Non-blocking Permutation Generator for Banyan Network

Joo-young Lee[1] and Jae-il Jung[2]
[1] Dept. of Electronic Engineering
Seokyeong University
16-1, Jungrung-dong, Sungbuk-gu, 136-704, Seoul, KOREA
Tel: +82-2-940-7490, Fax: +82-2-2293-0377
e-mail: jylee@skuniv.ac.kr
[2] Division of Electrical and Computer Engineering
Hanyang University
17, Hangdang-dong, Sungdong-gu, 133-791, Seoul, KOREA
Tel: +82-2-2290-0352, Fax: +82-2-2293-0377
e-mail: jijung@hanyang.ac.kr

Abstract Banyan network is a popular and basic structure of the multistage ATM switches. This paper presents a novel approach to resolve the internal blocking of the banyan network by using Non-Blocking Permutation Generator (NBPG). The NBPG performs two functions, i.e., the first is to extract the conflict cells from the incoming cells and the second is to re-assign new input port addresses to the conflict cells. As a result, NBPG generates non-blocking I/O permutations. To estimate the performance of NBPG, we provide several simulation results.

1. Introduction

Switching systems are central components in the communication network. Switching systems for ATM must address the fundamental issues of scalability, reliability and cost-effectiveness [1]. The multistage interconnection network, including the banyan network, has been one of basic structures for the ATM switch.

The banyan network is a popular method for realizing the self-routing fabric and has a simple and expandable structure. The banyan network is defined as a network with a unique path between each I/O pair, where each stage performs a fixed permutation on the incoming lines, and then routes them through 2 x 2 switching elements to the output.

The routing algorithm of the banyan network is as follows: each request contains an n-bit destination equal to the binary address of the requested output port. In the nth stage, a switch, which receives a request, examines the nth most significant bit in the destination and selects an output port based on the value of the bit. If the bit is 0, the upper output port is selected, and if the bit is 1, the lower output port is selected.

If only one request selects a particular output port, it is forwarded on the single output link. When two requests arrive at a switching element at the same time and select the same output port, the internal conflict occurs. In this case, one request is selected randomly and forwarded, and the other is blocked [2].

This internal conflict degrades the performance of the banyan network severely. A popular solution of this problem is to attach additional components. The additional components may be buffers, internal links, sort networks, and so on. In this paper, we have focused our interest on the sort network, since it seems to be a better solution to resolve the internal conflict of the banyan network.

The sort network, e.g., Batcher bitonic sort network [3], which rearranges incoming cells with their input addresses in increasing order before the banyan network [4]. However, Batcher sorter not only requires lots of hardware components, but also realizes only one type of the non-blocking I/O permutation pattern for the sake of the non-blocking transmission.

In this paper, we propose a new non-blocking permutation generator (NBPG) for the banyan network. Firstly, it classifies incoming cells into the conflict and non-conflict cells to indicate whether incoming cell undergoes an internal conflict or not. Secondly, we re-assign new input addresses to the conflict cells. All these processes are performed on the tables that we introduce.

Section 2 describes the relationship between a given I/O pair and conflict I/O pairs. In section 3, we present the conflict cell decision process using the conflict table. In section 4, we also present the input port re-assign process using the input port re-assign table. In section 5, the simulation results are discussed. Finally, a conclusion is given.

2. Internal Blocking in Banyan Network

The N x N banyan network has n ($= log_2 N$) stages, where each stage contains $N/2$ switching elements (SE's). For a given banyan network, up to N I/O port connections can be established concurrently.

The total number of non-blocking I/O permutations of N concurrent connections is equal to $2^{nN/2}$ because each SE has two basic non-blocking states. as shown in Figure 1.

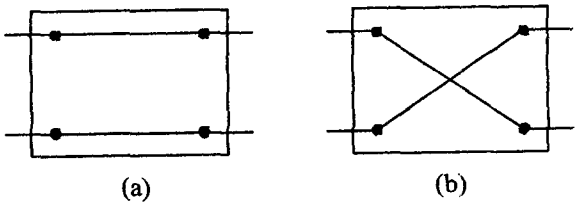

Figure 1. The basic states of the banyan network. (a) Parallel state. (b) Cross state.

