important. In NxN switch fabric, each N input has a Bernoulli probability α that a packet independently arrives in a time slot at each input and input traffic is uniformly distributed among the N outputs. So each packet is equally destined for each output with probability 1/N. Then the probability of k packets arriving in a time slot all destined for a given output is given by binomial distribution as per following equation:

$$P_k = (NC_k)(\alpha / N)^k (1 - \alpha / N)^{N-k}$$

$$k=0, 1, 2, \ldots\ldots\ldots\ldots, N$$

A packet is dropped in the N→L concentrator if the number of packets arriving in a time slot destined for the same output exceeds the number for concentrator outputs L.

Since the average number of packets lost equals the probability of packet arrival times the packet loss probability P_L, we obtain the probability of a packet loss

$$P_L = (1/\alpha) \times Ave.\ number\ of\ packets\ lost$$

$$= (1/\alpha) \times \sum_{k=L+1}^{N} (k - L) p_k \text{ .... } 4.1$$

$$= (1/\alpha) \times \sum_{k=L+1}^{N} (k-L) p_k (NC_k)(\alpha/N)^k (1-\alpha/N)^{N-k}$$

Taking the limit as N ⟶ ∞, For N very large, the binomial probability approaches the Poisson probability, we get Prob[packet loss] P_L

$$P_L = (1 - L/\alpha)(1 - \sum_{k=0}^{L} (\alpha^k / k!) e^{-\alpha}) + \alpha^L (e^{-\alpha} / L!) \text{ .... } 4.2$$

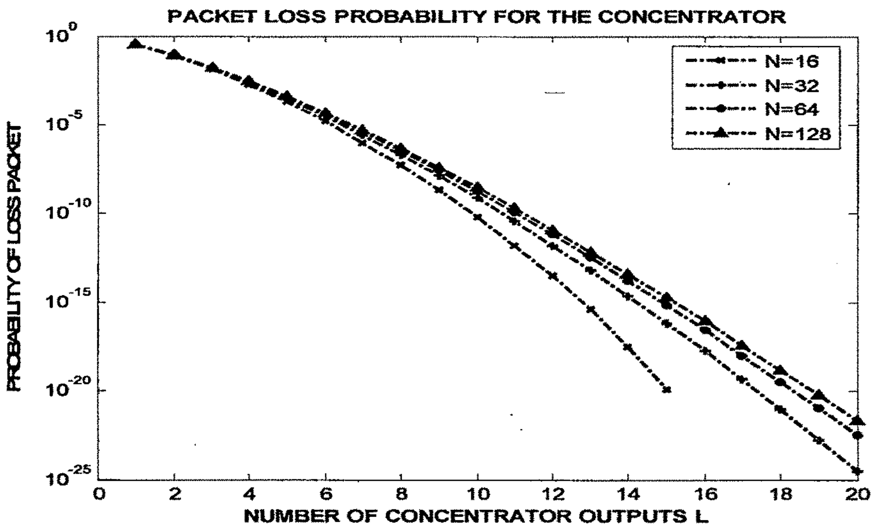

Figure 4.6 Packet loss probability for the concentrator

Using equation 4.1 and 4.2, the probability of packet loss versus the number of outputs on the concentrator L for N=16, 32, 64 and 128 at an offered load of 90% as shown in figure 4.6. For the probability of packet loss to be less than 10^{-6} for large N, only 8 concentrator output are required. Thus, the number of separate buffers needed to receive simultaneously arriving packet with relatively low probability of packet loss is reduced from large N to L=8. It was assumed that any packet is equally destined to any of the output lines. However, if the traffic pattern is non-uniform, L has to be high (up to 20) to sustain the same packet loss rate.

4.4 BANYAN/BATCHER-BANYAN/BATCHER-BANYAN-TRAP SWITCH

Banyan switch is a multistage space division type switch, in which a number of cross points are not a quadratic in the number of lines unlike knockout switch and crossbar switches.

4.4.1 Banyan Switch

As shown in figure 4.7(a), for 8 x 8 three-stage banyan switch only one path exists from each input line to each output line. Routing is done by looking up the output line for each cell, based on virtual circuit information and tables. This 3 bit binary number is then put in front of the cell, as it will be used for routing through the switch.

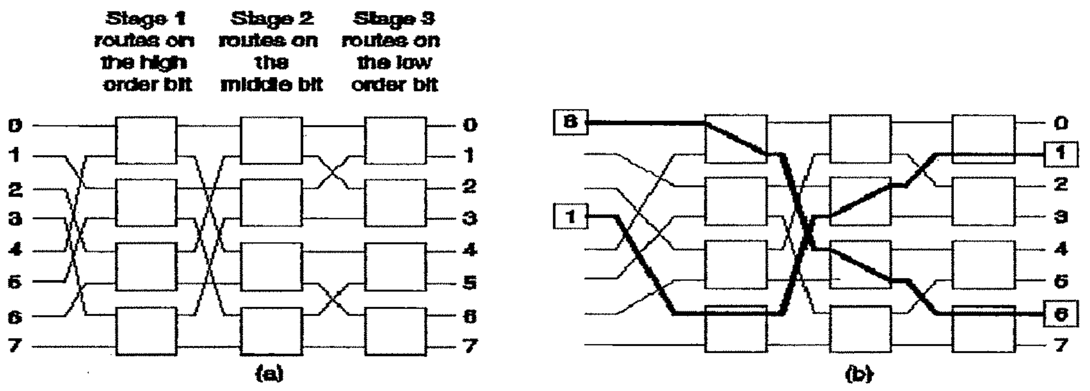
Stage 1 routes on the high order bit
Stage 2 routes on the middle bit
Stage 3 routes on the low order bit
0
1
2
3
4
5
6
7
(a)
6
1
(b)

Figure 4.7 (a) A banyan switch with eight input lines and eight output lines.
(b) The routes that two cells take through the banyan switch.

Each of the 12 switching elements in the banyan switch has two inputs and 2 outputs. When a cell arrives at a switching element, 1 bit of the output line number is inspected, and based on that, the cell is routed either to port 0 (the upper one) or port 1 (the lower one). In case of collision, one cell is routed and one is discarded? A banyan switch parses the output line number from left to right as shown in figure 4.7(b). Throughput of banyan switch depends on input data position, which is clear from the figure 4.8(a) and 4.8(b). In case of figure 4.8(a), throughput is only two packets, while in figure 4.8(b), with different input position throughput is 100%. So we need a sorting network ahead of a banyan switch.

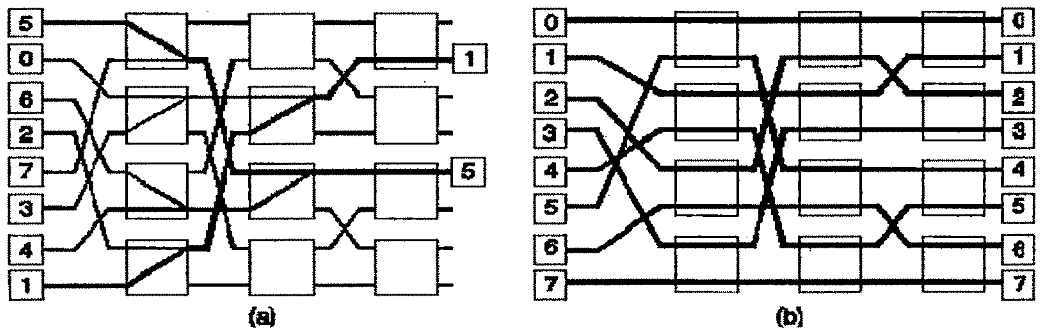

Figure 4.8 (a) Cells colliding in a banyan switch. (b) Collision-free routing through a banyan switch [89]

4.4.2 Batcher Banyan Switch

The idea behind the Batcher-Banyan switch is to put a batcher switch in front of the banyan switch to sort the cells into an order that the banyan switch can handle without loss. To sort the incoming cells we can use a Batcher switch built up of 2 x 2 switching elements. When such a switching element receives two cells, it compares their output addresses numerically (not just 1 bit) and routes the higher one in the direction of arrow, and the lower one on the other way. If there is only one cell, it goes opposite the way the arrow is pointing, as shown in Figure 4.9.

