25397

Multiplication and Exponentiation of Big Integers with Hybrid Montgomery and Distributed Karatsuba Algorithm[1]

Farij Omar Ehtiba and Azman Samsudin
School of Computer Science, Universiti Sains Malaysia
11800 Penang, Malaysia
{com20279, azman}@ cs.usm.my

Abstract

The public-key encryption scheme is vulnerable to a brute-force attack if the key size used is small. Thus, the key size must be large enough to make brute-force attack impractical. In practice, the key size that makes brute-force attack impractical will also make the process of encryption and decryption slow. There are several modular exponentiation algorithms that can be used for speeding up modular based public-key algorithms. One of the algorithms is Montgomery algorithm. However, using Montgomery algorithm for public-key purposes requires multiplication of large integers which runs very slow. Therefore, multiplication algorithms must be optimized in order to achieve acceptable performance. Karatsuba algorithm is one of the possible algorithms which can be used to perform multiplication of large numbers. In this paper, we will enhance Montgomery algorithm with distributed Karatsuba algorithm running on cluster of personal computers to decrease the running time of modular exponentiation that is required by the modular based public-key algorithms.

Keywords: *Public-Key Cryptography, Distributed Computing, Montgomery Algorithm, and Karatsuba Algorithm*

1. Introduction

The motivation for studying high-speed and space-efficient algorithms for modular multiplication comes from their applications in public-key cryptography such as RSA algorithm [1] and the Diffie-Hellman key exchange scheme [2], which are required fast hardware and software implementations of the arithmetic operations in $GF(2^k)$ for large values of k. There are many modular exponentiation algorithms available, and one of them is Montgomery algorithm [3]. Montgomery reduction algorithm is a technique that allows efficient implementation of modular multiplication without explicitly carrying out the classical modular reduction step.

Montgomery multiplication avoids a division at the expense of two multiplications, with overhead of the initial and final conversions to and from Montgomery representation. For single multiplication the Montgomery algorithm is disadvantageous. The advantages become apparent when we perform exponentiation. Therefore, Montgomery algorithm will be suitable to compute calculation such as M^k mod N, which normally appear in modular arithmetic based public-key cryptography. However, using Montgomery algorithm for public-key purposes requires multiplication of large integers which runs very slow. Therefore, multiplication algorithms must be optimized in order to achieve acceptable performance.

Karatsuba algorithm [4] is one of the possible algorithms which can be used to perform multiplication of large integers in fewer operations than the usual brute-force technique of long multiplication. It was invented by Karatsuba and Ofman [5]. In Karatsuba multiplication, to multiply two n-digit integers x and y, both of the numbers need to be divided into left and right halves (x_1, x_2 and y_1, y_2) and the halves are then use as shown in the following formula:

$$\begin{aligned} x \times y = {} & (x_2 \times y_2) \times R^2 \\ & + (x_2 - x_1) + (y_1 - y_2) \times R \\ & + (x_1 \times y_1) \times (R+1), \qquad (1) \end{aligned}$$

where R = *the size of the halves.*

The value of R is choose to be half the width of the numbers being multiplied, which simplify the process of multiplications by R to just a shifting operation [7]. Since the numbers need to be multiplied need to be halved (into 'left' and 'right' parts) the algorithm works best when the lengths of the numbers are the same size and are powers of two [8].

Karatsuba's performance gains lies on the fact that the multiplication process can be done with only three rather

[1] The authors acknowledge the research grant provided by Universiti Sains Malaysia that has resulted in this article.

than four multiplications. In normal method, we need four multiplications which are $(x_1 \times y_1)$, $(x_1 \times y_2)$, $(x_2 \times y_1)$ and $(x_2 \times y_2)$. In Karatsuba, these can be reduced to three multiplications:

$$(x_1 \times y_1), \qquad (A)$$

$$(x_2 \times y_2), \qquad (B)$$

$$(x_1 + x_2) \times (y_1 + y_2), \qquad (C)$$

where $(C - A - B) = (x_1 \times y_2) + (x_2 \times y_1)$.