Figure 2 illustrates an 8 x 8 banyan network. It has three stages and four SE's for each stage. And inter-connections between stages, are one shuffle exchange and several banyan exchanges. The stages are labeled in a sequence from 0 to n-1 (n = 3). I/O links are labeled using n binary digits ($a_0 \cdots a_{n-1}$). Using the topology describing rule [5], the interconnections between SE's can be given by the followings.

$$S[(a_0 \cdots a_{n-1})] = (a_1 \cdots a_{n-1}\, a_0) \quad (1)$$

$$B_k[(a_0 \cdots a_{n-1})_k] = (a_0 \cdots a_{k-1}\, a_{n-1}\, a_{k+1} \cdots a_{n-2}\, a_k)_{k+1} \quad (2)$$

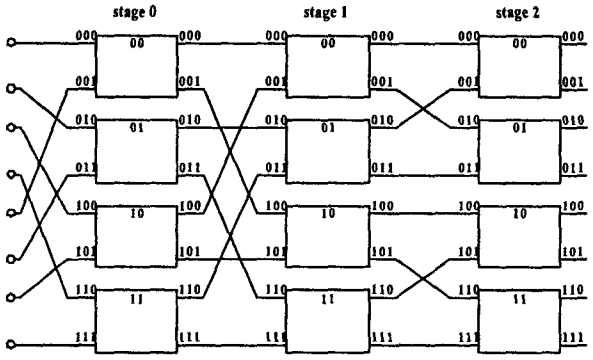

Figure 2. 8 x 8 banyan network.

Note that S describes the s huffle interconnection and B_k's the banyan interconnections by mapping an SE in stage k to an SE in stage k+1. $(a_0\, a_1 \cdots a_{n-1})_k$ means an I/O link at stage k. Throughout the rest of this paper, let $(s_0 \cdots s_{n-1})$ and $(d_0 \cdots d_{n-1})$ be the binary representation of source and destination address, respectively.

In theory of [6], Bhuyan, N. L. and Agrawal, P.D. showed the simple conflict condition of generalized shuffle network, but we extend their theorem to the banyan network and provide a new condition for the network.

The proposed conflict condition of the banyan network can be formulated as follows. x means '0' or '1', i.e., don't care condition. $\bar{s_k}$ means the complement of s_k.

Lemma: A given I/O pair $(s_0 \cdots s_{n-1} - d_0 \cdots d_{n-1})$ always conflicts with I/O pairs $(x_0 \cdots x_{k-1}\, \bar{s_k}\, s_{k+1} \cdots s_{n-1} - d_0 \cdots d_k\, x_{k+1} \cdots x_{n-1})$.

Proof: For a given pair $(s_0 \cdots s_{n-1} - d_0 \cdots d_{n-1})$, we can get (3), (4) from (1), (2).

$$S[(s_0 \cdots s_{n-1})] = (s_1 \cdots s_{n-1}\, s_0) \quad (3)$$

$$B_{k-1}[\cdots B_0[(s_1 \cdots s_{n-1}\, d_0)_0]\,] = (d_0 \cdots d_{k-1}\, s_{k+1} \cdots s_{n-1}\, s_k)_k \quad (4)$$

The input link in stage 0 after the shuffle exchange is denoted by $(s_1\, s_2 \cdots s_{n-1}\, s_0)$ in (3). The output link in stage 0, denoted by $(s_1\, s_2 \cdots s_{n-1}\, d_0)_0$ in the left hand of (4), is obtained by (3) with the routing bit d_0. The input link in stage 1 after a banyan exchange is denoted by $B_0[(s_1\, s_2 \cdots s_{n-1}\, d_0)_0]$. Therefore, k-1 iterations (each iteration including a routing in SE's and a banyan exchange between stages) result in the right hand of (4). As results, a given cell with I/O pair $(s_0 \cdots s_{n-1} - d_0 \cdots d_{n-1})$ is entered into an input link $(d_0 \cdots d_{k-1}\, s_{k+1} \cdots s_{n-1}\, s_k)_k$ of an SE $(d_0 \cdots d_{k-1}\, s_{k+1} \cdots s_{n-1})_k$ at stage k. This cell may compete with the cell from the other input link $(d_0 \cdots d_{k-1}\, s_{k+1} \cdots s_{n-1}\, \bar{s_k})_k$ for the cell transmission.