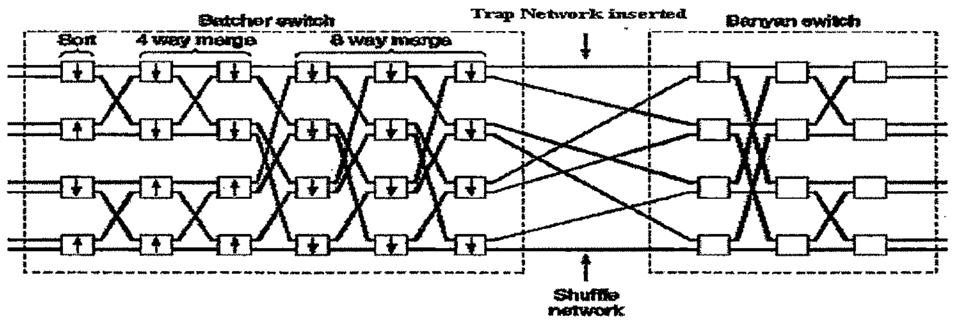

Figure 4.9 The switching fabric for a Batcher-Banyan switch.

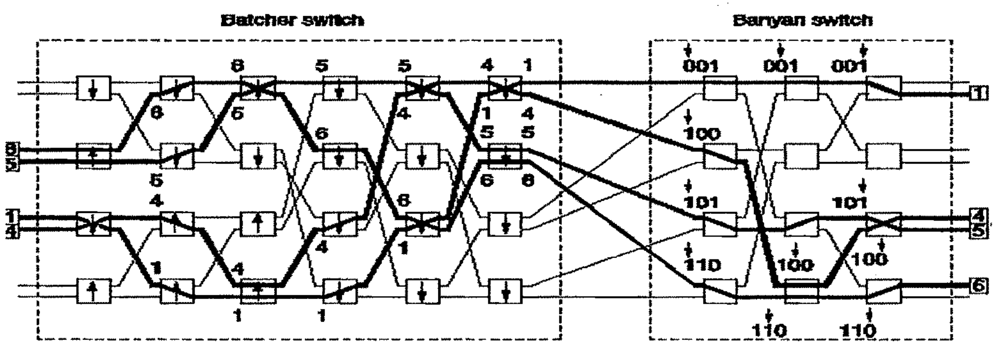

Figure 4.10 An example with four cells using the Batcher-Banyan switch.

The Batcher-banyan switch has two complications: output line collision and multicasting. If two or more cells are aimed at the same output, the switch cannot handle them, so a kind of buffering has to be introduced between Batcher and Banyan switch.

4.4.3 Batcher Banyan Switch with Trap

Output collision problem in Batcher-Banyan switch can be solved by inserting a trap network between the Batcher switch and banyan switch as shown in figure 4.11, which filters out duplicates and re circulates them for subsequent cycles, all the while maintaining the order of cells on a virtual circuit.

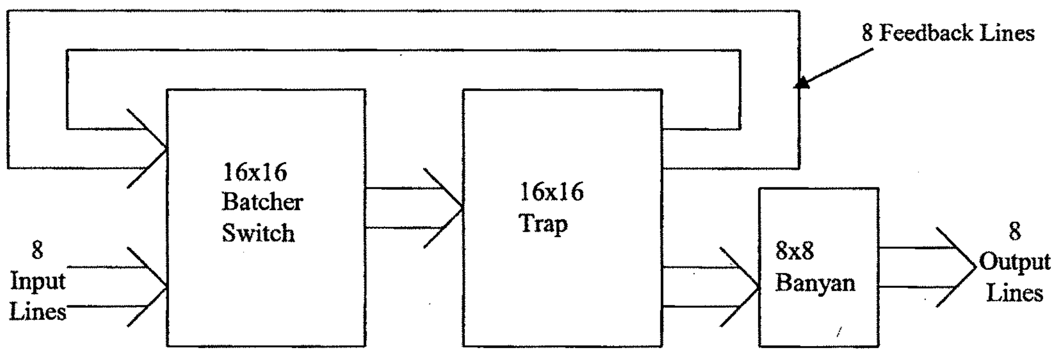

Figure 4.11 8x8 Batcher Banyan switch with Trap

4.5 SIMULATIONS OF KNOCKOUT AND BATCHER-BANYAN SWITCHES

To visualize the performance of the above two switches, we have simulated these switches for different traffic models using MATLAB 7.0 Tool.