Karatsuba's multiplication is not very fast for small values of n [8].

The distributed computation of Karatsuba can be done by performing three multiplications, (A), (B) and (C) in parallel on different nodes. The operations can be performed as (C) within the current task, while (A) and (B) are performed by two other parallel tasks.

Karatsuba's algorithm has been employed to multiply two integers of 2^n digits in [4]. Karatsuba's multiplication is recursively applied until 2^k digits, and then the grade-school multiplication is used to compute the product of 2^k-digit numbers. The experimental shows Karatsuba's multiplication is much more efficient than the grade-school multiplication and there exist a cutoff value 2^k that gives the best performance. The authors also observe that Karatsuba's algorithm gives the best performance when the length of the cutoff number is 16 digits, which is independent to the problem size.

Liu, *et al.* [6], have used digit numbers in order to evaluate Karatsuba Algorithm, and their results illustrated that the parallel Karatsuba algorithm is more efficient at number's length of approximately 256 digits.

2. Methodology and New Design

Public key cryptography is designed to provide many cryptographic solutions such as key management, user authentication and digital signature. Therefore the operating speed of modular multiplication, the essential operation for modular arithmetic based public key cryptography, has to be improved.

The goals are to design a speed efficient architecture and good performance of modular exponentiation with distributed Karatsuba algorithm for software implementation. The normal way to compute the modular exponentiation is by decompose the calculation into many modular multiplications. This can be done by using the efficient Montgomery algorithm. Multiplication of two very long digits in Montgomery algorithm usually takes time to compute. Distributed Karatsuba is then proposed to reduce the time of multiplication of these two very large numbers. In our design, distributed Karatsuba is chosen to perform the multiplication operations of big numbers only if the numbers being multiplied is more than 256 digits.

Modular exponentiation is performed by repeated modular multiplication. In this algorithm, the multiplication operations must be performed sequentially and the order of operation must be implemented in sequential order. However, the multiplication operations in Montgomery algorithm are being enhanced by distributed Karatsuba algorithm. The steps of multiplying two very big numbers are as following:

- The source (Server) package the two numbers and then sends them to the destinations (Clients).
- Each client unpacked the numbers, partition the numbers into four parts (as mentioned in Karatsuba algorithm earlier), and then calculates the intermediate result as shown in Figure 1.

  Client 1 – $(x_2 \times y_2) \times R^2$

  Client 2 – $(x_2 - x_1) + (y_1 - y_2) \times R$

  Client 3 – $(x_1 \times y_1) \times (R+1)$

  Then, each client returns the result back to the Server.

- The Server adds these numbers together, before continue with the next operation in the algorithm.

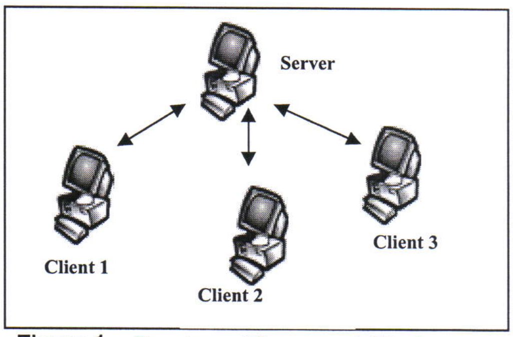

Figure 1. Typology of Server and Client

3. Implementation

Four machines (Pentium 4 with 2.4GHz processor) supported by windows XP operation system have been used to implement the proposed method. They communicated together by socket programming. The code were developed using Visual Basic version 6.0 and chip8086XT library to assist in the performing of mathematical operation for very large numbers. This library serves to overcome any overflow problem that might occur with such mathematical operations.

