Ph.D Thesis

A study on improving the performance of three-stage Clos networks

Labson Koloko

Graduate School of Engineering Science,

Department of Mathematical Science and
Electrical-Electronic-Computer Engineering,

Akita University

This dissertation is submitted in partial fulfilment for the award of the degree of Doctor of Engineering in Electrical and Electronic Engineering

September 2021

Dedication

This thesis is dedicated to my mum whose prayers and faith inspires me to move to higher heights. To my late Dad who taught me to always work hard and be resilient in life.

Personal quote

"Never let disappointment make you drop God's promises for your life"

Declaration

This dissertation is the result of my own work and does not include anything which is the outcome of work done in collaboration except where specified within the text. It has not been submitted in part or in whole to any university or institution for any degree or qualification.

Signed .....

Date .....

2021
Akita University.

List of Publications

This dissertation is based on my published international journal and conference papers.

International Journal papers

1. L. Koloko, T. Matsumoto and H. Obara, "Design and implementation of fast and hardware-efficient parallel processing elements to set full and partial permutations in Beneš networks," *IET Journal of Engineering*, 2021.4, Doi: 10.1049/tje2.12037.

2. L. Koloko,H. Obara and S.Kumagai, "Structure and non-blocking properties of bidirectional unfolded two-stage switches," *Electronics Letters*, 2021.6, doi:10.1049/ell2.12258.

3. L. Koloko, 小原仁, " 3段Clos 網のルーチング制御の高速化に向けたFPGAによる非同期並列処理回路の試作," 電子情報通信学会論文誌B, Vol. J104–B, No. 7, pp. 598–602, 2021,doi: 10.14923/transcomj.2020BLL0002

4. L. Koloko and H. Obara, "Applying bidirectional crossbar switches with extra sets of inlets and outlets to three-stage Clos networks," *International Journal of Energy, Information and Communications*, Accepted under publication.

International Conference Papers

1. L. Koloko, N. Takanohashi, Y. Katoh, P. Selin and H. Obara, "Network utilization monitoring technique using probe packet delay variance," *Proceedings of International Conference on Electrical Engineering*, G5-0930, Seoul, Korea, June 24-28, 2018.

2. L. Koloko and H. Obara, "Nonblocking Properties of three-stage Clos Networks Composed of crossbar switch elements with an Extra set of Inputs and Outputs," *The 9th International Conference on Electronics, Communications and Networks (CECNet 2019)*, CNT2476, Kitakyushu, Japan.Oct. 18-21,2019.

3. L. Koloko and H. Obara, "Nonblocking Properties of 3-stage Clos networks comprised of unidirectional crossbars," *Proceedings of 25th OptoElectronics and Communications Conference (OECC2020)*, pp.1-3 (doi:10.1109/OECC48412.2020.9273703), Taipei, Taiwan, October 4-8, 2020.

4. L. Koloko, T. Matsumoto and H. Obara, "FPGA implementation of parallel routing algorithm for three-stage Clos networks with component switch sizes of a power of two," *Proceedings of International Conference on Emerging Technologies for Communications*, D2-1, Tokyo, Japan (Virtual Conference),December 2-4, 2020

紀要（査読あり）

1.小原　仁，Koloko Labson, “3段Closスイッチ網の性能改善に関する研究（その1）－研究の背景とスコープ－,” 秋田大学大学院理工学研究科研究報告，第39号，pp.9-14，2018

国内大会講演　研究会（査読なし）

2. 中川原 敬佑, Labson Koloko, 加藤 陽介, 小原 仁, “3段Closスイッチ網の性能改善に関する研究（その2）－空きポートを利用した双方向クロスバースイッチの構成－,” 電気関係学会東北支部連合大会講演論文集，p.112，2018

3. 柳 晋登, Labson Koloko, 加藤 陽介, 小原 仁, “3段Closスイッチ網の性能改善に関する研究（その3）－FPGAを用いたハードウェア処理による3段Clos網の高速制御－,” 電気関係学会東北支部連合大会講演論文集，p.112， 2018.

4. Labson Koloko, 橋本 拓大, 加藤 陽介, 小原 仁, “Improving Performances of Three-Stage Clos Networks, Part Ⅳ,” 電気関係学会東北支部連合大会講演論文集，p.119，2018.

5. 渋谷 悠悟, Labson Koloko, 加藤 陽介, 小原 仁, “3段Closスイッチ網の性能改善に関する研究(その5) －3段Closスイッチ網の並列制御方式の提案－,” 信学技報，vol.118，no.232，pp.1-5，2018.