4.5.1 Simulations of Knockout Switch

4.5.1.1 Simulation of 4x4 Knockout switch with buffer size 2

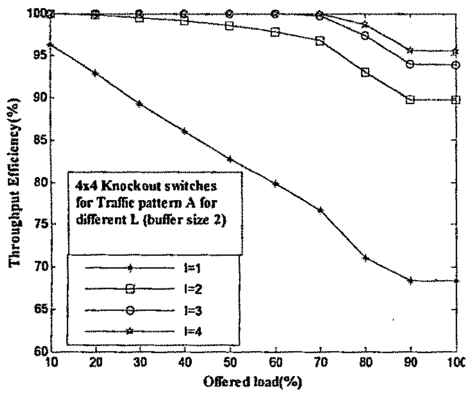

Figure 4.12 Traffic Pattern A

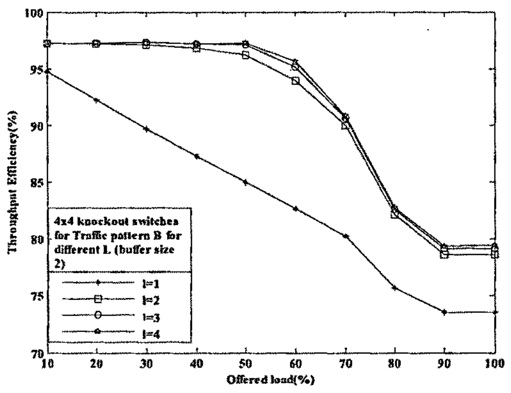

Figure 4.13 Traffic Pattern B

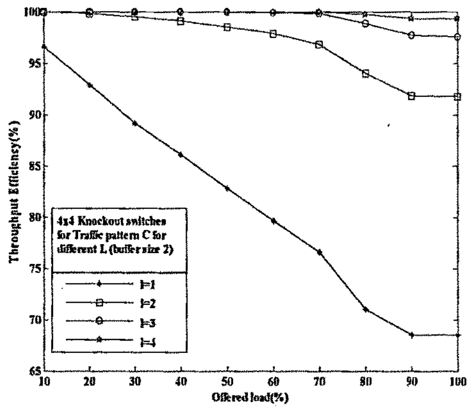

Figure 4.14 Traffic Pattern C

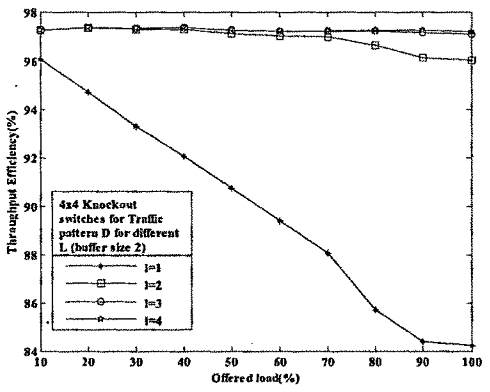

Figure 4.15 Traffic Pattern D

4.5.1.2 Simulation of 4x4 Knockout switch with buffer size 4

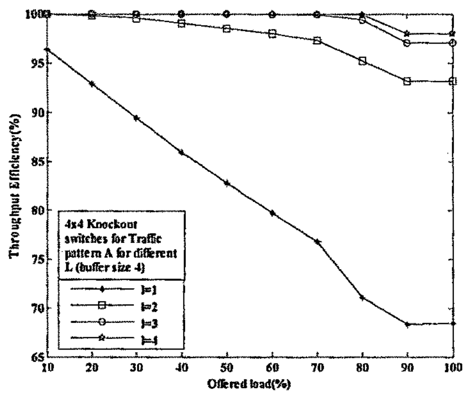

Figure 4.16 Traffic Pattern A

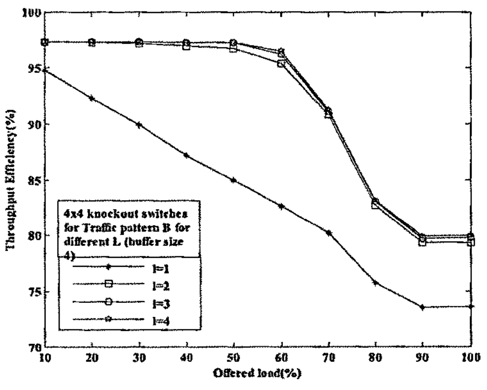

Figure 4.17 Traffic Pattern B

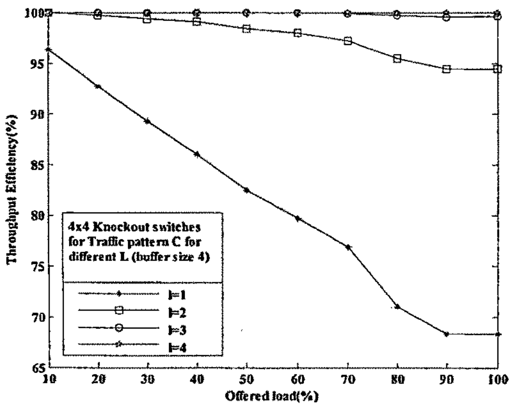

Figure 4.18 Traffic Pattern C

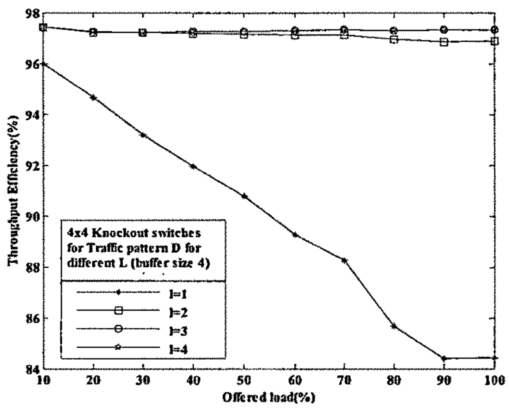

Figure 4.19 Traffic Pattern D

4.5.1.3 Simulation of 8x8 Knockout switch with buffer size 2 packets

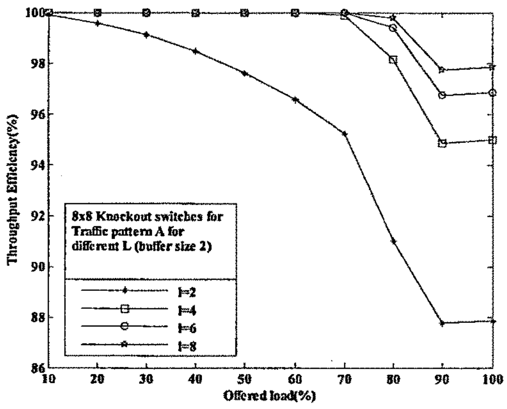

Figure 4.20 Traffic Pattern A

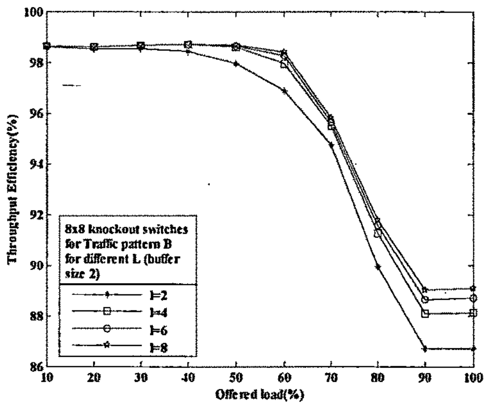

Figure 4.21 Traffic Pattern B

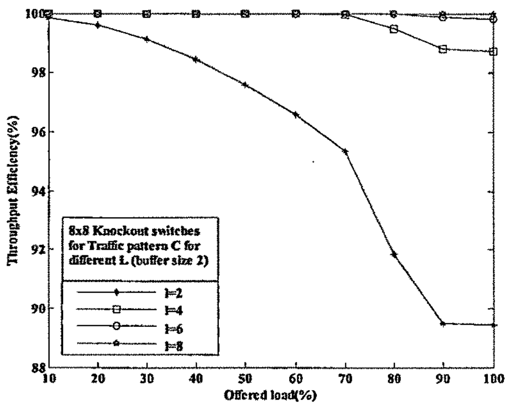

Figure 4.22 Traffic Pattern C

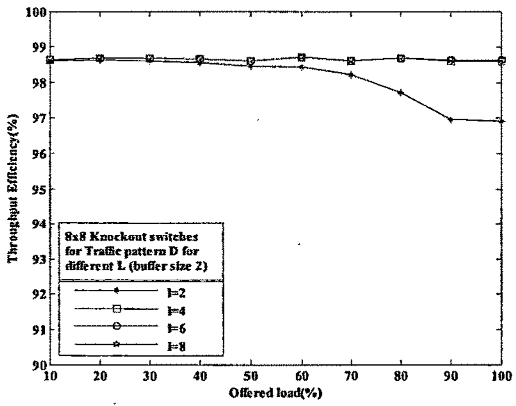

Figure 4.23 Traffic Pattern D

4.5.1.4 Simulation of 8x8 Knockout switch with buffer size 4 packets

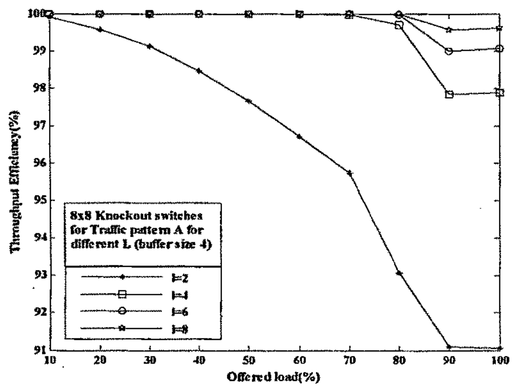

Figure 4.24 Traffic Pattern A

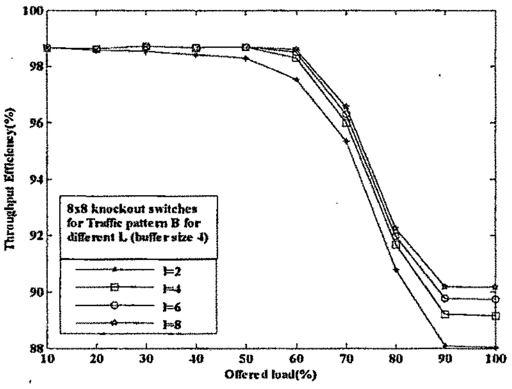

Figure 4.25 Traffic Pattern B

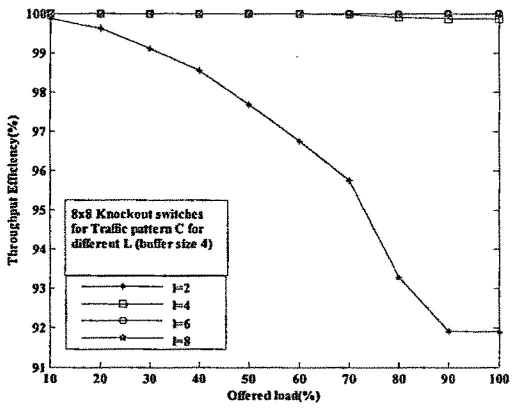

Figure 4.26 Traffic Pattern C

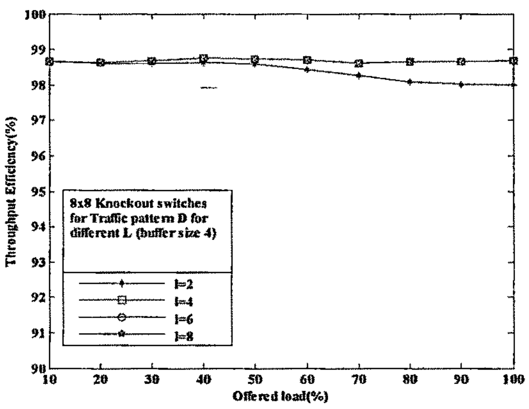

Figure 4.27 Traffic Pattern D

4.5.1.5 Analysis of Knockout switch

As shown in figures 4.12 to 4.27, in knockout switch throughput (efficiency) increases as L (concentrator output line) increases. The throughput (efficiency) increase in large steps at low values of L and remains nearly same at higher values of L. Efficiency for normal distributed output is lower than the uniformly distributed output. In 4x4 switch, from L=1 to L=2 throughput (efficiency) increases 22% in traffic pattern A and C, 5% in traffic method B, and 12% in traffic pattern D. In 8x8 switch, from L=1 to L=2 throughput (efficiency) increases 8% in traffic pattern A and C, 1.5% in traffic pattern B and D.

