Optimized Butterfly-Kautz Rearrangeable Nonblocking Networks*

Ibrahim A. Malik, Azman Samsudin, and Rahmat Budiarto
School of Computer Science, University Science Malaysia
11800 Penang, Malaysia
{ibraheem,azman,rahmat}@cs.usm.my

ABSTRACT

A parallel computer is a collection of interconnected nodes (CPUs or memories) and the media used to connect them. High-performance switching networks can provide the right balance between the data producing node and the communication bandwidth that connect them. Rearrangeable nonblocking network is one of the techniques used in switching networks. In this paper, we proposed Optimized Butterfly-Kautz Rearrangeable Networks (OBKN), a nonblocking network with reduction in the number of switching elements (SEs). The new reduced networks keeps all the features and properties of the original Butterfly-Kautz Networks (BKN) such as O(N $\log_2$ N) routing time, rearrangeable, and nonblocking. The only difference is that, the new Optimizes Butterfly-Kautz Networks utilizes less cross-points than the original Butterfly-Kautz Network, which will help reduce the cost of the switching network and increase network performance.

Keyword: Parallel computing, High speed and broadband networking, Rearrangeable nonblocking networks, Kautz digraph, and looping algorithm.

1. INTRODUCTION

Large-scale parallel processing system that incorporate 2^6 to 2^{16} processors working together to solve a problem represent one important approach to the design of supercomputer. These systems can provide the computational performance for tasks that require real-time response, that involve extremely complex calculations, such as weather forecasting, biomedical signal processing, speech understanding, satellite-collected image analysis, etc. A parallel computer is a collection of interconnected processors (CPUs or memories) and the media used to connect them. We often say, tightly coupled for the processors that are closed physical proximity and can communicate quickly and we call the machine parallel computer rather than a computer network.

High-performance switching networks play a main role in parallel computers by providing the right balance between the processing power of the processing units and the communication bandwidth that connect them [1,2].

The switching networks can be divided into two groups: fully connected nonblocking networks and fully connected but blocking networks. Furthermore, nonblocking networks can be classified

* The authors acknowledge the research grant provided by University Sains Malaysia, Penang that has resulted in this article

into three categories: rearrangeable nonblocking networks, wide-sense nonblocking networks, and strictly nonblocking networks [3,4]. A network is classified as rearrangeable if any idle input may be connected to any idle output providing that existing connection are allowed to be rearranged. Rearrangeable nonblocking networks are attractive in terms of its features and advantages. There are many researches that have been conducted based on rearrangeable nonblocking networks to achieve high connectivity and high performance in parallel computers with minimal hardware requirements.

This paper is based on rearrangeable nonblocking networks such as the well known Benes networks [4]. Most of the network switches were constructed by using *2* x *2* switching elements (SEs). In *N* x *N* Benes network there are 2*n* - 1 stages of *2* x *2* switching elements realizing all possible permutation, *N!*, where $n = log_2 N$. The features and advantages of rearrangeable nonblocking networks have attracted many researchers attention. The main goal of most researchers is to reduce the hardware complexity, hardware cost and to achieve optimal control algorithms. In this paper we proposed a rearrangeable nonblocking network with reduced switching elements. The reduction of the SEs is achieved by eliminating SEs which are not going to be used by the routing algorithm (looping algorithm).

The reminder of this paper describes Optimized Butterfly-Kautz Networks (OBKN). Section 2 presents Butterfly-Kautz Networks with introduction to Kautz digraph. Section 3 introduces Optimized Butterfly-Kautz Networks. Section 4 calculates hardware requirement for OBKN. In Section 5 result achieved by OBKN is given. In Section 6 OBKN utilizing only 2 x 2 SEs is discussed. The paper concludes with Section 7.