6. 松本崇寛大，Labson Koloko,加藤 陽介, 小原 仁, “Benes網の並列ハードウェア制御回路の高速化,” 電気関係学会東北支部連合大会講演論文集，2B07，2019.

7. プトリナディラビンティアズマン，Labson Koloko,加藤 陽介, 小原 仁, “3段Clos網のノンブロック性能評価のためのシミュレーション条件の最適化,” 電気関係学会東北支部連合大会講演論文集，2B08，2019.

8. 柳 晋登，Labson Koloko,加藤 陽介, 小原 仁, “並列ハードウェア制御による3段Closスイッチ網の高速制御,” 電気関係学会東北支部連合大会講演論文集，2B09，2019.

9. 松本崇寛，野中翔太，Labson Koloko,加藤 陽介, 小原 仁, “3段Clos網の並列ハードウェア制御アルゴリズムの提案,” 電気関係学会東北支部連合大会講演論文集，P05，2020.

This page is left blank intentionally

Abstract

A study on improving the performances of three-stage Clos networks

Labson Koloko

Abstract

With the increase in demand for IP (internet protocol) traffic on the internet and many communication networks in recent past, high-capacity and fast switching routers and switches have become essential in the backbone of communication networks. Routers and switches play a very important role of routing signals from one node to another in a network. Their capacity and the speed at which this is performed are key to achieving the desired performances in current communication networks. Most of the routers and switches are constructed with a multistage switching fabric at the core of its switching. The three stage Clos network and the Benes networks are such multi-staged switching fabrics. Multi-staged switches are organized in such a way that several smaller switches in one stage are connected to other switches in another stage to form a network with larger capacity. Each switch in each stage connects to every switch in the next stage. Multi staging helps solve the scalability problem that comes with the crossbar switch. Using a single crossbar switch as a switching fabric does not allow for scalability as the cross point increases exponentially with an increase in the number of ports. The three stage Clos network, usually denoted by $C(n, m, r)$, where n, m and r represent the number of inputs/output ports of the input/output switches, the number of middle stage switches and the number of inputs/output switches respectively, has been one of the most widely used designs in many switching architectures. This thesis discusses two approaches in improving the performances of the switch. Firstly, the study considers the structure of Clos networks that include conventional crossbar switch elements which are composed of 2×2 basic switching elements often used in optical switches. The study highlights the fact that crossbar elements have a number of idle ports unused and discuss how these idle ports can be used to improve the non-blocking properties of the network. An elaboration on the non-blocking properties of this network is provided. This work shows that the lower bound on the value of m for rearrangeably non-blocking switches can be reduced by 25% of the original clos network when idle ports are used. It has also been demonstrated in this study that when $m = n$, the number of rearrangements is reduced to 1 regardless of the values of n and r. Typical conventional Clos switches require $r-1$ rearrangements in a worst-case scenario. The work also shows that the number of middle stage switches required for wide sense non-blocking in the Clos network can be reduced approximately by 25% lower than the conventional Clos switch. Secondly, this work discusses the design and implementation of fast and hardware-efficient parallel processing elements to set full and partial permutations in Benes network. A new design of parallel and distributed processing elements for configuring Benes network is proposed. The novel parallel algorithm can realize full and partial permutations in a unified manner with very little overhead time and extra hardware. The proposed design reduces the hardware complexity of processor elements from $O(N^2)$ to $O(N(\log_2 N)^2)$ due to the distributed architecture. The distributed architecture also allows for pipelining in the architecture which in turn reduces the time complexity of processor elements from $O((\log_2 N)^2)$ to

$O(\log_2 N)$. To further reduce the time complexity, asynchronous operation was introduced in part. The prototype is implemented in a field programmable gate array and the performance is investigated for the switch size of $N = 4$ to 32. Further a fast parallel algorithm for setting up a three stage clos network in which the component switch sizes have a power of two is presented. The algorithm is also implemented in hardware using field programmable logic gate array and its performance investigated through experiments. This work reports that the algorithm can operate as fast as the time complexity of $O(\log_2 N)$ up to a certain switch size, in contrast to the conventional algorithms which require a time complexity of at least $O((\log_2 N)^2)$. The experimental results demonstrate that the proposed designs outperform recent methods several times in terms of hardware and processing complexities.

Acknowledgements

First of all, I thank God for giving me the breath of life and for the good health I have enjoyed throughout my studies.