Figures 4.16 to 4.19 and 4.24 to 4.27 show the same thing for buffer size of 4 packets. There is no significant difference in the throughput (efficiency), when buffer size increases from 2 to 4.

4.5.2 Simulations of Batcher Banyan Switch

4.5.2.1 Simulation of 4X4 Banyan/Batcher-Banyan/Batcher-Banyan-Trap

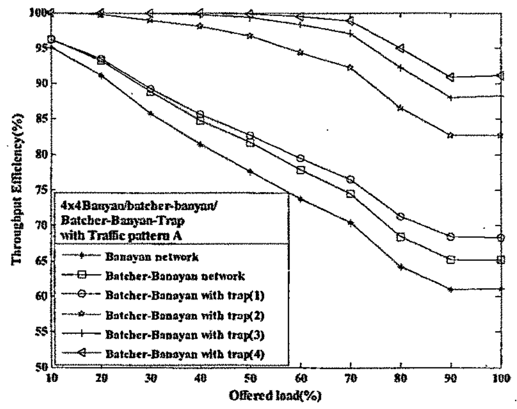

Figure 4.28 Traffic Pattern A

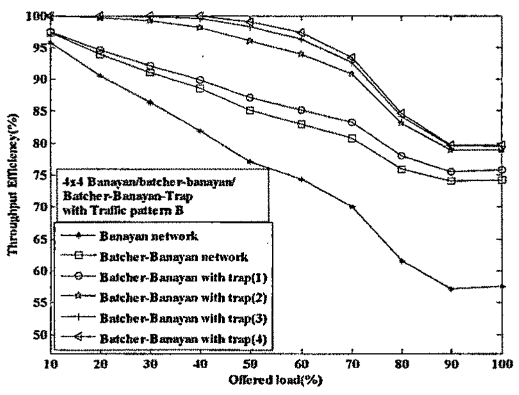

Figure 4.29 Traffic Pattern B

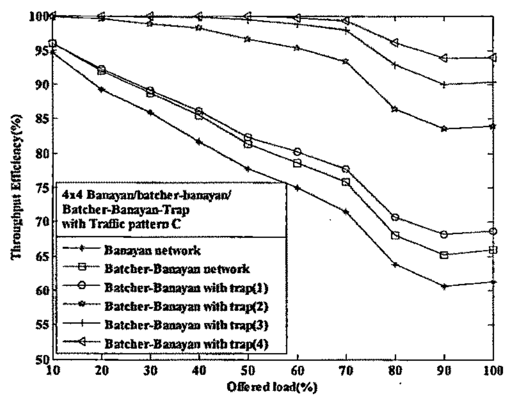

Figure 4.30 Traffic Pattern C

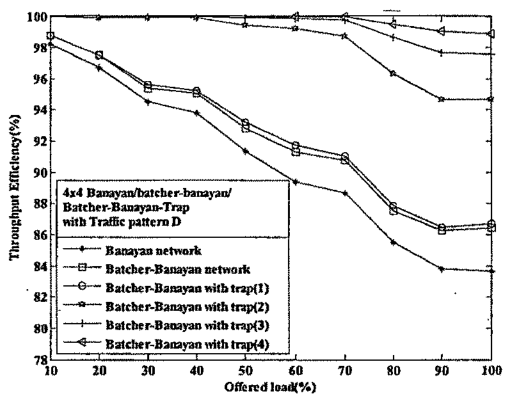

Figure 4.31 Traffic Pattern D

4.5.2.2 Simulation of 8X8 Banyan/Batcher-Banyan/Batcher-Banyan-Trap

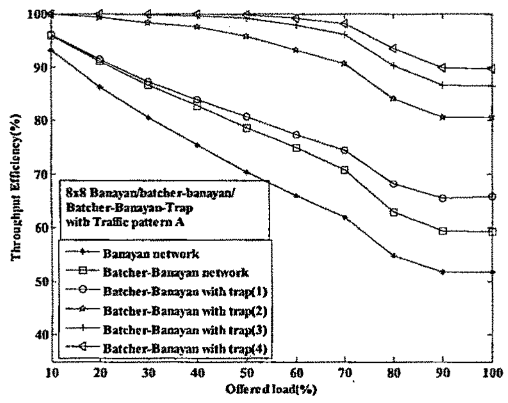

Figure 4.32 Traffic Pattern A

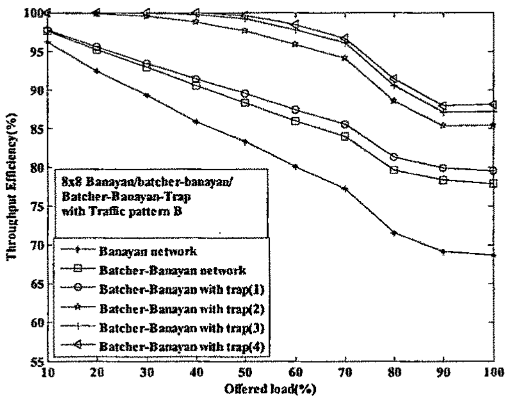

Figure 4.33 Traffic Pattern B

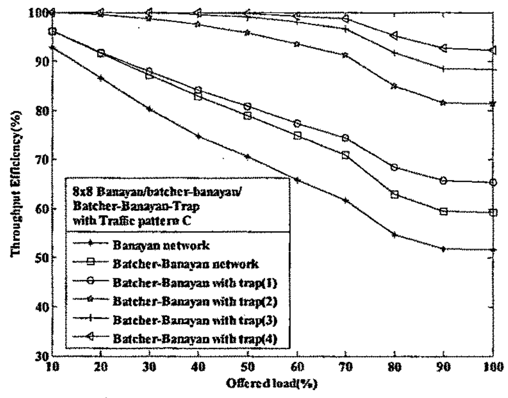

Figure 4.34 Traffic Pattern C

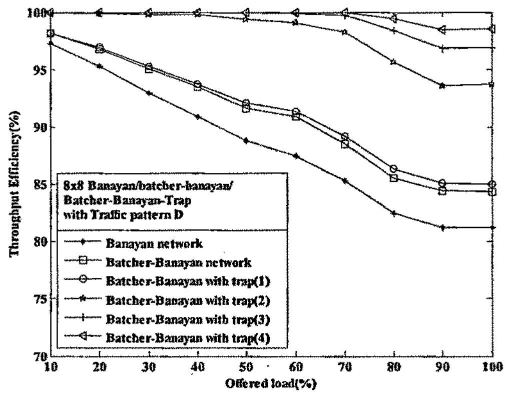

Figure 4.35 Traffic Pattern D

4.5.2.3 Analysis of Batcher Banyan switch

As shown in figures 4.28 to 4.35, throughput (efficiency) of Banyan switch is low. Efficiency of Batcher-Banyan switch decreases as offered load increases for an NxN Batcher Banyan switch. Efficiency of Batcher-Banyan switch is average. Efficiency of Batcher Banyan switch is increased by using the Trap network between the Batcher sorter and Banyan switch. Trap network is more efficient (i.e. 25% increase in throughput (efficiency) from trap 1 to trap 4) in traffic pattern A and C, while Batcher shorter is more efficient in traffic pattern B (i.e. 10% to 16% increase in throughput in traffic pattern B)

4.6 IMPLEMENTATIONS OF KNOCKOUT AND BATCHER-BANYAN SWITCHES

We have implemented above switches using Altera's QUARTUS II 4.0 tool. Figure 4.36 shows the format of the packet. The beginning of packet contains activity bit, which indicates the presence (1) or absence (0) of a packet in the arriving timeslot. Activity bit is followed by $\log_2 N$ bits destination address. At last the payload is attached. We make payload 8-bit for implementation simplicity. We use same 12-bit packet format for Batcher-Banyan-Trap and Knockout switches for comparison.

