

Multi-Channel Deflection Crossbar (MCDC): A VLSI Optimized Architecture for Multi-Channel ATM Switching

Peter Y. Yan, Kyeong Soo Kim, and Paul S. Min
Department of Electrical Engineering
Washington University in St. Louis, St. Louis, MO 63130 USA
(FAX) 1-314-935-7500, (E-mail) {yan,kks,psm}@rgit.wustl.edu

Manjunath V. Hegde
Department of Electrical and Computer Engineering
Louisiana State University, Baton Rouge, LA 70803 USA
(FAX) 1-504-388-5200, (E-mail) bandit@gate.ee.lsu.edu

Abstract

We propose the Multi-Channel Deflection Crossbar (MCDC) as a promising architecture for broadband ATM switching under the realistic constraints imposed by VLSI technology. MCDC is an internally non-blocking, buffer efficient, crossbar based architecture. MCDC is also capable of multi-channel switching with cell sequence integrity preserved. Through a comparative study against a well-known ATM switch architecture, we show that the proposed MCDC architecture is able to incorporate a greater number of input channels and buffers for a given IC die size and requires a lower clocking rate.

Keywords: Multi-Channel, Deflection Routing, Crossbar, Concentrator, Shared Buffering, ATM Switch, VLSI, Cell Sequence Integrity, Zero cell-loss channels.

1 Introduction

We have recently proposed and implemented a multi-channel ATM switching architecture, the Washington University Multi-Channel Switch (WUMCS) [7]. A notable feature of WUMCS is that any $k(\leq 16)$ channels can be configured to form a logical bit pipe running at $k \times 155$ Mbps which guarantees cell sequence integrity while allowing cells to traverse over distinct physical paths [6].

We now propose a architecture that can provide a significantly larger number of channels and significantly higher channel speed over the WUMCS architecture and other switches [1] [4].

The development of MCDC is motivated by the need for a switch architecture which is capable of supporting an extremely high throughput (i.e., size and speed) when implemented in a single IC. This approach re-

duces the IC count requirement in a switch system, which translates directly to a lower cost and system implementation complexity. Once we are able to develop a switch architecture which is VLSI optimized, we can utilize a *Multistage Interconnection Network (MIN)* of such switch ICs.

MCDC implements the multi-channel switching capability, which has been proposed and studied [2] [3] [5], as a means of alleviating the processing speed constraint of electronic switches in the broadband networks, and allowing for the construction of a logical bit pipe whose bandwidth is higher than can be implemented electronically in a single channel.

2 Multi-Channel Deflection Crossbar

In Figure 1, an MCDC with 8 input and 8 output channels ($N = 8$) is shown. It consists of the Routing Crossbar and the Concentrator. Recirculation paths connect the output of the Concentrator to the input of the Routing Crossbar. Arriving cells enter MCDC from the input channels located at the bottom left portion of the Routing Crossbar and are destined for the output channels located at the top portion of the Routing Crossbar.

The output channels are allocated to certain Group Numbers (GNs). Note that there are three output channels allocated to $GN = 3$, whereas for $GN = 0, 1, 2, 4, 5$, only one output channel is allocated to each. Any cell destined for $GN = 3$ can exit from one of the three output channels assigned to the group, and thus, MCDC can be configured as a multi-channel switch.

Figure 2 shows a Routing Crossbar crosspoint, (2×2 element) which can be implemented with a latch and simple combinational logic circuit performing a logical operation, $A \stackrel{?}{=} B$. The result of the logic operation

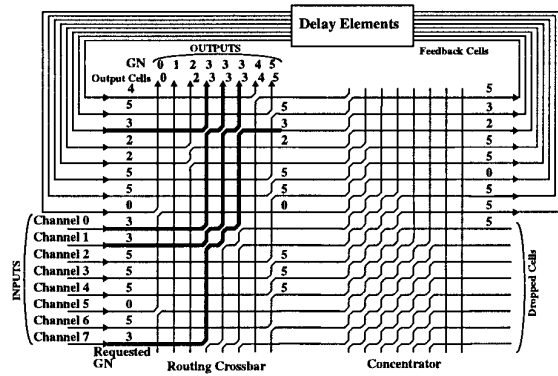


Figure 1: Multi-Channel deflection crossbar architecture.

determines the state of the 2×2 element. Readers may view A as the GN of the vertical line (i.e., the output address of the column), and B as the requested GN of the cell appearing on the horizontal line where the 2×2 element is located.

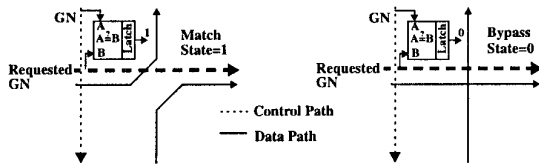


Figure 2: Routing crossbar switch element.