To determine the addresses of input ports which may send a cell to the other input link $(d_0 \cdots d_{k-1}\, s_{k+1} \cdots s_{n-1}\, \bar{s_k})_k$, we apply definitions to $(d_0 \cdots d_{k-1}\, s_{k+1} \cdots s_{n-1}\, \bar{s_k})_k$. In case of stage k, its output link in stage k-1 is $(d_0 \cdots d_{k-2}\, \bar{s_k}\, s_{k+1} \cdots s_{n-1}\, d_{k-1})_{k-1}$. However, this output link may receive a cell from two input ports, i.e., $(d_0 \cdots d_{k-2}\, \bar{s_k}\, s_{k+1} \cdots s_{n-1}\, 0)_{k-1}$ and $(d_0 \cdots d_{k-2}\, \bar{s_k}\, s_{k+1} \cdots s_{n-1}\, 1)_{k-1}$. We represent these input ports as $(d_0 \cdots d_{k-2}\, \bar{s_k}\, s_{k+1} \cdots s_{n-1}\, x_{k-1})_{k-1}$. Using the same methods, we can get the input ports, which can send a cell to the same SE at stage k, $(x_0 \cdots x_{k-1}\, \bar{s_k}\, s_{k+1} \cdots s_{n-1})$. To determine the addresses of output ports which may send a cell from the SE $(d_0 \cdots d_{k-1}\, s_{k+1} \cdots s_{n-1})_k$, we apply (1), (2) to an output link $(d_0 \cdots d_{k-1}\, s_{k+1} \cdots s_{n-1}\, d_k)_k$, where the routing tag of stage k is d_k. Similarly, we can get the output ports, where an SE $(d_0 \cdots d_{k-1}\, s_{k+1} \cdots s_{n-1})_k$ can send a cell, are $(d_0 \cdots d_k\, x_{k+1} \cdots x_{n-1})$. As results, we can conclude that a given I/O pair $(s_0 \cdots s_{n-1} - d_0 \cdots d_{n-1})$ always conflicts with the I/O pairs $(x_0 \cdots x_{k-1}\, \bar{s_k}\, s_{k+1} \cdots s_{n-1} - d_0 \cdots d_k\, x_{k+1} \cdots x_{n-1})$. We define these I/O pairs as the conflict pairs.

3. Conflict Cell Decision Process

To determine the conflict I/O pairs for a given I/O pair, we consider an example of 8 x 8 banyan network in Figure 3 and the related tables.

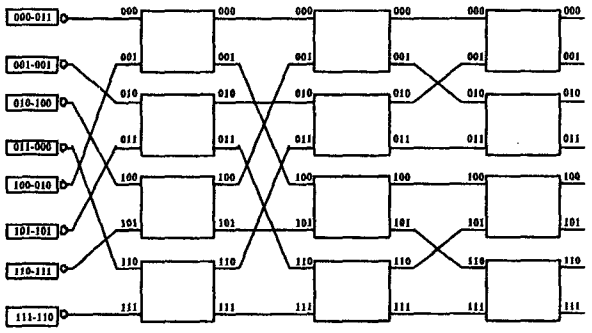

Figure 3. Example of 8 x 8 banyan network.

The table consists of lows and columns for input port addresses and output port addresses, respectively. An I/O pair of an incoming cell is represented as a circle and its conflict I/O pairs are represented as '1' in corresponding positions of input-output addresses. The rest of I/O pairs are represented as '0'. We define this table as a conflict table.In case of a given pair (000-011) at stage 0, its conflict pairs are (100-$0x_0x_1$), i.e., (100-000), (100-001),

(100-010), and (100-011). These relationships are depicted in Figure 4.

	000	001	010	011	100	101	110	111
000				○				
001								
010								
011								
100	1	1	1	1	0	0	0	0
101								
110								
111								

Figure 4. Conflict table of a given I/O pair (000-011).

From above, we can obtain a whole conflict table for stage 0 as shown in Figure 5. A notation (1) indicates a conflict cell because positions of an incoming cell and the conflict I/O cells are overlapped.