1-Active bit	3-Destination Address bit	8-Data bit

Figure 4.36 Implementation Packet format.

4.6.1 Implementations of 8x8 Knockout Switch

The architecture of Knockout switch is modular. Various building blocks or modules of the Knockout switch and its waveform are as follows:

4.6.1.1 Packet Filter

Filter checks active bit and destination address. If active bit is '1'and destination address is same for that particular output, then filter passes those packets and for others it takes all bits as '0'. Waveform for destination address "000" is shown in figure 4.37.

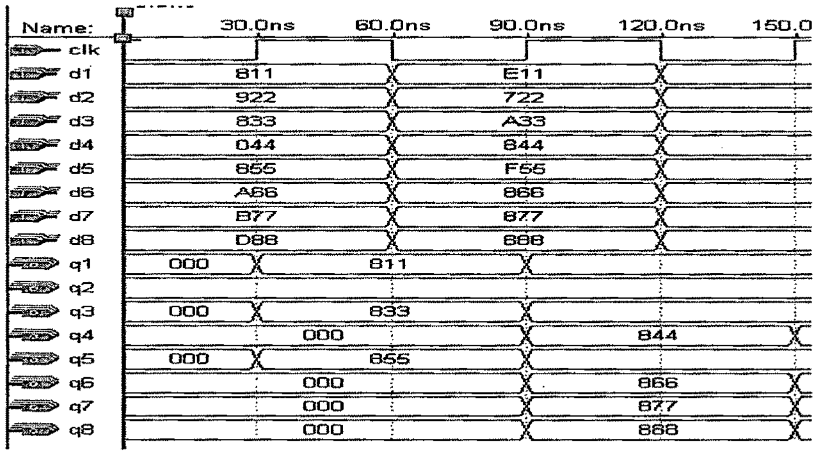

Figure 4.37 Waveform of Filter for destination address "000".

4.6.1.2 Contention

Contention checks active bit only. If both packets have active bit as '1', then input-1 packet is the winner and input-2 packet loses, and if only one of the packet has the active bit as '1', then that particular packet is the winner.

Waveform for contention is shown in figure 4.38.

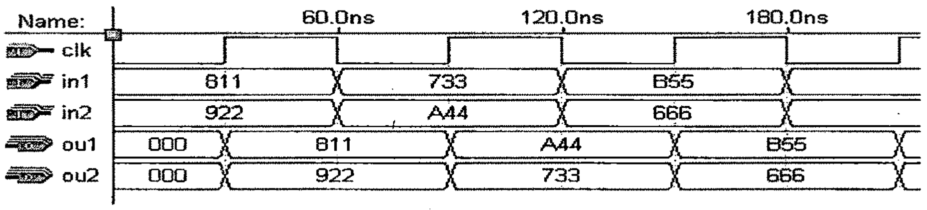

Figure 4.38 Waveform of contention.

4.6.1.3 Shifter

Shifter checks active bit only. Waveform of 4x4 shifter is shown in the figure 4.39. During first time slot only 2 packets are active, so shifter fills output 1 and 2 and for output 3 and 4 all bits are '0'. During second time slot 3 packets are active, so now it fills the output starting from output 3, 4 and 1.

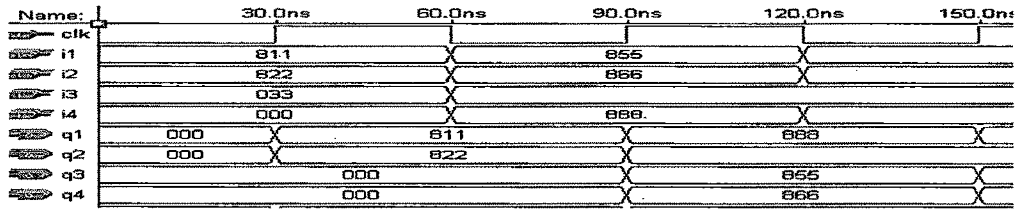

Figure 4.39 Waveform of 4x4 shifter.

4.6.1.4 Buffer read clock generator

Buffer read clock generator generates buffer read clock cyclically.

Waveform for buffer read clock generator is shown in figure 4.40

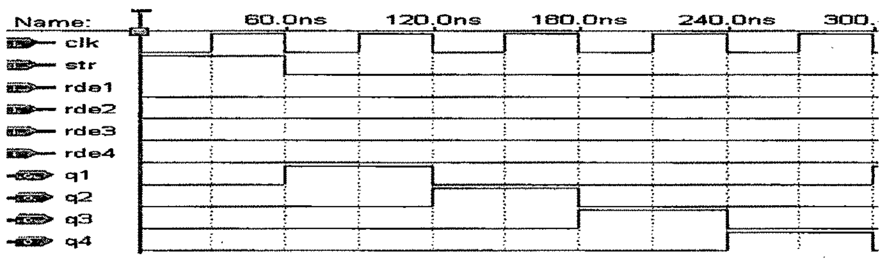

Figure 4.40 Waveform of Buffer read clock generator.

4.6.1.5 Buffer write clock generator

Buffer write clock generator checks active bit. If the active bit of packet is '1' then it generates buffer write clock. Waveform for buffer write clock generator is shown in figure 4.41

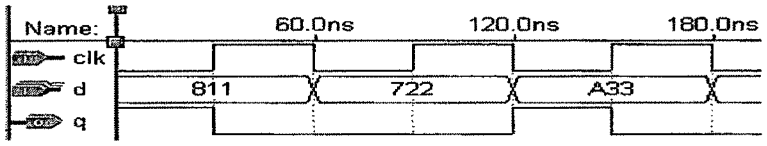

Figure 4.41 Waveform of Buffer write clock generator.

4.6.1.6 Output selector

Output selector reads buffer at rising edge of buffer read clock as shown in figure 4.42.

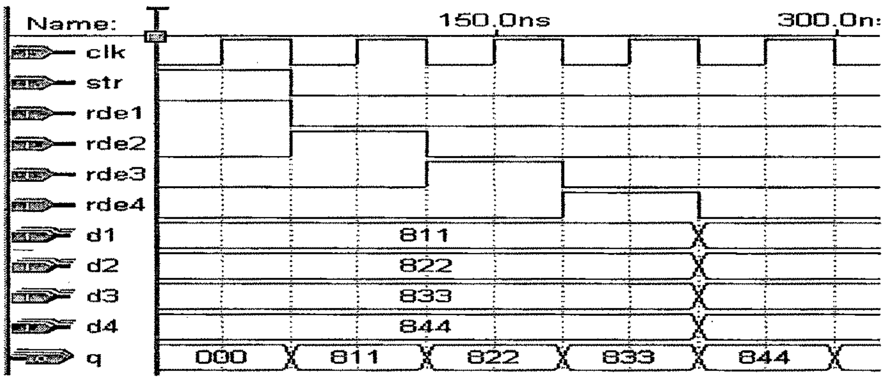

Figure 4.42 Waveform of output selector.

4.6.1.7 Concentrator

Concentrator checks active bit and takes L number of packets as output. Waveform for 8x4 concentrator is as shown in figure 4.43. During first time slot, more than 4 packets have active bit as '1'.