2. BUTTERFLY KAUTZ NETWORKS

2.1 Kautz MINs

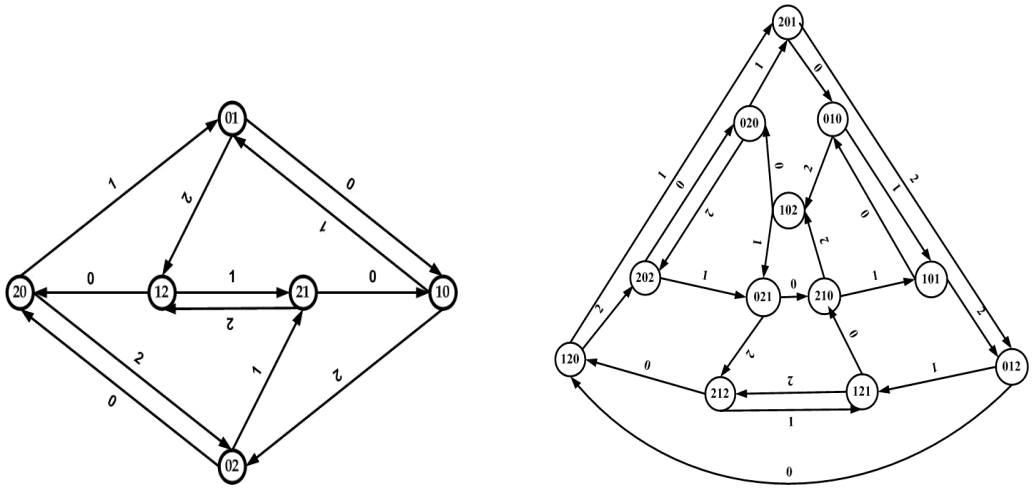
Kautz digraphs[5,6], K(*d,n*) is a regular digraph with given degree *d* and diameter *n*, K(*d,n*) has $(d^n + d^{n-1})$ nodes and $(d^{n+1} + d^n)$ edges. Figure 2.1 illustrates K(2,2),and K(2,3).

The nodes address is represented by a string of *n* (*d+1*)-ary digits with the condition that every two consecutive digits of the string are always different. There are two routing algorithms for Kautz digraphs, the shortest path algorithm and the long path algorithm. Let $X=\{ x_1 x_2 x_3 \ldots x_{n-1} x_n \}$ be the source node and $Y=\{ y_1 y_2 y_3 \ldots y_{n-1} y_n \}$ be the destination node in K(*d,n*). The long path routing from the source node to the destination node is via path:

$$
\begin{aligned}
x_1 x_2 x_3 \ldots x_{n-1} x_n &\rightarrow x_2 x_3 \ldots x_{n-1} x_n y_1 \\
&\rightarrow x_3 \ldots x_{n-1} x_n y_1 y_2 \\
&\ldots \\
&\rightarrow y_1 y_2 \ldots y_n. \qquad \text{if } x_n \neq y_1 \text{, and}
\end{aligned}
$$

$$
\begin{aligned}
x_1 x_2 x_3 \ldots x_{n-1} x_n &\rightarrow x_2 x_3 \ldots x_{n-1} x_n y_2 \\
&\rightarrow x_3 \ldots x_{n-1} x_n y_2 y_3 \\
&\ldots \\
&\rightarrow y_2 y_3 \ldots y_n. \qquad \text{if } x_n = y_1.
\end{aligned}
$$

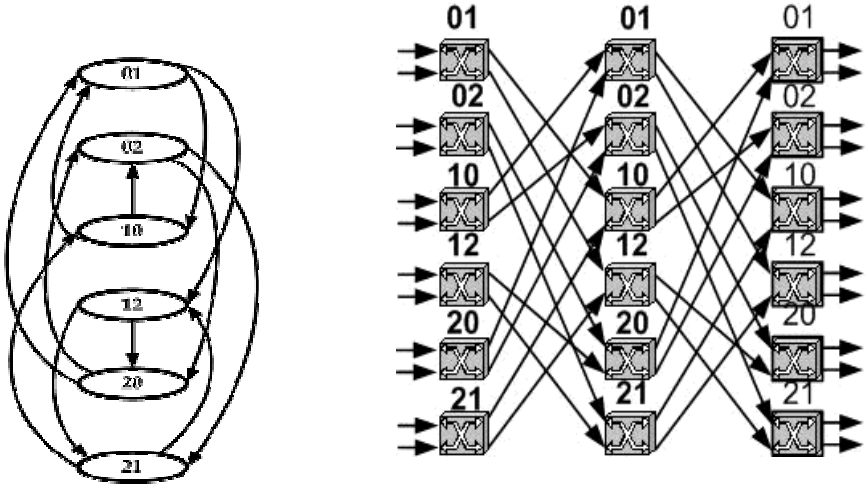
The nodes address is represented by a string of *n* (*d+1*)-ary digits with the condition that every two consecutive digits of the string are always different. There are two routing algorithms for Kautz



(a) (b)

Figure 1: Kautz digraph (a) K(2,2), (b) K(2,3).

Kautz MINs are formulated from Kautz digraphs [7,8], by redrawn with nodes in a column as shown in Figure 2(a). In Figure 2(b) a few columns of K(2,2) are drawn together. In this diagram, Kautz nodes are represented by 2 x 2 switching elements (SEs), and Kautz edges are now connect nodes from column i to nodes in column i+1. These connections represent the connection of Kautz MINs.



(a) (b)

Figure 2: (a) K(2,2), (b) K(2,3) MINs.