The main function of a 2×2 element is to determine the connection between the two inputs located at the left and bottom of the crosspoint and two outputs located at the right and top of the crosspoint. Each 2×2 element can be in one of the following two states:

- **Match State:** This state corresponds to $A = B$. The left input is connected to the top output; the bottom input is connected to the right output.
- **Bypass State:** This state corresponds to $A \neq B$. The left input is connected to the right output; the bottom input is connected to the top output.

At each crosspoint, the $A \stackrel{?}{=} B$ operation takes place simultaneously, and if a match is found, the corresponding 2×2 element goes into the match state. Otherwise, it goes into the bypass state. In each time slot, the states of 2×2 elements are computed first (distributedly and in parallel), and then the cells traverse along the paths determined by the computed states

(denoted as the data path in Figure 2). A cell initially moves horizontally rightward until one of two conditions occur: (1) it encounters a 2×2 element in the match state, at which point it starts to move vertically upwards. While traveling upwards, if a cell encounters a 2×2 element in the match state, it is deflected to the right, traveling horizontally rightward again. (2) it does not encounter a 2×2 element in the matched state; the horizontal movement continues and the cell arrives at the input to the Concentrator to be concentrated and recirculated.

For example, the cell appearing on input channel 0 with the requested GN=3 moves right until it reaches the first column labeled as GN=3. At this crosspoint, the 2×2 element is in the match state, and the cell moves upward. The movement continues until it encounters another 2×2 element above in the match state, at which point the cell is deflected to the right. Since the next 2×2 element it encounters is also in the match state, the cell moves upward again, and it reaches an output channel with GN=3.

In order for a cell to move vertically, it must be routed through a 2×2 element in the match state. Thus, no cell with a requested GN which does not match the GN of a particular output can exit from that output. The state of a 2×2 element is determined solely based on the requested GN on the horizontal line it is on. This means that a cell must make a vertical turn at the first 2×2 element in the match state, and therefore, the leftmost output for a GN is utilized by the cell appearing at the highest row with the equal requested GN. For a cell arriving at the output, the number of times that the cell is deflected to the right equals the number of cells above it with the same requested GN. Suppose that there are k output channels assigned to the same GN. Consider a tagged cell arriving at the i^{th} leftmost output in the GN. The consecutive nature of the output assignment indicates that the tagged cell must have been deflected $i - 1$ times to the right by the $i - 1$ cells appearing at horizontal rows located above. From this we see that output channels with the same GN are utilized from the left to right. Since all 2×2 elements with the same GN go into the match state at the same time for a given row, we can also see that no cell may enter into the Concentrator while there is a free output channel with the same GN.

MCDC is thus an internally nonblocking multi-channel switch. The cells that reach one of the outputs of the Routing Crossbar have been delivered to the correct output and can exit MCDC in the current time slot. Cells which have been deflected and failed to reach one of the outputs of the Routing Crossbar are delivered to the Concentrator.

The Concentrator recirculates failed cells to the input of the Routing Crossbar in order to vie for the outputs again in the following time slot. As shown in Figure 3, the state of each 2×2 element is determined by a logic operation, $A + \bar{B}$, where $+$ is the logical OR operation. A is the state of the 2×2 element directly above, i.e., $A = 1$ if it is in the match state, and $A = 0$ if it is in the bypass state. B indicates whether or not there is a cell arriving from the left at the crosspoint, i.e., $B = 1$ if there is a cell, and $B = 0$ otherwise. If the result of the operation $A + \bar{B}$ equals 1, then the 2×2 element goes into the match state. Otherwise, it goes into the bypass state.

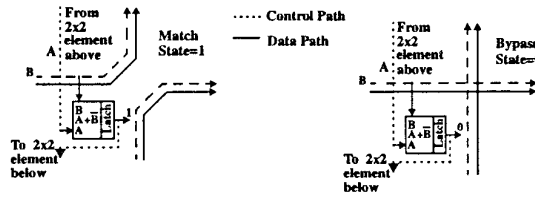


Figure 3: Concentrating crossbar switch element.

It is easy to see that the number of rows that a cell can move up in the Concentrator equals the number of empty rows above it, and this cannot exceed the number of columns in the Concentrator. Since in a time slot at most N cells can exit MCDC, if there are N columns in the Concentrator, then no cell is lost from MCDC when there are idle recirculation paths.

In the Concentrator, the state of a 2×2 element is determined only through combinational logic, and thus synchronization between the two inputs A and B are not required; after a sufficient amount of time, all electrical signals have been propagated and the correct states of the 2×2 elements have been computed. At this point, the states in the 2×2 elements can be latched and the content of cells can be routed over the computed paths (the datapath). Therefore, the latches are clocked only at the end of path computation (i.e., once per time slot), which reduces the frequency of synchronization, and in turn reduces power consumption and synchronization problems. This translates to a potentially higher clocking rate possible for data transfer.