	000	001	010	011	100	101	110	111
000	1	1	1	(1)	0	0	0	0
001	0	(0)	0	0	1	1	1	1
010	0	0	0	0	(1)	1	1	1
011	(0)	0	0	0	1	1	1	1
100	1	1	(1)	1	0	0	0	0
101	1	1	1	1	0	(0)	0	0
110	0	0	0	0	1	1	1	(1)
111	1	1	1	1	0	0	(0)	0

Figure 5. Conflict table for stage 0.

On the other hand, a notation (0) indicates an incoming cell without conflict. In Figure 6, there are two conflicts at stage 0, due to four (1)'s. Figure 6 shows an example of internal conflict at stage 0 in Figure 5. There are two internal conflicts at stage 0. To eliminate these conflicts, only one cell can continue to forward to its next stage but the other cell must be suspended to forward, because each SE has a 2 x 2 I/O ports. The eliminated cell can be randomly selected for each conflict without considering the network performance.

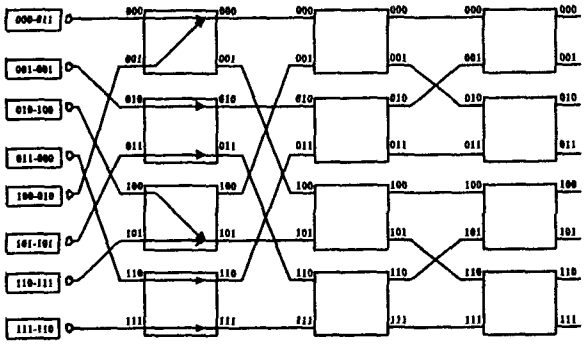

Figure 6. Examples of internal conflict at stage 0.

We assume that two cells, (100-010) and (110-111) are selected to be eliminated in Figure 6. From this assumption we re-construct a conflict table for stage 0 as shown in Figure 7. There are no conflicts in this table.

	000	001	010	011	100	101	110	111
000	0	0	0	(0)	0	0	0	0
001	0	(0)	0	0	1	1	1	1
010	0	0	0	0	(0)	0	0	0
011	(0)	0	0	0	1	1	1	1
100	1	1	1	1	0	0	0	0
101	1	1	1	1	0	(0)	0	0
110	0	0	0	0	1	1	1	1
111	1	1	1	1	0	0	(0)	0

Figure 7. Re-constructed conflict table for stage 0.

This process continues from 0 stage to n-2 stage, because the output conflict problem is out of scope in this paper. After this process, n-1 conflict tables are achieved. We name these processes as the conflict cell decision process.

4. Input Port Re-assign Process

To re-assign new input ports to conflict cells, we should utilize n-1 conflict tables. To find the conflict-free input port for each conflict cell, conflict tables should be logically ORed and we define this table as the input port re-assign table. Figure 8 shows an input port re-assign table.

	000	001	010	011	100	101	110	111
000	0	0	0	(0)	1	1	0	0
001	0	(0)	0	0	1	1	1	1
010	0	0	1	1	(0)	0	0	0
011	1	1	0	0	1	1	1	1
100	1	1	1	1	0	0	0	0
101	1	1	1	1	0	(0)	1	1
110	0	0	1	1	1	1	1	1
111	1	1	0	0	1	1	(0)	0

Figure 8. Input port re-assign table.

A

B

	000	001	010	011	100	101	110	111
000	0	0	0	(0)	1	1	0	0
001	0	(0)	0	0	1	1	1	1
010	0	0	1	1	(0)	0	0	0
011	1	1	0	0	1	1	1	1
100	1	1	1	1	0	0	0	0
101	1	1	1	1	0	(0)	1	1
110	0	0	1	1	1	1	1	1
111	1	1	0	0	1	1	(0)	0

Figure 9. Selection example of a new input port.

There are two conditions to re-assign available inputs for

these cells. The first one is that available input ports should be empty input ports. The second one is that available input ports should be chosen from I/O pairs which have a '0' value.

In the input port re-assign table, these available inputs of a given conflict cell can be detected by searching the intersections of a corresponding column of an output port of a given conflict cell and a raw of an empty input port with a '0' value. In case of an eliminated cell with a destination (000), its new input port is to be the intersection between a section A and B in Figure 9.