2.2 Butterfly Kautz Networks (BKN(n))

Butterfly Kautz Networks, BKN(n), is a rearrangeable nonblocking networks based on Kautz digraphs of degree two, with size N x N where $N=2^n + 2^{n-1}$ [8]. BKN(n) has $2n$-1 stages. All stages consist of *2* x *2* SEs except the middle stage which consists of *3* x *3* SEs. Figure 3(a) illustrates BKN(2), and its connection. The first and third stages consist of 2 x 2 SEs, the middle (2^{nd}) stage

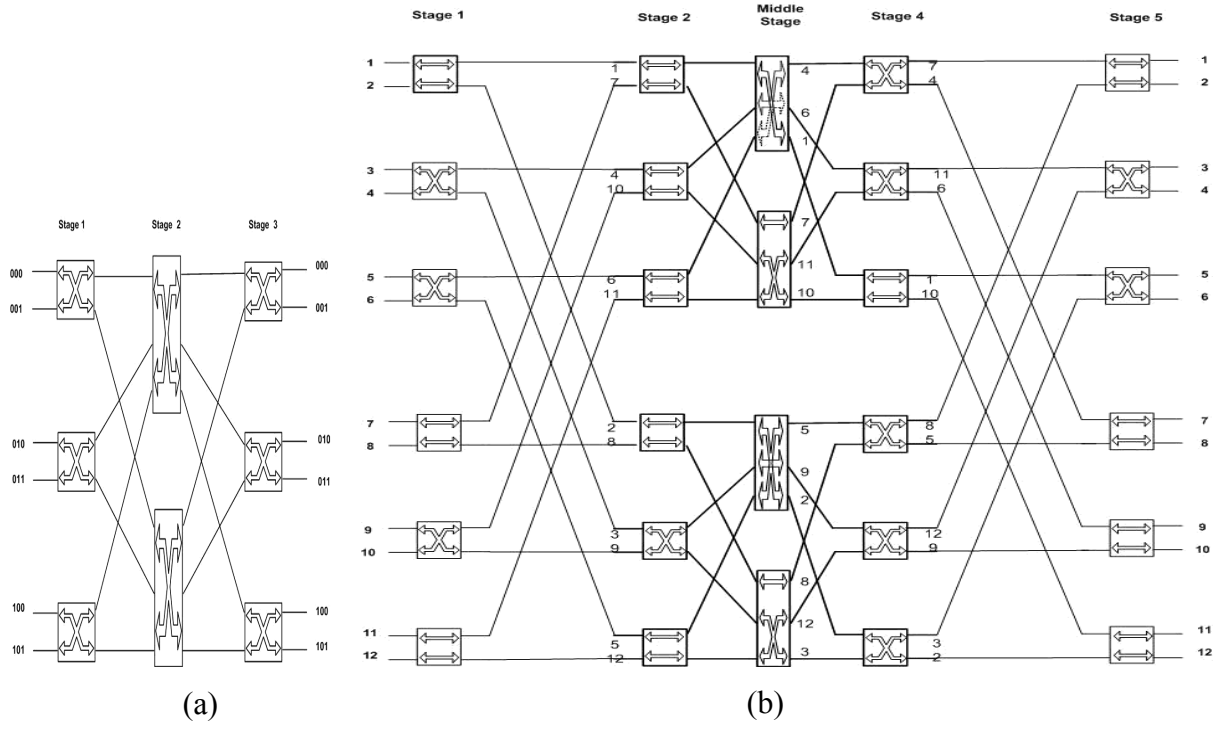
consists of 3 x 3 SEs. BKN(3) has five stages and it was constructed by stacking two BKN(2)'s as illustrated in Figure 3(b). Similarly, two BKN(n-1)'s stacked together will constitute the 2n-3 middle stage for BKN(n). The connection from stage 1 to stage 2 and from stage 2n - 2 to stage 2n - 1 will follow the Kautz digraph connection as mentioned in section 2.1.

Hardware requirement of BKN(n) are calculated by calculating the cross-points. This can be achieved by counting the cross-points used by each switching elements. Switching elements are considered as a cross-bar switches, therefore *2* x *2* SEs has four cross-points while *3* x *3* SEs has nine cross-points. The cross-points count for N x N BKN networks is $2N(2 Log_2 \frac{2N}{3} - 2) +3N$. The cross-points of Benes networks is $2N(2 \log_2 N-1)$. Routing algorithm for BKN(n) rearrange the SEs in BKN(n) similar to the looping algorithm used in Benes networks. However, a different procedure is needed to rearrange the middle stage which consists of 3 x 3 SEs. The state for the 3 x 3 SEs are determined while we are setting the state for the *2* x *2* SEs of BKN(n). If the input port of BKN(*2*) is x and the destination output port is y, then the corresponding 3 x 3 SE input port $\left\lceil \frac{x}{2} \right\rceil$ will connect to output port $\left\lceil \frac{y}{2} \right\rceil$ of the *3* x *3* SEs.