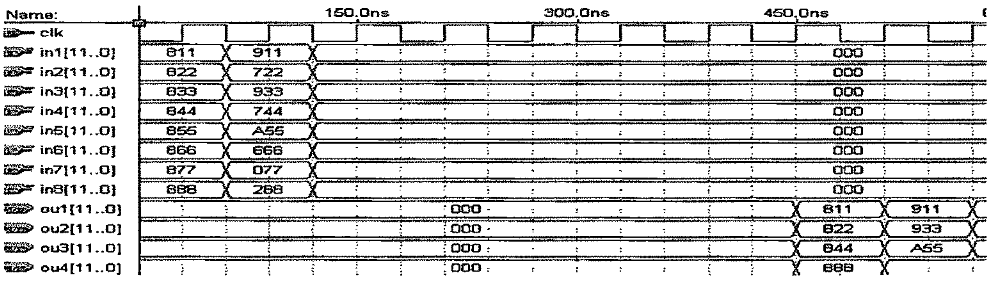

Figure 4.43 Waveform of 8x4 Concentrator.

4.6.1.8 Output module unit

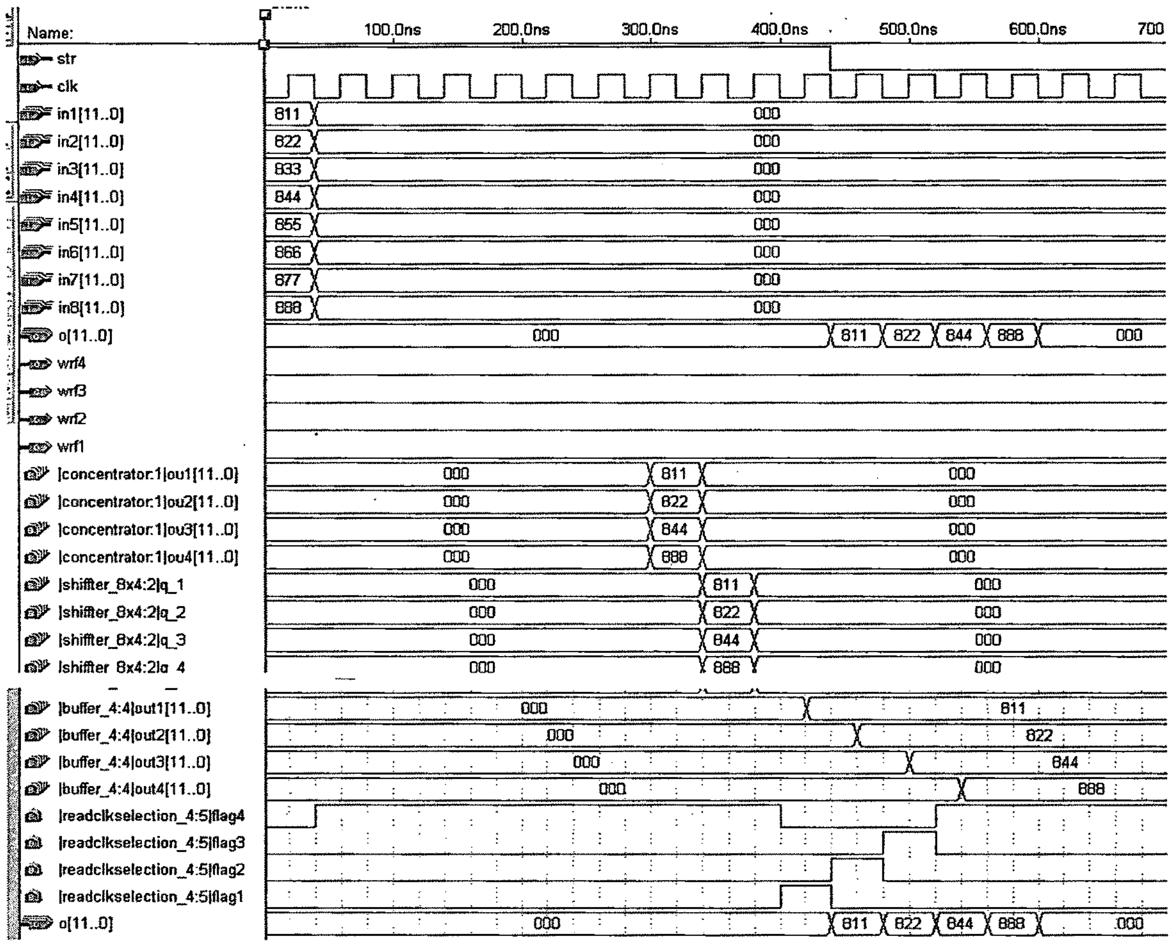

Figure 4.44 Waveform of output module unit for output 1.

4.6.1.9 Blocks required for generalized NxN Knockout switch

Table 4.2 shows the number of block required for NxN Knockout switch. Where L is the numbers of concentrator output line.

Table 4.2 Numbers of blocks required for NxN Knockout switch

Block	Quantity
Filter	N
Concentrator	N
Shifter	N
Output selector	N
Buffer	N x L
Buffer read clock Generator	2 x N
Buffer write clock generator	N

Numbers of blocks required of concentrator for different value of L for 8x8 Knockout switch are as shown in the table 4.3.

Table 4.3 Block requirement for 8x8 Knockout switch.

Block	Quantity for 8x2 concentrator	Quantity for 8x4 Concentrator	Quantity for 8x6 concentrator	Quantity for 8x8 concentrator
Contention	13	22	27	29
Delay block	4	14	29	32

4.6.1.10 VLSI area requirement of 8x8 Knockout switch

- Table 4.4 shows the VLSI area requirement of 8x8 knockout switch for different value of L.

Table 4.4 8x8 VLSI area requirement for 8x8 Knockout switch

Project	Device	Total Logic Elements	Total Pins	Total Memory Bits	Maximum Clock frequency in MHz
Knockout_8X2	EP20k1500 EBC652-1	3,326 / 51,840 (6%)	194 / 488 (40 %)	1536 / 442,368 (1 %)	58.65
Knockout_8X4	EP20k1500 EBC652-1	9,224 /68,416 (18 %)	194 / 488 (40 %)	3,072/1,152,000 (< 1 %)	40.84
Knockout_8X6	EP20k1500 EBC652-1	17,571 / 68,416 (34 %)	194 / 488 (40 %)	4,608/1,152,000 (< 1 %)	24.46
Knockout_8X8	EP20k1500 EBC652-1	25,475 / 68,416 (49 %)	194 / 488 (40 %)	6,144/1,152,000 (1.5 %)	18.83

4.6.2 IMPLEMENTATIONS OF 8x8 BATCHER BANYAN SWITCH

For testing of the hardware of Batcher Banyan switch, 12-bit packet format as shown in figure 4.36, is taken.

4.6.2.1 Building blocks of 8x8 Batcher Banyan switch

This section explains various building blocks of 8x8 Batcher-Banyan switch.

4.6.2.1.1 Banyan switching element

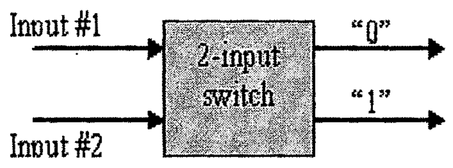

Figure 4.45 2x2 Banyan switching element.

It checks the active bit and address bit of input packet and depending on that it switches the packet to output 1or 2 as shown in figure 4.45.

Figure 4.46 shows the waveform of 2x2 Banyan switching element.

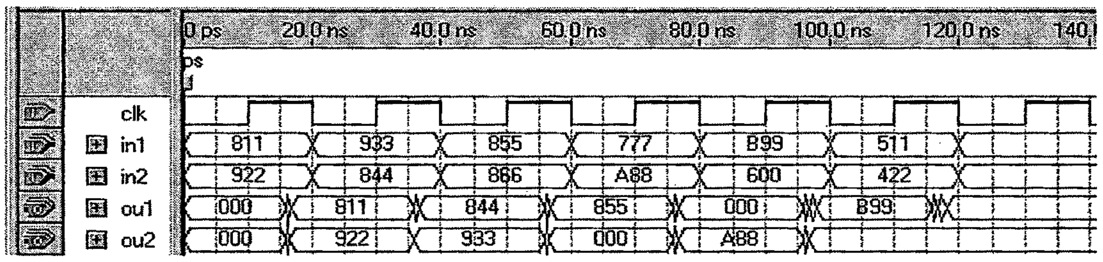

Figure 4.46 Waveform of 2x2 Banyan switching element.