The section A is a corresponding column of an output port 000 and the section B is a raw of an empty input port with a '0' value. Therefore an input port of the conflict cell, which is expected to forward to an output 000, should be an input port 110.

In case of the N x N banyan network, there may be more than two intersections for a conflict cell during the input port re-assign process. In this case, the priority technique is needed. That is, the conflict cells with fewer intersections should have higher priority than the conflict cells with more intersections. This is to inhibit the conflict cells with more intersections from monopolizing the available input ports.

5. Simulation Results

Simulations are performed to evaluate the performance of the N x N banyan network with NBPG. The incoming cell is directed to any one of the output ports with an equal probability ($1/N$). The simulation result shows the maximum throughput of the banyan network with NBPG and without NBPG. In case of the banyan network with NBPG, its values are smoothly decreased from 0.656001 to 0.63367. In case of the banyan network without NBPG, its values are rapidly decreased from 0.51535 to 0.37027.

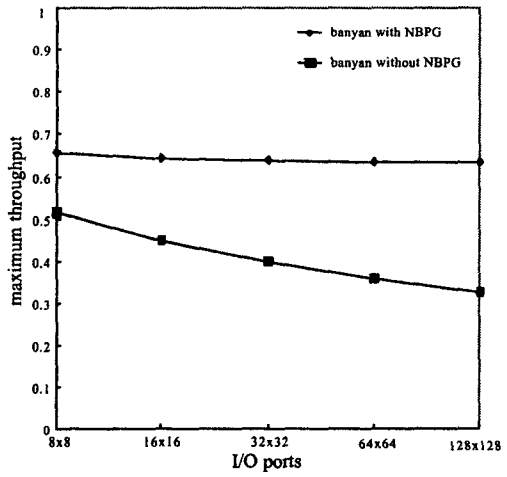

Fig 12. Maximum throughput vs. I/O ports.

From these results, we can verify that the proposed banyan network has better performance than the normal banyan network as I/O ports increases. To verify the proposed banyan network resolves the internal conflict problem of the banyan network, we define the number of success cells in the transmission as N_o and find following relationship.

$$N_t = number\ of\ successfully\ transmitted\ cells + N_o \quad (5)$$

Equation (5) means that the proposed banyan network resolves the internal conflict of the banyan network completely. As a result, the proposed NBPG generates non-blocking I/O permutations.

6. Conclusions

In this paper, we have presented NBPG to resolve the internal blocking in the banyan network. We formally provided and proved the relationship between a given I/O pair and its conflict I/O pairs. From these results, we propose two processes; the conflict cell decision process and the input port re-assign process for NBPG. In the former process, the conflict cells are detected from the incoming cells by using the conflict tables. In the latter process, new input ports are re-assigned to the conflict cells for the non-blocking transmissions. As a result of the simulation, we verify that the proposed NBPG generates the non-blocking I/O permutations completely. In case of the proposed NBPG, the processing time may be a bottleneck for the cell transmission. However, this problem is expected to be resolved by the parallelism of the processes.

Acknowledgement

This research was supported by Center of Innovative Design Optimization Technology (ERC of Korea Science and Engineering Foundation).

References

[1] Jonathan Turner, Naoaki Yamanaka, "Architectural Choices in Large Scale ATM Switches," *IEICE Trans. comm.*, vol. E81-B, no. 2, pp. 120-137, Feb., 1998.

[2] Komain Pibulyarojana, Shigetomo Kimura, Yoshihiko Ebihara, "A Study on a Hybrid Dilated Banyan Network," *IEICE Trans. comm.*, vol. E80-B, no. 1, pp. 116-126, Jan., 1997.

[3] Batcher, K. E.: 'Sorting networks and their application', *Proc. AFIPS Spring Joint Comp. Conf.*, pp.307-314, 1968.

[4] Thomas, M. C. and Stephen, S. L.: 'ATM switching systems' Artech House , Norwood, MA, 1995.

[5] Wu, C. L. and Feng, T. Y.: 'On a Class of Multistage Interconnection Networks', *IEEE Trans. Comput.*, vol. 29, pp.694-702, 1980.

[6] Bhuyan, N. L. and Agrawal, P. D.,: 'Design and Performance of Generalized Interconnection Networks', *IEEE Trans. Comput.*, pp.1081-1090, Dec., 1983.