The recirculation paths shown in Figure 1 provide shared buffering for the cells experiencing output contention. These cells are fed back to the input in order to vie for the output with the freshly arriving cells in the next time slot. Delay elements are needed to provide separation between time slots.

We now summarize MCDC characteristics.

- MCDC is internally nonblocking and buffer efficient: No output can be idle if there is a cell destined for it, and no cells are lost when there exists an empty buffer slot.
- Assignment of a GN to each output is software configurable, and any number of consecutive outputs can be grouped to form a channel group. Thus, MCDC is a multi-channel switch which allows arbitrary (multi-rate) grouping of output channels.
- Since computation of paths are done using combinational logic simultaneously at all 2×2 elements, the processing of headers is done in a parallel and distributed manner. This increases the processing rate of the switch in computing the data paths.
- In the Routing Crossbar and Concentrator, due to the manner by which states is determined, cells at higher horizontal positions have priority over those below when computing data paths. Also recirculated cells appear at higher horizontal positions than those at input channels. These facts have the following significance: Suppose for example, input channels 0 and 1 are grouped together. As shown in Figure 1, in a time slot, two cells belonging to the same session appear at the two input channels. If the cell generated earlier in time is associated with a higher horizontal position (i.e., channel 0), then, the operation of MCDC dictates that the cell generated earlier in time reaches the output of MCDC no later than the other cell (i.e., the one entering from input channel 1). If both cells reach the output in the same time slot, then the one generated earlier appears to the left of the other. In this view, time sequence is equated with the top-down ordering among the input channels and is equated with the left-right ordering among the output channels. Cells unable to exit in the previous time slot are fed back via the recirculation paths, which are located above the input channels. Since the top-down ordering is never altered along the paths of the Routing Crossbar and Concentrator, the time sequence between the cells can be maintained from the implicit relationship between the location of input and output channels at which they appear. MCDC thus provides multi-channel switching with cell sequence integrity.

3 Scalability in an IC

In order to clearly demonstrate the merits of the proposed MCDC architecture in these aspects, we utilize a specific switch architecture known in the literature as the Batcher-banyan network [1]; the Batcher

network sorts the arriving cells into a monotone ascending sequence according to their destination addresses. The sorted cells can then be delivered to the destination addresses via the banyan network, which is nonblocking with respect to the sorted sequence without output contention. This comparison is fair in the sense that they are both internally nonblocking and efficiently utilize shared buffering.

3.1 Scaling Number of Channels in an IC

An important question to be answered is the maximum number of channels that can be incorporated in an IC. Suppose that the transmission rate of each channel is fixed at a certain value, for instance, 155 Mbps. If we fix the IC die size, how do different architectures influence the maximum number of channels that can be incorporated?

One advantage of the Batcher-banyan network, is that it requires a lower number of 2×2 elements than other well known architecture such as the crossbar which requires N^2 elements (we have ignored the complexity in the trap operation in the Batcher-banyan network for simplicity.) However, in the VLSI environment, die size is determined from the combination of gate and wire areas, and thus the number of 2×2 elements alone is not sufficient to determine the efficacy of an architecture. In [8], a comparison is made for the die size requirement between the banyan and crossbar networks in the context of a circuit-switched interconnection of multi processor systems. It is shown in [8] that the ratio of die size requirement between the two architectures is bound by a small factor which is linear in N , and becomes constant as N gets large. The Batcher network has similar interconnection patterns as the banyan, but requires a factor of $\log_2 N$ higher number of 2×2 elements. From this, it is suspected that the ratio of die size between the Batcher-banyan and crossbar architecture would be in favor of crossbar, even in the context of ATM switching.

Figure 4 shows an example of the Batcher-banyan architecture.

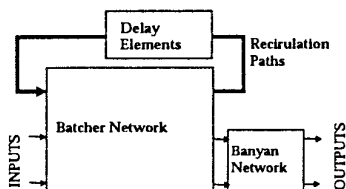


Figure 4: Batcher-banyan network.

Let us, for the sake of simplicity, assume that the 2×2 sort elements in the Batcher network are of

roughly the same hardware complexity as the other 2×2 elements in the banyan network and MCDC.

The number of 2×2 elements needed for the Batcher networks with N inputs is $0.25N \log_2 N (\log_2 N + 1)$. While not identical functionally, we combine an additional $0.5N \log_2 N$ 2×2 elements in the banyan network, resulting in a total of $0.25N \log_2 N (\log_2 N + 1) + 0.5N \log_2 N$ elements. The number of 2×2 elements needed for the Batcher-banyan architecture with M inputs (N external inputs and $M - N$ inputs from the shared buffers) is $0.25M \log_2 M [\log_2 M + 1] + 0.5N \log_2 N$. The number of 2×2 elements in MCDC is $2NM$. For $N = 16$ and $M = 64$, there are 704 2×2 elements required for the Batcher-banyan whereas 2048 2×2 elements are needed for MCDC.