A socket provides an endpoint for communication between the various processors. Connection between the server and the clients was maintained via the use of WinSocket library, which is provided by Microsoft™, within a Visual Basic package. The communication is established by transmitting messages between sockets done by the processors. Any processor may make use of multiple ports to receive messages, but a processor cannot share ports with other processors on the same computer. The obvious benefits of using socket programming are:

- High performance.
- Easy to program.
- Provides primitive support for communication.

3.1 The problems during the implementation

As mentioned earlier, the socket programming under VB6 has been used to connect the machines together. There are many disadvantages to used Winsock library such as:

- It does not support synchronization between the clients and server.
- The buffer size is limited.
- VB6 does not have a function to convert a value from decimal representation form to binary representation form.
- VB6 does not support big number calculation.

These problems were solved as follow: First, the synchronization problem was solved using infinite loop with "Doevents" command to make the server waiting for all results to arrive. With the same idea the buffer size problem was also solved. This solution has a drawback that sometimes it takes extra loop which will increase the execution time. The rest of the problem was solved coding.

4. Data Analysis and Discussion

Before distributed Karatsuba can be used in Montgomery algorithm, it is important to find the critical length number when distributed Karatsuba performs better than sequential Karatsuba. This can be done by testing the efficient execution time between sequential and distributed Karatsuba. The critical length number was found to be 256 digits as shown in Figure 2, in agreement with the above authors [6].

Experiment was done and various data are collected from the implementation of Montgomery algorithm, using both sequential and parallel Karatsuba algorithms.

Referring to the original algorithm, the aim of the analysis of the collected data, which is obtained by the implementation of the algorithm in sequential mode and with parallel Karatsuba algorithm, is to use these tow modes optimally. Different input parameters were used by the two implementations, such as the number, the exponent, and the modulo (M^E mod N). Our implementations have been tested with the argument length M (8 and 16 digits), exponent E (64, 128, 256, 512, and 1024), and modulo N (100, 200, 300, 400, 500 and 600 digits).

The first input parameters are obtained by fixing the length M = 8 digits and modulo E = 64, and ranging the value of exponent, N from 100 to 600 in steps of 100. The results are displayed on Figure 3. It can be observed that the execution time at low N (100 and 200) using the sequential mode is more efficient than the parallel mode. At value of N = 300, namely the critical point, a reverse behaviour starts to appear which clearly reflects that the Montgomery algorithm supported by distributed Karatsuba algorithm is better than the algorithm in sequential mode. Above this critical point, $N > 256$, the distributed mode is clearly the more efficient mode.

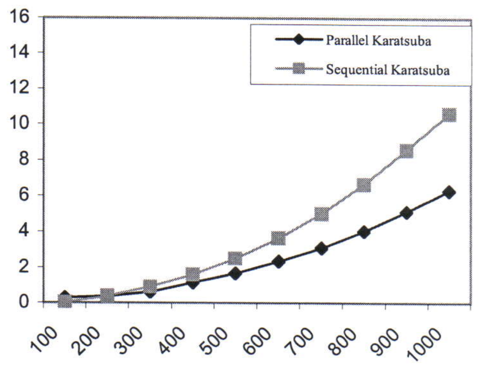

Figure 2. Comparison Between Sequential and Parallel Karatsuba Algorithm

A similar result is also obtained when using E = 128, 256, 512 and 1024. Figure 4, shown the results that obtained when the exponent E = 128.