4.6.2.1.2 8x8 Banyan Switch

As shown in the figure 4.47, only 2 packets gets out of 8 because of internal blocking and output conflict.

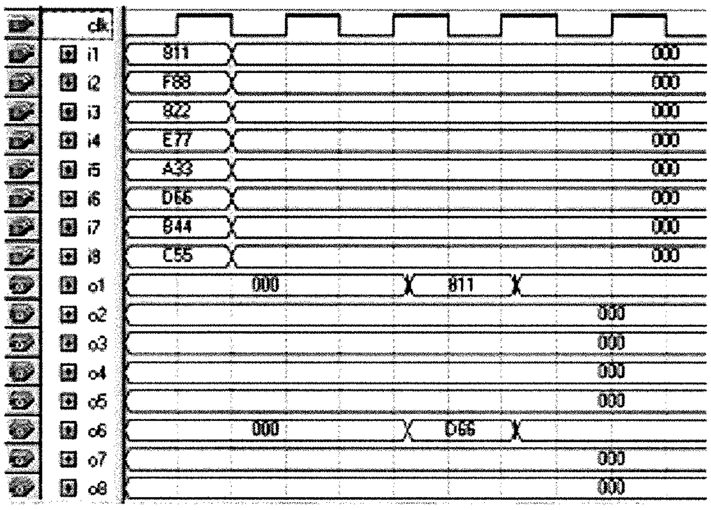

Figure 4.47 Waveform of 8x8 Banyan switch.

4.6.2.1.3 2x2 Batcher Up module

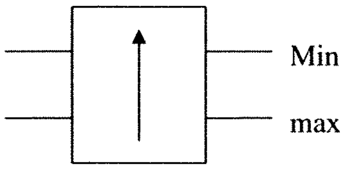

Figure 4.48 2x2 Batcher up module

2x2 Batcher Up module checks active bit and address bit. As shown in figure 4.48, it passes the higher value addressed packet to output 2 and lower value addressed packet to output1. Figure 4.49 shows the waveform of 2x2 Batcher Up module.

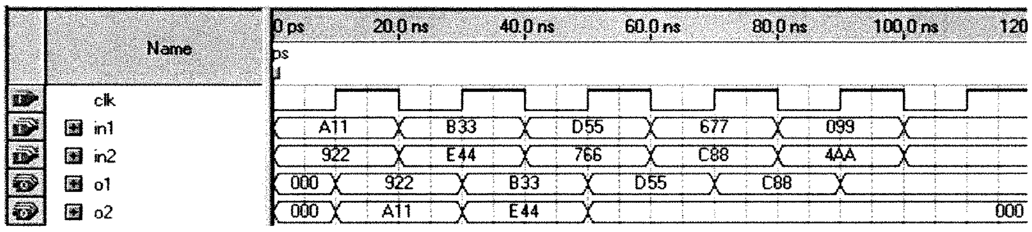

Figure 4.49 Waveform of 2x2 Batcher Up Module.

4.6.2.1.4 2x2 Batcher Down module

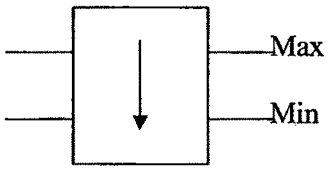

Figure 4.50 2x2 Batcher Down module

It is same as 'up' module, but it transfers higher addressed packet to the output 1 and lower addressed packet to the output 2 as shown in figure 4.50. Figure 4.51 shows the simulation waveform of 2x2 Batcher Down modules.

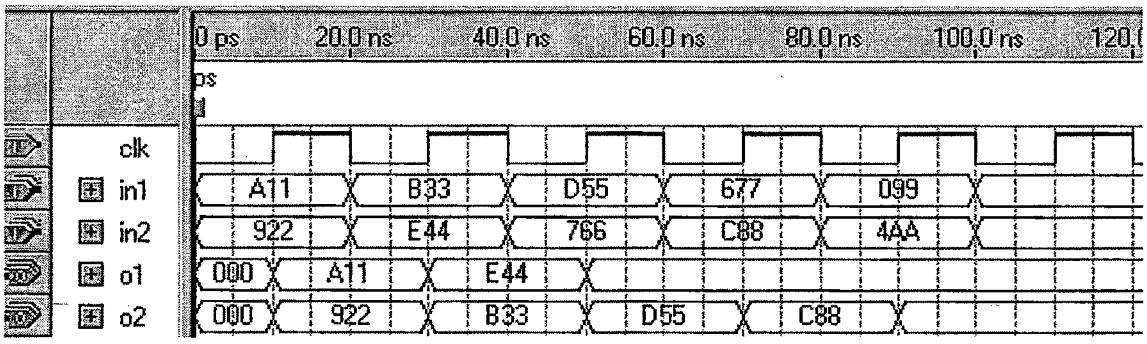

Figure 4.51 Waveform 2x2 Batcher Down Module.

4.6.2.1.5 8x8 Batcher sorter

It checks the active bit and address bit. If active bit is '1', then it sorts the packet according to destination address. Figure 4.52 shows the waveform of 8x8 Batcher sorter.

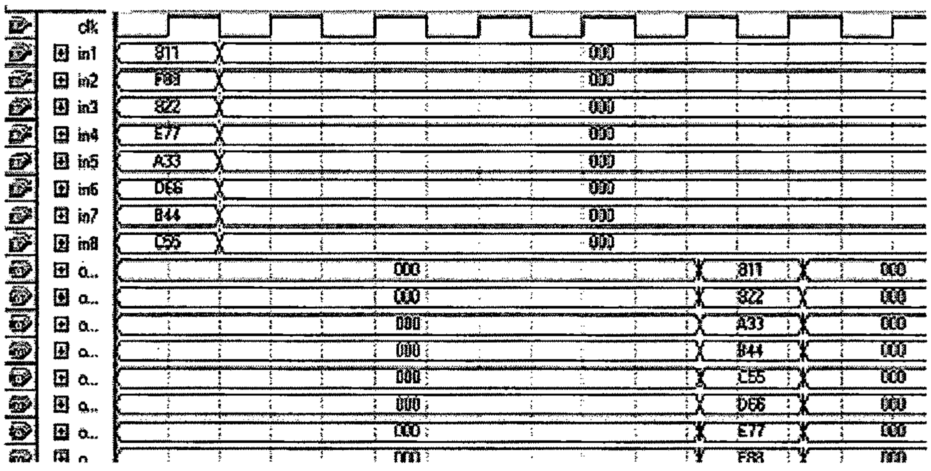

Figure 4.52 Waveform of 8x8 Batcher sorter.

4.6.2.1.6 8x8 Batcher Banyan switch

Internal blocking of Banyan switch can be avoided by using Batcher Banyan switch. As shown in figure 4.53, 8x8 Batcher Banyan switch gets 7 packets out of 8 instead of 2 as in the case of Banyan switch.

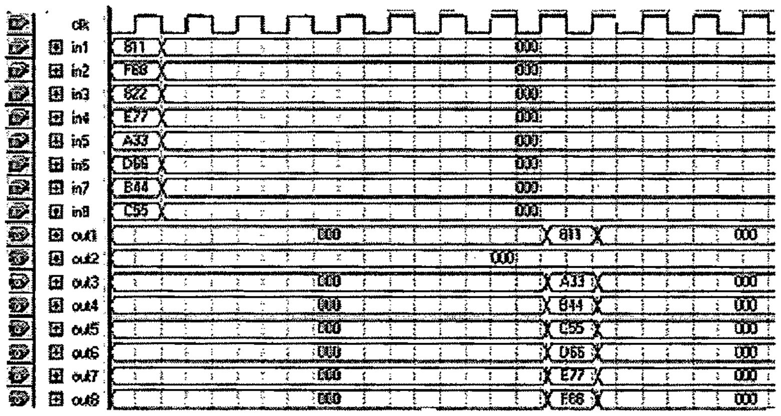

Figure 4.53 Waveform of 8x8 Batcher Banyan switch.

4.6.2.1.7 16x16 Trap switch

Concept of Trap is to trap the blocked packets in the current arbitration cycle and give them a chance to compete for the output ports in the next arbitration cycle. This in turn, increases the throughput (efficiency) of Batcher Banyan switch. Figure 4.54 shows waveform of 16x16 trap switch.