However, if the long wire length and complex interconnection patterns in the Batcher-banyan architecture is considered, we can show that the die size requirement for the two architectures can change significantly. See Figure 5 for a very efficient arrangement of a 16 input banyan network to be incorporated in an IC. In the figure, we assume that there are two metal layers which can be used for interconnection of 2×2 elements. From the arrangement shown in the figure, we see that between stages i and $i + 1$, 2^i equals the minimum possible number of vertical line spaces; for example, between stages 1 and 2, there are 3 ($\geq 2^1$) vertical line spaces, between stages 2 and 3, there are 4 ($\geq 2^2$), and between stages 3 and 4, there are 9 ($\geq 2^3$). We assume that the number of vertical line spaces needed between the stage i and $i + 1$ equals 2^i .

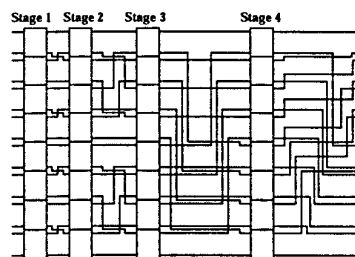


Figure 5: An efficient arrangement for the banyan network.

Each vertical line is assumed to be w parallel wires over which data can be transferred simultaneously. It is required that these wires are separated by space which is approximately equal to the width of a wire. Suppose that the feature size of the VLSI technology is denoted as 2λ . Then, a common design rule specifies that the wire width (and therefore also the space between wires) is 3λ , which makes the vertical spacing requirement

between stages i and $i + 1$ to be $3\lambda(2^{i+1}w + 1)$.

Let the total width for the banyan network with N channels to be $W_b(N)$. Then,

$$W_b(N) = \left[\sum_{i=1}^{\log_2 N - 1} 3\lambda(2^{i+1}w + 1) \right] + 3\lambda(2^{\log_2 N}w + 1) \quad (1)$$

where the extra term is needed to account for the shuffle pattern in the wires after stage $\log_2 N$.

Suppose that each 2×2 element is implemented in an area of dimension $l_s \times l_s$. We note that between stages i and $i + 1$, there are N horizontal lines for all $1 \leq i \leq \log_2 N - 1$. If we assume that $l_s > 12w\lambda$ (that is, a side of 2×2 element is sufficient to fit at least two horizontal lines, which is most likely the case) then the 2×2 elements can be abutted in a column for each stage. Therefore, the height (denoted as $H_b(N)$) of the banyan network,

$$H_b(N) = \frac{N}{2}l_s \quad (2)$$

From (1) and (2), the area requirement for the banyan network with N channels is $A_b(N) = W_b(N) \cdot H_b(N)$.

The Batcher network of size M is constructed from banyan networks of sizes, $2, 4, 8, \dots, M$. Therefore for the Batcher network, the width $W_B(M)$, the height $H_B(M)$, and the area $A_B(M)$ become equal to $A_B(M) = W_B(M) \cdot H_B(M)$ where $W_B(M) = \sum_{i=1}^{\log_2 M} W_b(2^i)$ and $H_B(M) = \frac{M}{2}l_s$. The total area for the Batcher-banyan network with N inputs and $M - N$ shared buffers, denoted as $A_{Bb}(N, M)$ is $A_B(M) + A_b(N)$. (We ignore the area for the recirculation paths since they are very similar for the Batcher-banyan architecture and MCDC.)

The area requirement for MCDC with the same input and shared buffer requirement is simple to calculate. Since in the crossbar architecture, all 2×2 elements can be abutted both vertically and horizontally, and there is no additional wire area requirement if $l_s > 12w\lambda$. Then the total area, $A_M(N, M)$, needed for MCDC with N inputs and $M - N$ shared buffers becomes

$$A_M(N, M) = 2MNl_s^2,$$

which account for the areas for the Routing Crossbar and the Concentrator. For M fixed at $4N$, $A_{Bb}(2, 8) = 0.08$ and $A_M(2, 8) = 0.06$; $A_{Bb}(4, 16) = 0.4$ and $A_M(4, 16) = 0.3$; $A_{Bb}(8, 32) = 1.8$ and $A_M(8, 32) = 1.0$; $A_{Bb}(16, 64) = 7.0$ and $A_M(16, 64) = 4.2$; $A_{Bb}(32, 128) = 25.4$ and $A_M(32, 128) = 16.6$;

$A_{Bb}(64, 256) = 92.9$ and $A_M(64, 256) = 66.4$; and $A_{Bb}(128, 512) = 334$ and $A_M(128, 512) = 265$ (given in mm^2 unit). In these calculations, we assumed $2\lambda = 0.6 \times 10^{-6}m$, $w = 8$, and $l_s = 150\lambda$, which are realistic values for the present state of VLSI technology. From these values, we can see clearly the die size advantage of MCDC. The current commercial VLSI technology can accommodate the die size requirement of an MCDC of size $N = 64$. If 2λ is reduced by half, i.e., $2\lambda = 0.3 \times 10^{-6}m$, then the die size requirement of an MCDC of size $N = 128$ is possible in an IC.