Figure 3(b) illustrates an example of the looping algorithm on the following mapping.

Let,

$$p = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 6 & 12 & 5 & 7 & 8 & 9 & 1 & 2 & 10 & 11 & 4 & 3 \end{pmatrix}$$



(a) (b)

Figure 3: Butterfly-Kautz networks (a) BKN(2), (b) BKN(3).

First, the routing algorithm start with input 1 to be routed to output 6 through the upper BKN(2). This mean the path from input 3 to output 5 has to use the lower BKN(2) (since output 5 and 6 should chose a different BKN(2)). Consequently input 4 is routed to output 7 through upper BKN(2) (since input 3 and 4 should choose a different BKN(2)). Continuing with the process, input 5 is routed to output 8 through the lower BKN(2), input 6 is routed to output 9 through the upper BKN(2), input 9 is routed to output 10 through the upper BKN(2), and input 10 is routed to output 10 through the upper BKN(2) which close the first loop. To start the second loop we may decide to route input 7 to output 1 through the upper BKN(2), and consequently route input 8 to output 2 through the lower BKN(2), which close the second loop. The third loop may route input 11 to output 4 through upper BKN(2). The third loop finished by routing the input 12 to 3 through lower BKN(2).

2.3 Other Applications of BKN Networks

Besides the usage of BKN networks in parallel computing and telecommunications, it can also be used in other applications such as Mix-net [10, 11] in cryptography; which has been used in electronic voting and card gaming [12].

3. OPTIMIZED BUTTERFLY-KAUTZ NETWORKS (OBKN(*n*))

Based on the looping algorithm discussed in the previous section, it is found that the left-most switch of BKN(*n*) can be set to either cross-state or straight-state without affecting the working of the looping algorithm. Therefore in OBKN(*n*) this switch is eliminated. Since OBKN(*n*) was constructed base on stacking two OBKN(*n-1*), the elimination of the left-most switch on both OBKN(*n-1*) can also be made. The process of elimination of the switches can continue recursively until OBKN(*2*).

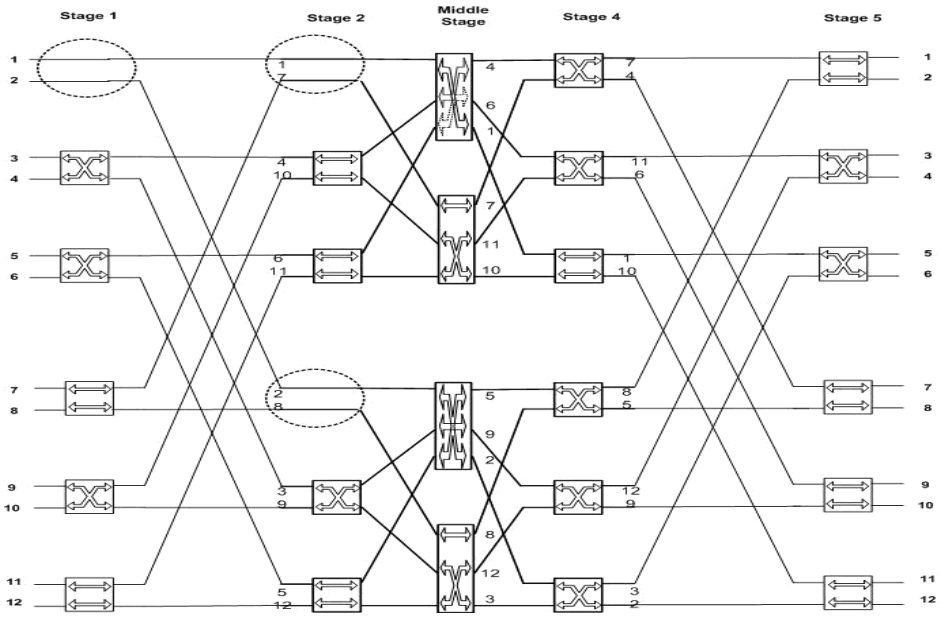


Figure 4: OBKN(3)

Here, the concern is to take the advantages on the first decision of the looping algorithm, and that by choosing the straight state of the SEs rather than the cross state. This method does not affect the networks performance, Since OBKN(*n*) is made of a stacked of two subnetworks of size OBKN(*n-1*) and the looping algorithm works independently for each one the subnetwork, therefore, both leftmost switch of each subnetwork can be eliminated and set to straight connection. This advantage can be exploited by eliminate the leftmost switch of any subnetworks recursively until we reach OBKN(*2*). Based on the mapping that mentioned in the previous section, Figures 4 shows the OBKN(*3*) routing similar to Figure 3, except the leftmost switch of OBKN(*3*) and OBKN(*2*) that depicted by circles are eliminated.