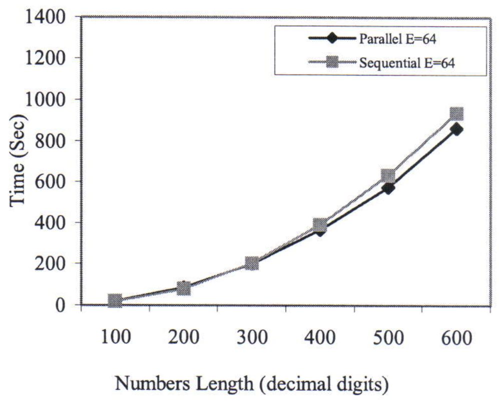

Figure 3. Result of Montgomery Algorithm (*E=64*)

There are two results which imply that the communication time cost is greater then the multiplication cost. As the exponent value increases, the length of number will also increase lending to increase in the multiplication cost. At the exponent value of E = 256 and above, the Montgomery algorithm enhanced by distributed Karatsuba gives the best performance and the communication cost becomes less than the multiplication cost.

When the input parameter M is changed to 16 (M = 16 digits) and N being the same as pervious case (N = 100, 200, 300, 400, 500, and 600 decimal digits), the performance time is also increases. However, as before at the exponent value of 256 and above, our enhanced algorithm gives the best performance. Figure 5 shown the result when the input parameter M =16, N ranged from 100 to 600 in steps of 100, and the exponent E = 128.

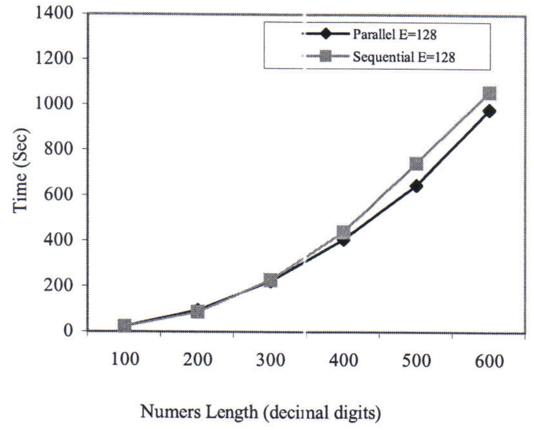

Figure 4. Result of Montgomery Algorithm (*E=128*)

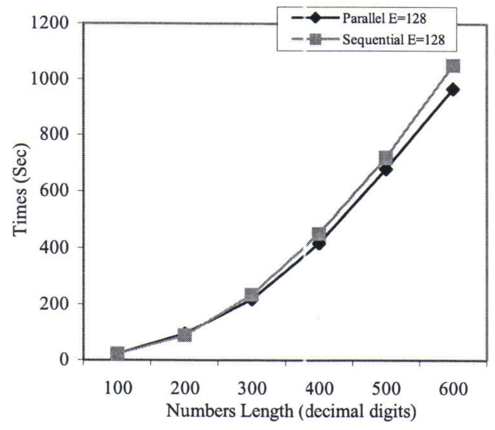

Figure 5. Result of Montgomery Algorithm (*E=128*)

5. Conclusion

It has been observed that Montgomery Algorithm with distributed Karatsuba can run faster on distributed computers when the numbers that being multiplied is more than 300 digits in length each. Similar behavior of the results was noticed at the same previous input parameters when the input argument has changed from M=8 to M=16, the only difference between the two results were that the large execution time was obtained at M=16 compare to M=8.

6. References

[1] R. L. Rivest, A. Shamir, and L. Adleman. "*A Method for obtaining digital signatures and public-key cryptosystems*". Comm. ACM, 21:120 126, 1978.

[2] Daniel. M. Gordon, *A Survey of Fast Exponentiation Methods*, Journal of Algorithms 27, pg 129–146 (1998).

[3] Peter L. Montgomery. "*Modular multiplication without trial division*". Mathematics of Computation, 44(170): 519-521, April 1985.

[4] Tudor Jebelean, *Using the Parallel Karatsuba Algorithm for long Integer Multiplication and Division*, European Conference on Parallel Processing, 1997.

[5] A. Karatsuba and Y. Ofman. "Multiplication of multidigit numbers by automata", Soviet Physics-Doklady, 7:595-596, 1963.