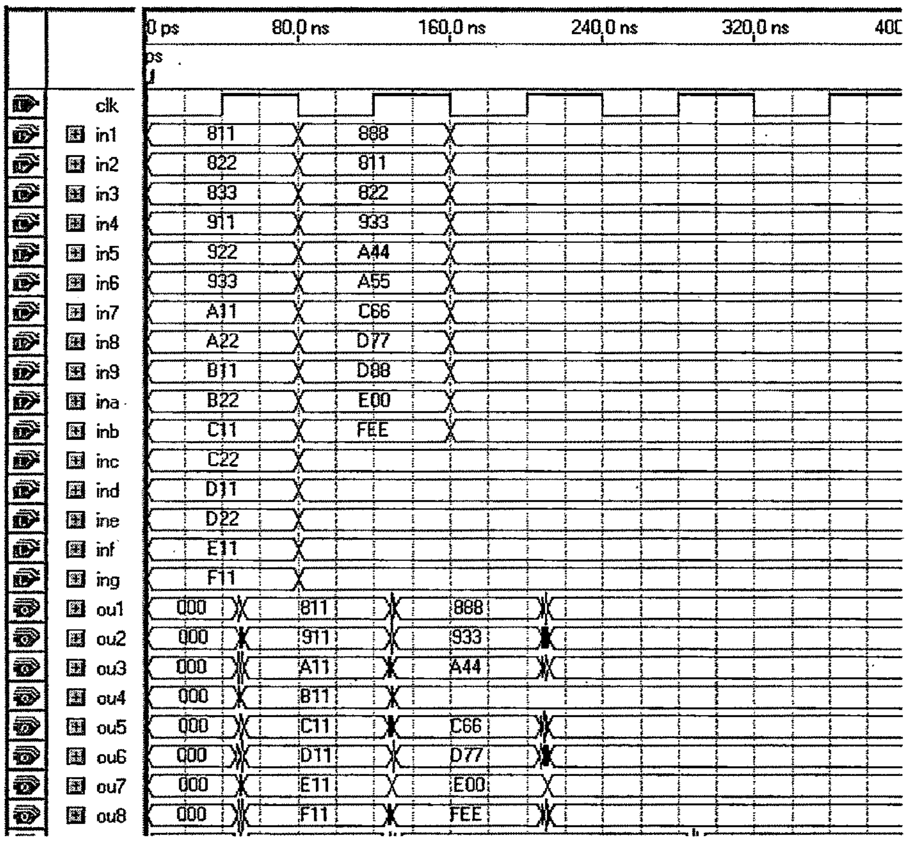

Figure 4.54 Waveform of 16x16 Trap switch.

4.6.2.1.8 8x8 Batcher Banyan switch with Trap

As shown in the figure 4.55, 8x8 Batcher Banyan Trap switch outputs 8 packets out of 8. Hence, output conflict problem in the Batcher Banyan switch is removed by using Trap network.

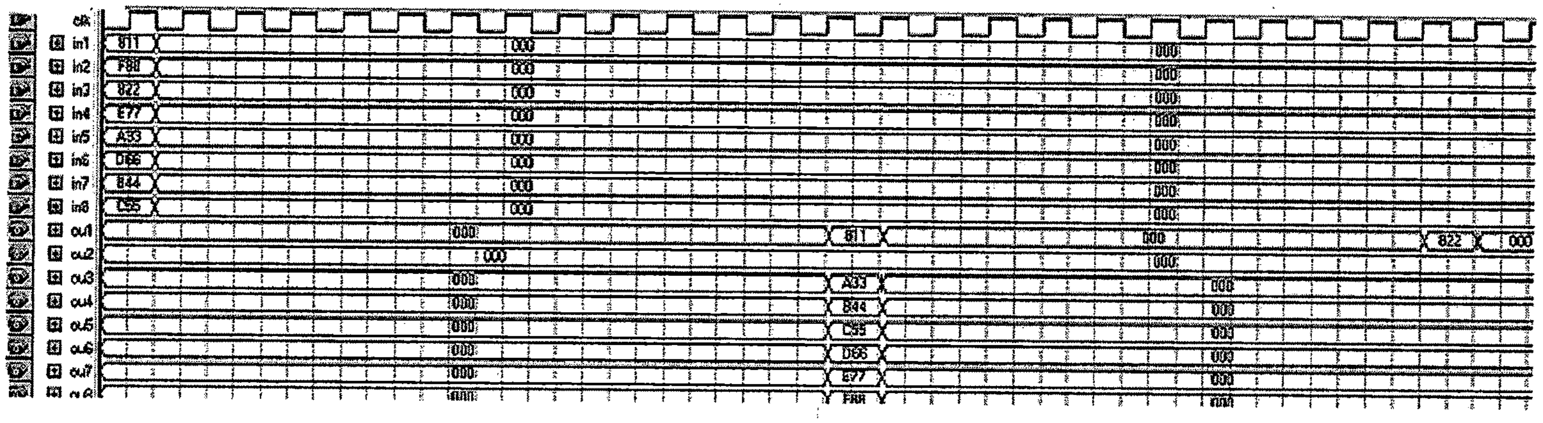

Figure 4.55 Waveform of 8x8 Batcher Banyan switch with Trap

4.6.2.2 Required 2x2 switching elements

- Required number of 2x2 switches for NxN Banyan switch

  NxN Banyan switch has a $\log_2 N$ stages, with N/2 2x2 switches per stage. The total number of 2x2 switches required to implement this class of multistage switches is $S = N/2 * \log_2 N$
- Required number of 2x2 module for NxN Batcher sorter

  $M = ½*\log_2 N [\log_2 N+1]$
- Required number of 2x2 switches for NxN Batcher Banyan switch

  $S_{B-B} = N/4 * \log_2 N [3+\log_2 N]$

4.6.2.3 VLSI area requirement of 8x8 Batcher Banyan switch

Table 4.5 summarizes the VLSI area requirement of 8x8 Banyan, Batcher Banyan and Batcher Banyan Trap switches.

Table 4.5 VLSI area requirement for 8x8 Banyan/Batcher-Banyan/Batcher-Banyan-Trap switches

Project	Device	Total Logic Elements	Total Pins	Total Memory Bits	Maximum Clock Frequency in MHz
8x8 Banyan switch	EP20k1500 EBC652-1	552 / 51,840 (1%)	193 / 488 (40 %)	0 / 442,368 (0 %)	156.76
8x8 Batcher Banyan switch	EP20k1500 EBC652-1	1,982 / 51,840 (4 %)	193 / 488 (40 %)	0 / 442,368 (0 %)	94.47
8x8 Batcher Banyan Trap switch	EP20k1500 EBC652-1	8,240 / 51,840 (16%)	193 / 488 (40 %)	0 / 442,368 (0 %)	25.48

4.7 Comparison of Batcher Banyan Trap and Knockout Switches

Table 4.6 Compares Batcher-Banyan-Trap and Knockout switches.

Table 4.6 Comparison of 8x8 Batcher-Banyan-Trap and knockout switches

8x8 Switches	Total Logic Elements (Device EP20k1500 EB C652-1)	Total Memory Bits	Maximum clock Frequency (In MHz)	MATLAB Simulation Speed (in Second)	Throughput (efficiency) in Percentage
Batcher Banyan Trap	8,240 / 51,840 (16%)	0 / 442,368 (0 %)	25.48	1.86	92.853
Knock Out L=4	9,224 /51,840 (18 %)	3,072/442,368 (< 1 %)	40.84	0.18	94.945
Knock Out L=8	25,475 / 51,840 (49 %)	6,144/442,368 (1.5 %)	18.83	0.20	98.764

4.8 SUMMARY

In this chapter, we have discussed simulation mechanism and data generation using different traffic models. We have simulated and implemented Knockout and Banyan/Batcher-Banyan/Batcher-Banyan-Trap switches. Performance of these switches under different traffic distribution is analyzed and comparison is done on the basis of VLSI area requirement. From the table 4.6, we can state that knockout switch requires more VLSI area but gives high throughput at high speed.