Figure 5 shows the OBKN(5) after switches elimination, the circles depicted in the Figure 5 indicates the location of the switches that have been eliminated. The number of switches being eliminated in OBKN(5) are 15 switches.

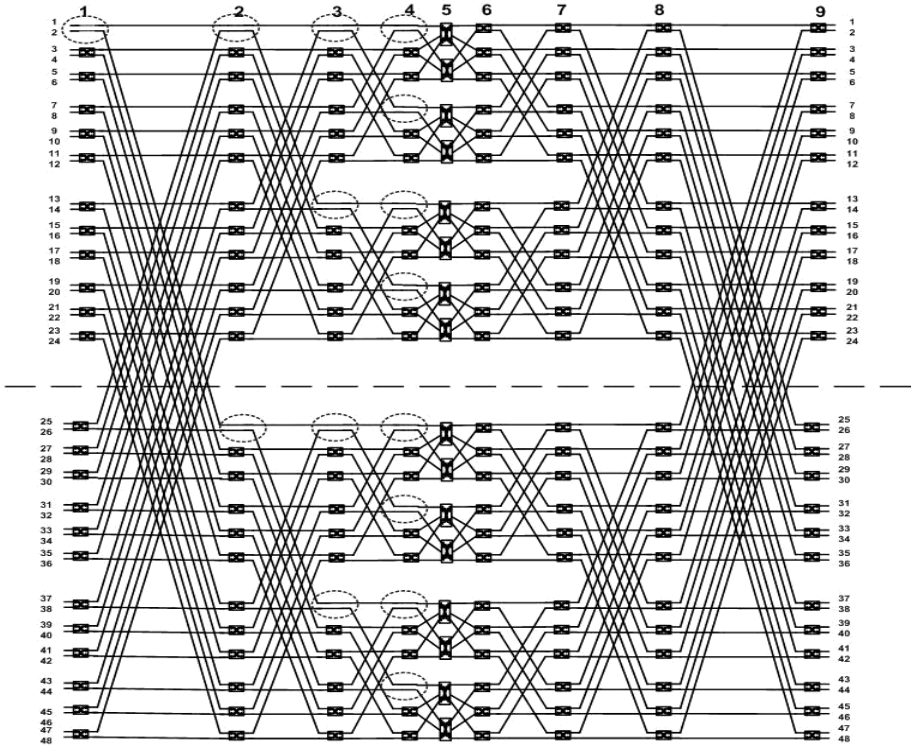


Figure 5: OBKN(5)

The recursive function given below can calculate the number of 2 x 2 switches that can be eliminated from OBKN(*n*). The function can then be summarized to a summation (1).

Function *Eliminated (n)*

Begin
If (n = 2) then ;because the smallest
OBKN(*n*) is OBKN(2)
reduce = 1
Else
reduce = 2^n + Eliminated (n-1)
return(reduce)
End

4. OBKN(*n*): HARDWARE COMPLEXITY

4.1 OBKN(*n*) Cross-Points

The number of 2 x 2 switches, S, which can be eliminated in OBKN(*n*) are:

$$S = \sum_{i=0}^{n-2} 2^i \quad (1)$$

Since we assume each 2 x 2 SE consists of 4 cross-points, the number of cross-points, *C*, eliminated in OBKN(*n*) are:

$$C = S * 4 \quad (2)$$

In general, the number of cross-points used in OBKN(*n*) are:

$$(2N(2\,Log_2 \frac{2N}{3} - 2) + 3N) - 4\sum_{i=0}^{n-2} 2^i, \quad \text{where } n = Log_2 \frac{2N}{3} \quad (3)$$

4.2 Switch Rearrangement Delay

Each switch has delay time while setting up the switch either straight or cross states. This delay time is very small, but the delay increases in rearrangeable network. According to the algorithm, the network needs to rearrange all switches before transmission. The following equations help to calculate the total delay for switches configuration.

Let, λ be single switch delay for setting up.

Total number of SEs in BKN(*n*) = $\frac{N}{2}\left(2\log_2 \frac{2N}{3} - 2\right) + \frac{N}{3}$.

The total delay of all switches in BKN(*n*) = $\frac{N}{2}\left(2\log_2 \frac{2N}{3} - 2\right) + \frac{N}{3}$ x λ.

Whereas in OBKN (*n*) = $(\frac{N}{2}\left(2\log_2 \frac{2N}{3} - 2\right) + \frac{N}{3} - \sum_{i=0}^{n-2} 2^i)$ x λ.