[6] Chin-Bou Liu, Chua-Huang Huang, and Chin-Laung Lei , "Design and Implementation of Long-Digit Karatsuba's Multiplication Algorithm Using Tensor Product Formulation " , The Ninth Workshop on Compiler Techniques for High-Performance Computing, 2003, pages 23-25.

[7] *Parallel Karatsuba Algorithm at* http://careybloodworth.tripod.com/karatsuba.htm.

[8] http://ece.gmu.edu/crypto/student_projects/projects99/suneela/suneela_report.htm

25395

Optimized Arbitrary Size Networks[1]

Zaidi Jumandi, Azman Samsudin and Rahmat Budiarto
School of Computer Science, University Science Malaysia
11800 Penang, Malaysia
{zaidi, azman, rahmat}@cs.usm.my

Abstract

A rearrangeable nonblocking network of Benes network is well known for its realization for any arbitrary permutation. Later, the introduction of Arbitrary Size (AS) Benes network which improving Benes network in term of the size of network, hence giving a better performance by performing a Benes network to arbitrary size. By reducing the number of switching elements in the network, the performance and utilization of switching elements (SEs) can be increased and optimized. In this paper, we introduce an Optimized AS-Benes (OAS-Benes) that comes with an enhancement by reducing the number of SEs based on the previous model of AS-Benes.

***Keyword**: Interconnection Network, Rearrangeable Nonblocking Network, and Benes Network.*

1. Introduction

A nonblocking network is a switching network that can connect an unused input by a path through unused edges to any unused output, regardless of which inputs and outputs have been already connected. In addition, it is rearrangeable if, given any set of disjoint paths between some inputs and outputs; every unused input can be connected by a path through unused edges to any unused output with possible rearrangements of the previously established paths [3].

The Benes network is rearrangeable nonblocking network which can realize any arbitrary permutation. The Benes network of dimension n is shown to be strictly nonblocking if only a suitable chosen fraction of $1/n$ of inputs and outputs is used. In Benes network, the r-dimensional Benes network connects 2^r inputs to 2^r outputs through $2r-1$ levels of 2 X 2 switches. Here, each level of switches consists of 2^{r-1} switches, and therefore the size of the network has to be a power of two.

This paper is based on rearrangeable nonblocking networks. In order to provide a better computational performance especially for tasks that require real – time response, a high performance switching networks is needed. This can be achieving through a good utilization of processors and a good control algorithms. Here, we proposed a rearrangeable nonblocking network with reduced switching elements. The reduction of the SEs is achieved by eliminating SEs which is not going to be used by the routing algorithm.

2. Arbitrary Size (AS) Benes Network

AS - Benes network is an extended of Benes model with arbitrary size. It is constructed from 2 X 2 switches with the internal state being determined by a binary control signal $b = \{0,1\}$ [2]. Each element of switches can be set by a control line into a direct-connection state or a crossed-connection state as shown in Figure 1, thus realizing all permutations from two inputs to two outputs [3].

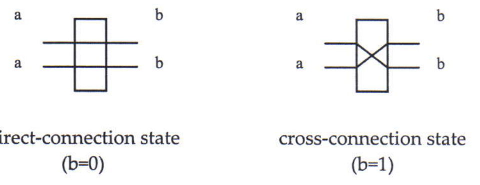

Figure 1. Two State of 2 x 2 Switch

Here, by introducing a 3 X 3 AS-Benes network, it can be generalized to recursively construct a network of any size. To construct a 3 X 3 permutation, it consist three 2 X 2 switches. By considering a simple wire to be a network that can realize any 1 X 1 permutation, we can view the 3 X 3 network in Figure 2 as being built from a 2 X 2 network and 1 X 1 network [1].

[1] The authors acknowledge the research grant provided by Universiti Sains Malaysia that has resulted in this article.