From the exercise in this section we conclude that for a given die size (e.g., 200 mm^2), it is possible to incorporate a greater number of input channels and buffers via the MCDC architecture than via the Batcher-banyan architecture. This is inconsistent with the traditional analysis of hardware complexity which is only based on the number of 2×2 elements. Therefore, when implemented in an IC, it is essential to consider wiring area.

3.2 Scaling Channel Speed in an IC

Let us investigate the case of scaling the bit rate of each channel utilizing the Batcher-banyan network and MCDC with N inputs and $M - N$ buffers.

Let b be the number of words in a cell to be transmitted (all bits of a word being transmitted on a parallel bus) sequentially, i.e., $b = \frac{53 \text{ bytes} \times 8 \text{ bits/byte}}{w}$. Suppose that switching is done such that each cell is routed through the switch in a single time slot. This means that in each time slot, the path for cells is computed first and then the cells are routed through the computed paths.

We consider the MCDC first. For N inputs, $\log_2 N$ clock cycles are needed to transmit the requested GN bits across the horizontal lines. Also, the concentrator requires $(N + M) \cdot d_s$ time to compute its states with combinatorial logic (where d_s is the delay of each 2×2 element). To see this, note that the worst case path is for the leftmost M Concentrator elements to compute down the M rows, followed by N elements to compute diagonally through the N columns.

At the end of the $\log_2 N$ clock cycles, and after $(N + M) \cdot d_s$ delay, all the paths have been computed and the (match or bypass) states of 2×2 elements have been set in the Routing Crossbar and the Concentrator.

Cells now can traverse through the computed paths. A little thought suggests that the longest path contains $(2N + M) 2 \times 2$ elements, each of which has delay of d_s . To increase the bit rate, we need to insert pipeline stages along the path and suppose there are p of such. Ignoring synchronization overhead, the clock period,

t_c , becomes,

$$t_c = \frac{d_s \cdot (2N + M)}{p} \quad (3)$$

The total number of clock cycles needed to route a b word long cell equals $\log_2 N$ cycles to set the paths, p cycles to route the first bit and $b - 1$ more cycles to route the rest. Thus

$$(\log_2 N + p + b - 1) \cdot t_c \leq T - (N + M) \cdot d_s \quad (4)$$

where T is the length of a single time slot which is a function of the external channel rate.

We are interested in determining the required clock period t_c from (4) for each value of T . By solving (3) for p and substituting the result in (4), we get,

$$t_c \leq \frac{T - d_s \cdot (3N + 2M)}{\log_2 N + b - 1} \quad (5)$$

The N -input, M - N buffer Batcher-banyan network requires $\frac{1}{2} \log_2 M (\log_2 M + 1) + 2 \log_2 N$ clock cycles to compute the paths.

To determine the worst case delay for cell routing, we note that the number of 2×2 elements for all the paths is equal to $\frac{1}{2} \log_2 M (\log_2 M + 1) + \log_2 N$. For all the paths, the total length of horizontal wires, $D_H(N, M)$, that cells traverse is

$$D_H(N, M) = W_B(M) + W_b(N) - \left(\frac{1}{2} \log_2 M (\log_2 M + 1) + \log_2 N \right) \cdot l_s \quad (6)$$

The total length of vertical wires is, however, dependent on the paths. It is easy to show that the total length of the vertical wires in the longest path, denoted as $D_V(N, M)$, equals

$$D_V(N, M) = \left[\sum_{j=1}^{\log_2 M} \left(\sum_{i=1}^{j-1} 2^{i-1} - 1 \right) \right] + 2^{j-2} - 1 + \left(\sum_{i=1}^{\log_2 N-1} 2^{i-1} - 1 \right) + 2^{\log_2 N-2} - 1 \cdot l_s$$

The total delay for the worst case path is $\left(\frac{1}{2} \log_2 M (\log_2 M + 1) + \log_2 N \right) \cdot d_s + (D_H(N, M) + D_V(N, M)) \cdot d_w$, where d_w is the delay on the unit wire length. Suppose once again, along this path, we insert p pipeline stages. Ignoring any synchronization overhead, the clock period, t_c , becomes,

$$t_c = \frac{1}{p} \left\{ \left(\frac{1}{2} \log_2 M (\log_2 M + 1) + \log_2 N \right) \cdot d_s + (D_H(N, M) + D_V(N, M)) \cdot d_w \right\} \quad (7)$$

We can again utilize an equation analogous to (4), and solve (7) for p and substitute it in the equation analogous to (4). The result of this operation is the required clock period t_c for a given value of channel speed which determines T .