5. RESULTS

Table 1 shows the number of input/output ports for Benes, BKN(*n*), and OBKN(*n*) and their respective usage of cross-points. Figure 6 illustrates the number of cross-points used in Benes, BKN, and OBKN against the number of input/output ports. Figure 7 shows the time needed for switch rearrangement (switch setup either straight state or cross state).

Table 1: Approximate number of cross-points for Benes, Butterfly Kautz Networks (BKN), and Optimized BKN.

I/O port	Benes	BKN	OBKN
8	80	69	65
32	576	533	505
60	1298	1217	1157
128	3328	3156	3032
400	13030	12494	11806
1200	46698	45091	43047
1800	74259	71847	69803
2400	102997	99781	95689
3000	132609	128589	124497
3600	162918	158095	154003
4200	193808	188180	184088
4800	225193	218762	210574
5400	257013	249778	241590
6000	289218	281179	272991
6600	321770	312927	304739
7800	387793	377342	369154
8400	421215	409961	401773
9000	454886	442827	426447

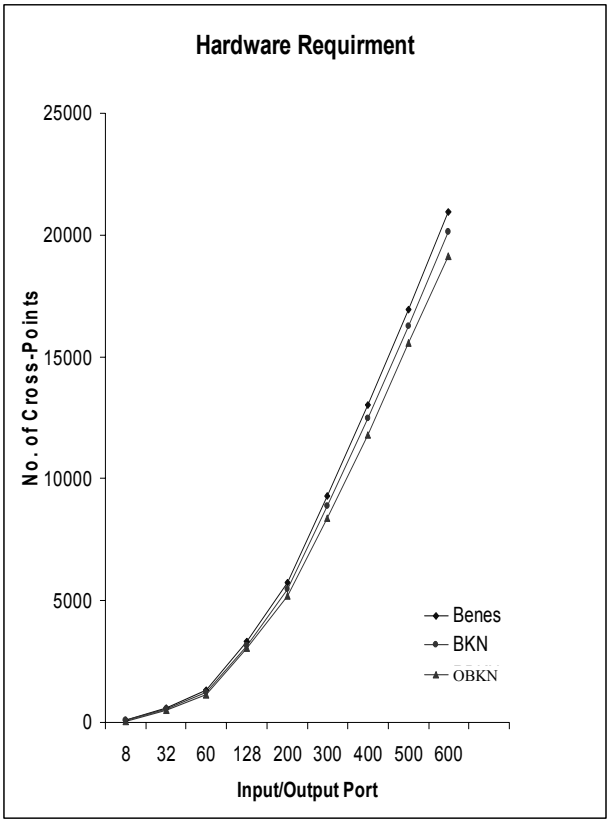


Figure 6: Cross-points usage on the three networks against input / output ports.

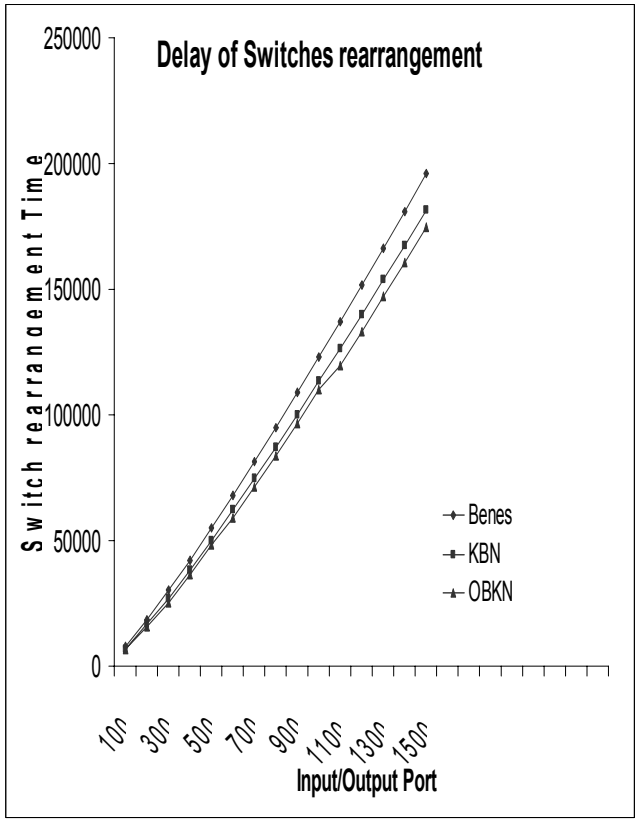
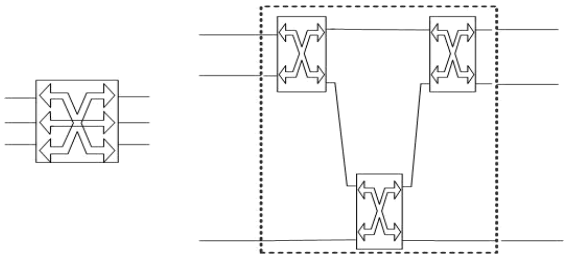


Figure 7: Total time of switch rearrangement for the three networks against the input/output ports.