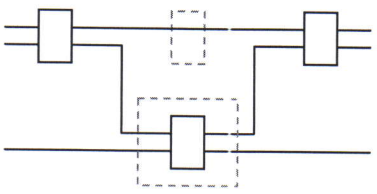

Figure 2. A 3 x 3 AS-Benes Network

Specifically, an AS-Benes of size n is constructed recursively from an upper sub-network AS-Benes of size $\lfloor n/2 \rfloor$ and lower sub-network AS-Benes of size $\lceil n/2 \rceil$. When n is even the construction is similar to that of the Benes network. Meanwhile, for odd size, the first $n-1$ output are connected to upper sub-network of size $\lfloor n/2 \rfloor$ switches and each switch is connected to the two smaller ($\lfloor n/4 \rfloor \times \lfloor n/4 \rfloor$) AS-Benes networks. The last input and the last output are connected directly to the lower sub-network AS-Benes of size $\lceil n/2 \rceil$.

2.1 Routing Algorithm

A connection between outputs and inputs may be established through either the upper AS-Benes sub-network (of size $\lfloor n/2 \rfloor$) or the lower AS-Benes sub-network (of size $\lceil n/2 \rceil$). Given that each switch at the first and last levels in an AS-Benes has precisely one connection to each of the upper and lower sub-networks, the realization of any given permutation, $\prod$, in an AS-Benes should satisfy the property that paths sharing any switch at the first or last levels must go to different sub-networks. Using this property, it can be shows that, given any one-to-one mapping, $\prod$, of n inputs to n outputs, there is as set of edge disjoint paths from the inputs of a size n AS-Benes to its outputs connecting input i to output $\prod(i)$ for $0 \leq i \leq n-1$ [1].

As an example, we illustrate the paths in Figure 3 for the mapping

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 7 & 4 & 8 & 6 & 2 & 1 & 0 & 3 & 5 \end{bmatrix}$$

in a 9 X 9 AS-Benes network. The bold paths represent the first loop which starts at input $n-1$ and terminate output $n-1$. After this loop, there are only two pairs of input/output left which form a second loop. In this way, all paths can be assigned to the upper or lower sub-networks without any conflict.

2.2 Applications of Benes Networks

The Benes network has been proposed for use in telephone networks [4]. It and other closely related networks were also used as interconnection networks for parallel computers. It has also been suggested for use in ATM switches [5]

3 Optimized AS-Benes Networks

From the looping algorithm introduced before, a connection between inputs and outputs can be constructed through either the upper or lower AS-Benes network. This can happen since, to construct any size n×n AS-Benes network, recursively a network of size 3 can be used for an odd size while for an even size, it's based an original Benes network.

Here the left-most of the network can be eliminated by taking an advantage of the straight state of the SEs rather than the cross-state. This method can be accepted since the network is constructed from two sub network (upper and lower) and each sub network is based on its own routing algorithm. From the routing algorithm perspective, each sub-network has an independent algorithm and did not affect each other; therefore, the upper leftmost switch of even sub-network can be eliminated. This approach will continue until we reach the smallest size of AS-Benes, AS- Benes of size 3×3 or AS-Benes of size 2×2.. Figure 4 shows a 9 X 9 Optimized AS-Benes similar to Figure 3, except the leftmost switches depicted by circles are eliminated.

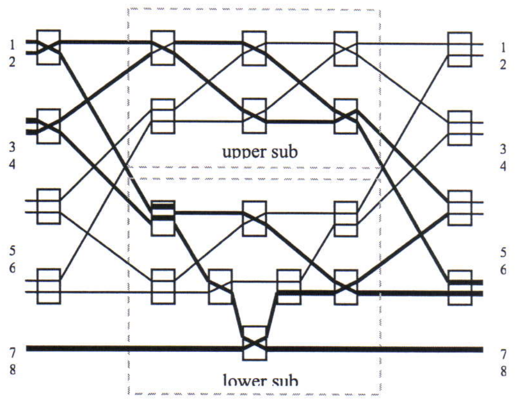

Figure 3. Two Loops In the Realization of a Permutation in a 9 x 9 AS-Benes

Figure 5 shows the OAS-Benes for all upper leftmost switches being eliminated. Circles depicted in the Figure 5 indicate the location of the switches that are being eliminated. The numbers of switches being eliminated in OAS-Benes of size 17 are 5 switches.