We now determine the clocking rate for different N and T values. Suppose $w = 8$. If the channel rate is 620 Mbps, with $M = 4N$, we can see for $N = 16$, the minimum clocking rate needed for MCDC equals 90 MHz whereas it is 123 MHz for the Batcher-banyan. Now fix the channel rate at 155 Mbps and consider $N = 64$. Then, the minimum clocking rate needed for MCDC is 23 MHz and for the Batcher-banyan, it is 38 MHz.

4 Performance Analysis

In this section we analyze the performance of the MCDC routing networks.

4.1 System Modeling

Consider a routing network shown in Figure 6, where the number of input channels is N , the size of the shared buffer is R , the number of groups is G , and the group capacity of channel group i (i.e., the number of output channels for channel group i) is l_i , $i = 0, 1, \dots, G - 1$.

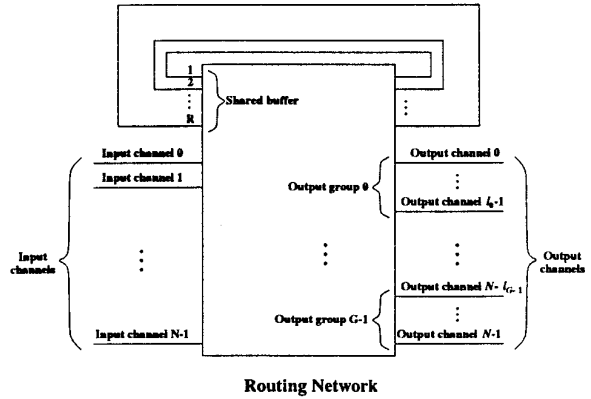


Figure 6: Model for routing networks.

At each input channel of the routing network, we assume that a cell arrives with probability p_a per time slot, independently of events in any other time slot and independently of events in any other input channel. The channel group numbers (GN) for different cells are assumed to be independent and identically distributed (i.i.d.) with probability $p_r(i)$ that the channel group number of the arriving cell is i .

First, we define G -dimensional random vector $\mathbf{x}(n)$ to describe the state of the shared buffer at the beginning of time slot n . The i^{th} element of the vector

$\mathbf{x}(n)$ denotes the number of cells in the shared buffer for channel group i . Therefore the state space for $\mathbf{x}(n)$ is given by

$$E_X = \{(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{G-1}) \mid \mathbf{x}_i \geq 0, \quad (8)$$

$$i = 0, 1, \dots, G-1, \text{ and } \sum_{i=0}^{G-1} \mathbf{x}_i \leq R\}$$

Since $\mathbf{x}(n+1)$ depends on $\mathbf{x}(n)$, and input cell arrival and routing processes which are independent of $\mathbf{x}(k)$, $k \leq n$, $\mathbf{x}(n)$ forms a Markov chain which we call M_A . To obtain the transition probabilities of M_A , we define another G -dimensional random vector $\mathbf{y}(n)$ which describes the state of the output channels at the end of the time slot n shown in Figure 7, where the $\mathbf{y}(n)$, $n = 1, 2, \dots, N$, denotes the state of the output channels just after the input channel $n-1$ is processed while $\mathbf{y}(0)$ represents the state of the output channels just after all the stored cells in the shared buffer is processed.

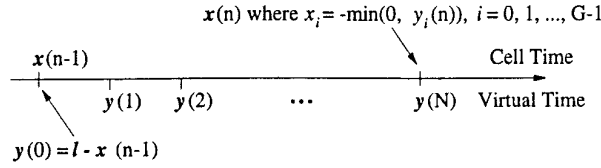


Figure 7: Relation between virtual time slot and cell Time.