6. OBKN(*n*): UTILIZING ONLY 2 x 2 SEs

As mentioned previously, there are some applications use rearrangeable networks such as electronic voting and card gaming. These applications utilize network constructed from 2 x 2 SEs. However, the middle stage of OBKN utilizes 3 x 3 SEs. We need to modify OBKN to be applicable for those applications. Therefore, each 3 x 3 SE will be replaced by three 2 x 2 switches. Figure 8(a) shows the structure of 3 x 3 SE, whereas, Figure 8(b) illustrates the connection and the structure of three 2 x 2 switches (the replacement of 3 x 3 SE).



(a) (b)

Figure 8: Switching elements (a) 3 x 3 SE, (b) the replacement of 3 x 3 SE (three switches utilizing 2 x 2 SE).

Figure 9 shows the connection of BKN(3), that is constructed from a single type of modular switching elements. Each element is a 2 x 2 SE, which can be set by the control bit into straight state or cross state. The rectangular in Figure 9 depicts the replaced 3 x 3 SE. Figure 10 illustrates the connection and the structure of OBKN(3), which is similar to BKN(3) as illustrated in Figure 9, but with the exceptional that, the leftmost switches of the network and its subnetworks are being eliminated. The algorithm used in OBKN(*n*) is the same as the looping algorithm used in BKN(*n*).

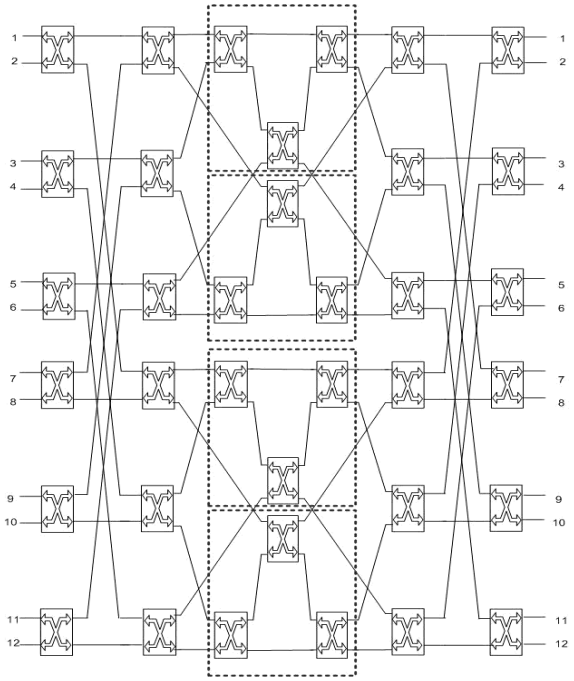


Figure 9: BKN(3) using 2 x 2 SEs

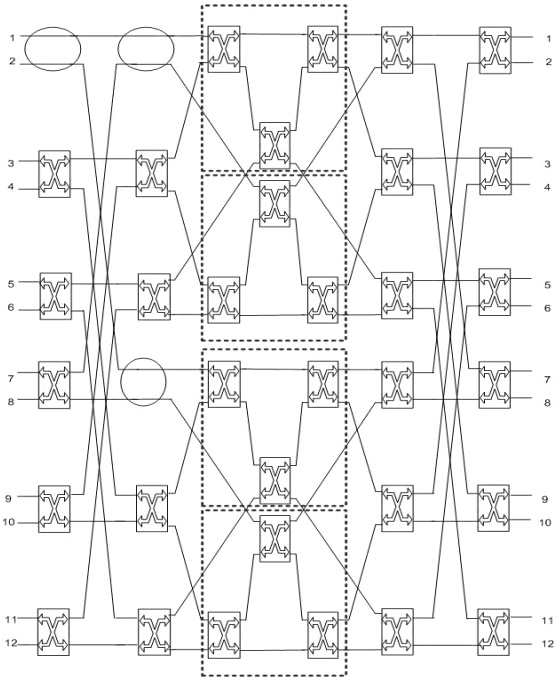


Figure 10: OBKN(3) using 2 x 2 SEs

The following equations calculate the total numbers of switches and cross-points used in OBKN network:

Total number of SE in OBKN(n)= $\frac{N}{2}\left(2\log_2\frac{2N}{3}-2\right)+N-\sum_{i=0}^{n-2}2^i$ (4)