The recursive function was developed to calculate the number of switches that will be used for each node after the elimination in each sub-network. This recursive function can be used to compare the number of switches needed in an $n \times n$ AS-Benes. This comparison is shown in Figure 6 for n up to 32.

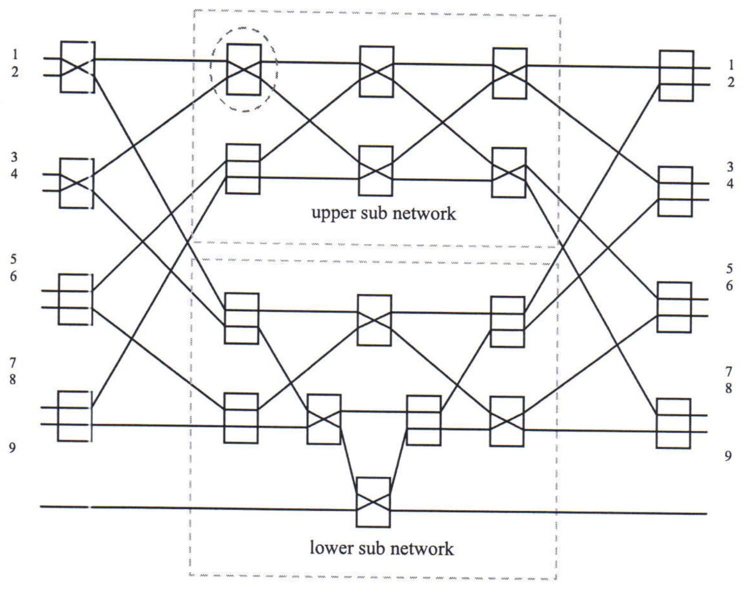

Figure 4. A 9 x 9 Optimized AS-Benes

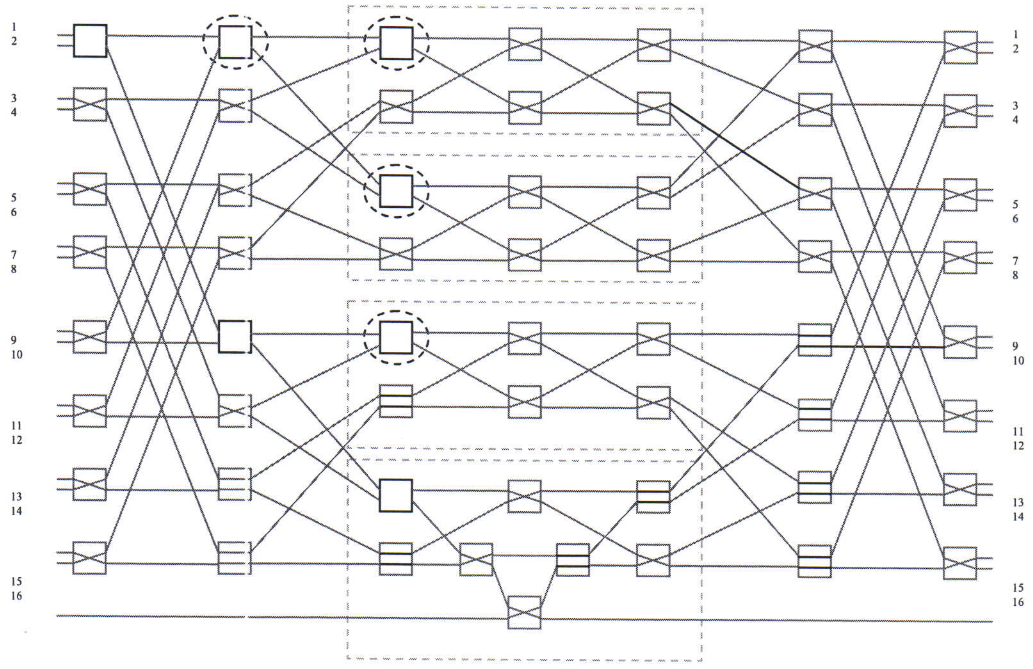

Figure 5. A 17 x 17 Optimized AS-Benes

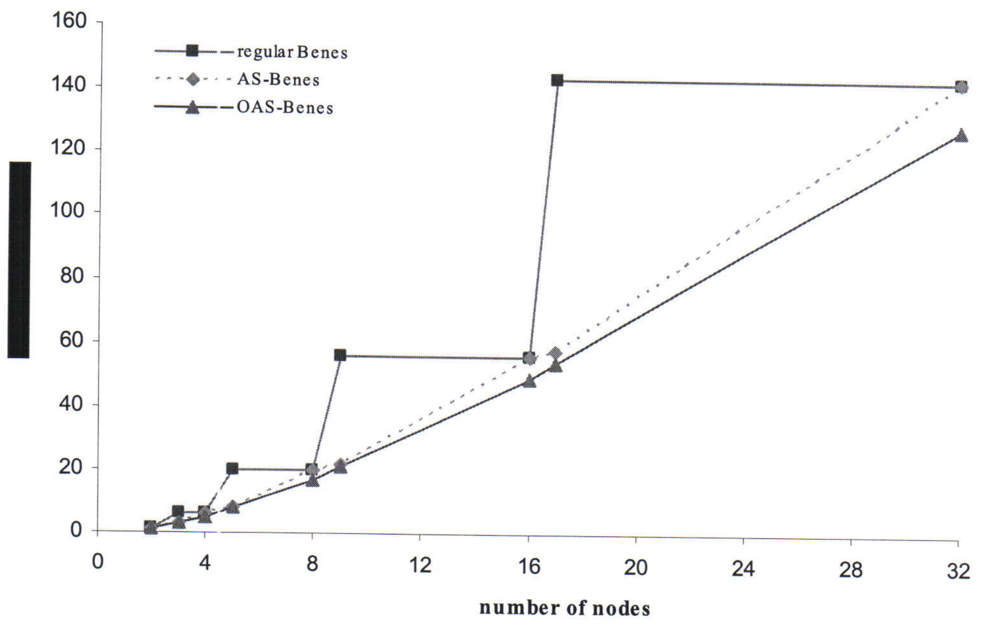

Figure 6. Comparison of Permutation Networks

4 Conclusions

By reducing the number of Switching Elements used in this design, the cost of switches can be cut and performance can be increase simultaneously. In term of a large networks or a huge usage of Switching Elements, the number of reduced Switching Elements can be significant.

5 References

[1] C. Chang and R. Melham, "Arbitrary Size Benes Networks," *Parallel Processing Letters*, vol. 7, no. 3 pp. 279-284, 1997.

[2] W. H. Soo, A. Samsudin, and A. Goh, "Efficient Mental Card Shuffling via Optimised Arbitrary-Sized Benes Permutation Network," *Information Security Conference 2002*, LNCS 2433, pp. 446-458, Sau Paulo, Brazil, Sep. 2002.

[3] P. Kolman., "On Nonblocking Properties of the Benes Network".

[4] L. A Bassalygo and M. S. Pinsker, "Complexity of An Optimum Nonblocking Switching Network Without Reconnections," *Problems of Information Transmission*, 9:64-66, 1974.

[5] J. Turner and N. Yamanaka, "Architectural Choices In Large Scale ATM Switches," *Technical Report WUCS-97-21*, Department of Computer Science, Washington University Saint Louis, 1997.