The state space for $\mathbf{y}(n)$ is as follows:

$$E_Y = \{(\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{G-1}) \mid -R \leq \mathbf{y}_i \leq l_i, \quad (9)$$

$$i = 0, 1, \dots, G-1, \text{ and } -\sum_{i=0}^{G-1} \min(0, \mathbf{y}_i) \leq R\}$$

where the i^{th} element of the vector $\mathbf{y}_i$ denotes the state of the output channels for channel group i ; if $\mathbf{y}_i$ is greater than or equal to zero, it means that there are $\mathbf{y}_i$ channels available for transmission of the cells for channel group i ; if $\mathbf{y}_i$ is less than zero, it means that there are $-\mathbf{y}_i$ cells stored in the shared buffer for channel group i . Also $\mathbf{y}(n)$ forms a Markov chain, M_B , because $\mathbf{y}(n+1)$ depends on $\mathbf{y}(n)$, and input cell arrival and routing processes at each input channel. Note that the transition probabilities for $\mathbf{y}(n)$ are time-homogeneous because of the *i.i.d.* assumptions on the input arrival and the routing processes. Then, considering the timing diagram in Figure 7, we can obtain the transition probabilities of M_B as follows: For $\mathbf{s}, \mathbf{t} \in E_Y$,

$$P_{\mathbf{s}, \mathbf{t}} \triangleq P\{\mathbf{y}(n+1) = \mathbf{t} \mid \mathbf{y}(n) = \mathbf{s}\} \quad (10)$$

$$= \begin{cases} 1 - p_a + \sum_{k \in E_Y^-(\mathbf{s})} p_a \cdot p_r(k) & \text{if } -\sum_{i=0}^{G-1} \min(0, \mathbf{s}_i) = R \text{ and } \mathbf{s} = \mathbf{t}, \\ 1 - p_a & \text{if } -\sum_{i=0}^{G-1} \min(0, \mathbf{s}_i) < R \text{ and } \mathbf{s} = \mathbf{t}, \\ p_a \cdot p_r(i) & \text{if } \mathbf{s}_i = \mathbf{t}_i + 1 \text{ and } \mathbf{s}_j = \mathbf{t}_j \ \forall j \neq i, \\ 0 & \text{otherwise,} \end{cases}$$

where $E_Y^-(\mathbf{s})$ is the set of the indexes of the elements of the vector $\mathbf{s}$ which are less than or equal to zero. We also obtain the n -step transition probability $P_{\mathbf{s}, \mathbf{t}}^n \triangleq P\{\mathbf{y}(k+n) = \mathbf{t} \mid \mathbf{y}(k) = \mathbf{s}\}$ using the one-step transition probability $P_{\mathbf{s}, \mathbf{t}}$ and the following Chapman-Kolmogorov equations: For $n = 2, 3, \dots, N$,

$$P_{\mathbf{s}, \mathbf{t}}^n = \sum_{\mathbf{y} \in E_Y} P_{\mathbf{s}, \mathbf{y}}^{n-1} \cdot P_{\mathbf{y}, \mathbf{t}} \quad (11)$$

Now, considering the relations between $\mathbf{x}(n)$ and $\mathbf{y}(n)$ shown in Figure 7, we can obtain the transition probability of M_A as follows: For $\mathbf{u}, \mathbf{v} \in E_X$,

$$P\{\mathbf{x}(n+1) = \mathbf{v} \mid \mathbf{x}(n) = \mathbf{u}\} = \sum_{\mathbf{y} \in E_Y(\mathbf{v})} P_{\mathbf{l}-\mathbf{u}, \mathbf{y}}^N \quad (12)$$

where $\mathbf{l} = (l_0, l_1, \dots, l_{G-1})$, and

$$E_Y(\mathbf{v}) = \{(\mathbf{y}_0, \mathbf{y}_1, \dots, \mathbf{y}_{G-1}) \mid -\min(0, \mathbf{y}_i) = \mathbf{v}_i, \ i = 0, 1, \dots, G-1\} \quad (13)$$

With these transition probabilities of M_A , we can finally obtain the steady-state probability of the shared buffer, $\pi(\mathbf{x}) \triangleq \lim_{n \rightarrow \infty} P\{\mathbf{x}(n) = \mathbf{x}\}$.¹

4.2 Cell Loss Probabilities

Let $P_{L,i}$ be the conditional probability that a tagged cell is lost due to buffer overflow conditioned on that the tagged cell arrives at input channel i . Also, we define $P_{L,i,j}$ as the conditional probability that a tagged cell is lost due to buffer overflow conditioned on that the tagged cell arrives at input channel i and that the tagged cell has a channel group number j . Consider a tagged cell which arrives at input channel i with a channel group number j . Since the tagged cell is lost if there is no output channel for channel group j and no room in the shared buffer after processing all the input channels from 0 to $i-1$, $P_{L,i,j}$ is given by

$$P_{L,i,j} = \sum_{\mathbf{y} \in L_j} \left(\sum_{\mathbf{x} \in E_X} P_{\mathbf{l}-\mathbf{x}, \mathbf{y}}^i \pi(\mathbf{x}) \right) \quad (14)$$

¹For numerical calculation of $\pi(\mathbf{x})$, it is necessary to form a transition matrix for Markov chain M_B , which requires mapping of the multi-dimensional state space E_X into a corresponding one-dimensional state-space. Refer to [9] for the details.