Total number of cross-points in OBKN(n)= $(2N(2\,Log_2\frac{2N}{3}-2)+4N)-4\sum_{i=0}^{n-2}2^i$ (5)

6.1 OBKN Introduce New Sizes for Switching Network

There are two ways of evaluating and comparing between Benes and OBKN networks. From the equations above, OBKN(n) is better than Benes in term of hardware requirements. Secondly, is by the size offered by each network. The size (input/output ports) of Benes network has to be a power of 2, $N=2^n$, which n is the diameter of the network. The size (input/output ports) of OBKN network is $N=2^n+2^{n-1}$. In Benes, if the size of the needed network is not a power of 2, a larger than needed network has to be used, and many resources in the used network will remain idle. For example, assume n=10,

- N= 1024 input/output ports (Benes network).
- N= 1536 input/output ports (OBKN network).
- Total number of 2 x 2 switches =9728 SEs (Benes network).
- Total number of cross-points=38912 cross-points (Benes network).
- Total number of 2 x 2 switches =14849 SEs (OBKN network).
- Total number of cross-points=59396 cross-points (OBKN network).

If there is an application requires 1400 input/output ports, then the only number of input/output ports that can realize this requirement is 2048 I/O ports (using Benes network) which is 2^{11}, n=11, The number of resources that remain idle (in term of input/output ports) is 648 I/O ports, and in term of SEs is 7572 SEs. however the same requirement can be realized using OBKN networks without that much of hardware requirements; 1536 I/O ports, n=10. With this implementation the number of resources that remain idle (in term of input/output ports) is 136 I/O ports, and in term of SEs is 1547 SEs.

7. CONCLUSIONS

In this paper, we reduced the number of switches used in BKN(n) by removing switches which are not being used in the routing algorithm. The new network, OBKN(n), uses the same routing algorithm and has the same features as BKN(n) except less cross-points are used. Reducing the hardware in term of number of cross-points will reduce the cost of the switching network and increase switching network performance since less switches need to be configured during routing. OBKN utilizes 2 x 2 SEs can be applicable for Mix-net in cryptography. In these kinds of applications the reduction of SEs in OBKN is very useful in reducing heavy calculation associated with mixing nature of the cryptography applications.

8. REFERENCES

[1] F. Tse-Yun, "A Survey of Interconne-ction Networks," Computer, vol. 14, no.12. pp. 12-27, December 1981.

[2] Y. Raed, Y. Awdeh, and H.T. Muftah, "Survey of ATM Switch Architecture," Computer and ISDN systems, vol. 27. pp. 1567-1613.1995.

[3] V.E. Benes, "Optimal Rearrangeable Multistage connecting Networks," Bell Syst. Tech. J, vol. 43, pp 1641-1656, July 1994.

[4] V.E. Benes, "*Mathematical Theory of Connecting Networks and Telephone Traffic*," Academic Press, (1965).

[5] W. H. Kautz, "Design of Optimal Interconnection Networks for Multiprocessors," Architecture and Design of Digital Computers, NATO Advanced Summer Institute, pp. 249-272, 1969.

[6] P. Tvrdik, "Factoring and Scaling Kautz Digraphs," Research report 1398, LIP ENSL, 96364, LYON, France, May 1994.

[7] I. A. Malik and A. Samsudin, "Butterfly-Kautz Networks: New Rearrangeable Nonblocking Multistage Interconnection Networks," 5th World Multiconference On Systemics, Cybernetics, and Informatics – SCI 2001, vol. V, pp. 251-255, July 2001.

[8] I. A. Malik, A. Shakir, and A. Samsudin, "Multiple Kautz Interconnection networks with Parallel Loading,"4th World Multiconference On Systemics, Cybernetics, and Informatics – SCI 2000, vol. IV, pp. 267-271, July 2000.

[9] A. Shakir and A. Samsudin, "Switching Networks with Deflection Routing Based on Kautz Digraph, " International Conference on Communication in Computing 2000 – CIC 2000, pp. 259-263, Jun. 2000.

[10] M. Abe, "Mix-Networks on Permutation Networks", Asiacrypt 99, pp. 258-273, 1999, LNCS 1716.

[11] M. Abe and F. Hoshino. Remarks on Mix-Network Based on Permutation Networks. *Proceedings 4th International Workshop on Practice and Theory in Public Key Cryptography PKC 2001*, Lecture Notes in Computer Science, pages 317-324, Springer-Verlag, 2001.

[12] Koutarou Suzuki , " Permutation Networks With Arbitrary Number of Input and its Appilcation to Mix-net", IEICE Trans. A, Vol. E00-A, No. 1 January 2002.