where L_j is a subset of E_Y which is defined as follows: For $(y_0, y_1, \dots, y_{G-1}) \in E_Y$,

$$L_j \triangleq \{(y_0, y_1, \dots, y_{G-1}) \mid y_j \leq 0 \text{ and } -\sum_{k=0}^{G-1} \min(0, y_k) = R\} \quad (15)$$

Note that $P_{s,t}^0$ becomes $\delta(|s - t|)$ where $\delta(x)$ is 1 if $x = 0$ or 0 otherwise. So we can easily obtain P_{L_i} as follows:

$$P_{L_i} = \sum_{j=0}^{G-1} P_{L_i,j} p_r(j) \quad (16)$$

From these results, the cell loss probability P_L or the probability that a tagged cell is lost due to buffer overflow is given by

$$P_L = \frac{1}{N} \sum_{i=0}^{N-1} P_{L_i} \quad (17)$$

4.3 Numerical Results and Discussions

In Figure 8 we show the average cell loss probability (P_L) and the cell loss probability at each input port (P_{L_i}) together with the corresponding simulation results. We set the number of channels N to 8, the size of the shared buffer R to 4, and the number of groups G to 4 with $l_i = 2$, $i = 0, 1, 2, 3$.

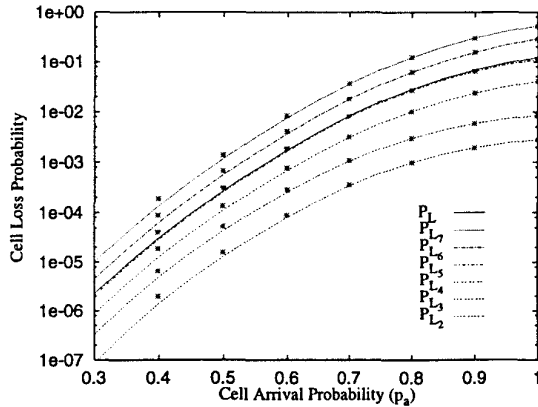


Figure 8: Cell loss probability vs. cell arrival probability for $N = 8$, $G = 4$, $R = 4$, and $l_i = 2$, $i = 0, 1, 2, 3$.

From this example we can verify the accuracy of the analytical results in this paper. Note that in Figure 8, there are no curves for the top two ports (IPN=0 and 1) which also corresponds to the configuration with $\min_i l_i = 2$ where $0 \leq i \leq G - 1$. The reason is that the cell loss probability for cells entering on the the top two ports is zero.

In [9] we show that with channel group i as the channel group with minimum group capacity ($l_i = l_{min}$), the first l_{min} input channels will have zero cell loss.

References

- [1] Giacomelli, James N. and Hickey, Jason J. and Marcus, William S. and Sincoskie, W. David and Littlewood, Morgan, "Sunshine: A High-Performance Self-Routing Broadband Packet Switch Architecture," *IEEE J. Selected Areas in Communications*, Vol. 9, No. 8, Oct. 1991, pp. 1289-1298.
- [2] Lin, A., and Silvester, J.A., "Fixed-Node Routing and Its Performance in ATM Networks," *Proceedings of IEEE INFOCOM '90*, June, 1990, pp. 803-810.
- [3] Pattavina, A., "Multichannel Bandwidth Allocation in a Broadband Packet Switch," *IEEE Journal of Selected Areas in Communications*, Volume 6, No. 9, Dec, 1988, pp. 1489-1499.
- [4] Turner, J. S., "Design of a Broadcast Packet Switch Network," *IEEE Trans. Commun.*, Vol. 36, No. 6, June 1988, pp. 734-743.
- [5] Min, P.S., Saidi, H., and Hegde, M.V., "Nonblocking Architecture for Broadband Multi-Channel Switching," *IEEE Transactions on Networking*, vol. 3, no. 2, pp. 181-198, Apr. 1995.
- [6] Min, P.S. and Saidi, H., "Nonblocking Multi-Channel Switching with Multicasting Capability," U.S. Patent No. 5,440,549, 35 claims, August 8, 1995.
- [7] Min, P.S., Chandra, A.E., Saidi, H., Yan, P.Y., and Zar, D.M., "Washington University Multi-Channel Switching (WUMCS) Project: System Architecture Document," *Technical Report*, Department of Electrical Engineering, Washington University, December 1993.
- [8] Franklin, M. A., "VLSI performance comparison of banyan and crossbar communications networks," *IEEE Trans. Comput.*, Vol. C-30, Apr., 1981.
- [9] P. Y. Yan, K. S. Kim, P. S. Min, and M. V. Hegde, "Multi-channel deflection crossbar (MDCD): A VLSI optimized architecture for multi-channel ATM switching," *Tech. Rep. WUEE-96-203*, Department of Electrical Engineering, Washington University, Aug., 1996.