I am greatly indebted to my supervisor, Professor Hitoshi Obara for accepting me into his laboratory and for allowing me to carry out research under his supervision in the field of internet networks and Optical communication systems. My whole period of research under his guidance has been very rewarding. He has taught me how to do research. Approaching research problems in a piecewise manner, presenting research results, writing papers and how to give seminars. I have learnt a lot of things from him including his passion for research, being hardworking and his patience and kindness towards students. He made time to listen to me and discuss his ideas with me whenever I asked. He has always been encouraging and pushing me to work extra hard and never to give up. He has been more than a usual supervisor can be. His influence on me is far beyond. I am very fortunate and proud to have been his student. I cannot thank him enough! I remain indebted to him the rest of my life.

I owe my special thanks to professor Kumagai the head of Department who took me up as his student and Professor Motoshi Tanaka who allowed me to be part of his research laboratory after my supervisor retired.

Special thanks go to the other members of the supervisory committee, Professor Saitou and Professor Yamaguchi. They provided not only useful comments but they sparked numerous discussions that helped me understand my subject even better.

I am very grateful to Mr. Yosuke Kato who was always available to help me setup experiments and provisioning all the necessary equipment and tools needed in my research.

I am grateful to all the members of the Obara laboratory who made life easier and provided an enjoyable friendly working atmosphere in the laboratory.

I am grateful to my mum for her prayers and never ending encouragements, my brothers and sisters for their support and who continuously cheered me to push on.

Finally, I would like to thank the Japanese government for offering me the MEXT scholarship to come and study in Japan. I thank all members of the Electrical Engineering department, the staff, faculty members and fellow students who made the department a great place to work and study. My years at Akita University were very enjoyable.

Contents

List of Tables

List of Figures

List of Abbreviations and Acronyms

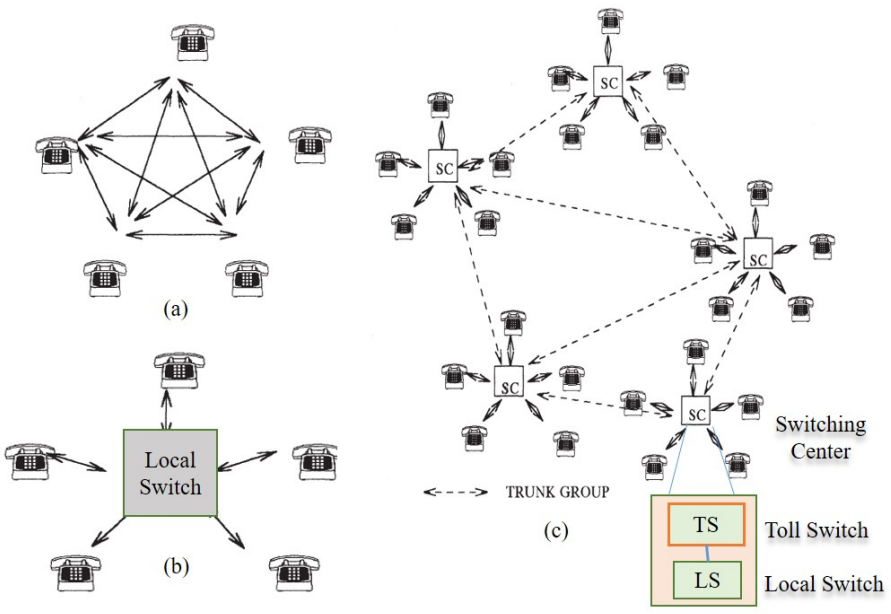
BNW	Benes network
BSE	Basic switching element
FPGA	Field programmable gate array
IP	Internet protocol
MIN	Multi staged interconnecting network
PE	Processor elements
PCU	Parallel control unit
RNB	Rearrangeably non blocking
SCB	Switch control bit
SE	Switching element
SNB	Strictly non blocking
SW	Switch
TSCN	Three stage Clos network
UTSN	Unfolded Two stage network
WSNB	Wide sense non blocking
XBS	Crossbar switches
NoC	Network on Chip

Chapter 1

Introduction

1.1 Background

Recently, the internet and many communication networks have experienced an exponential rise in the demand for internet protocol (IP) traffic as predicted by the cisco report [1]. This rise in demand is as a result of an increase in real-time services. With this increase in traffic, high-capacity, fast switching routers and switches are needed in backbone communication switching networks. Switching is an important component of any telecommunication system. In a telecommunication system, a switching system provides a means to pass information from one node to another node in a network and therefore is core to any communication system. For example, in earlier days of telephony networks, the role of the switching function was performed by a pair of cables that connected two terminals that needed to connect to each other. As can be seen in fig. 1(a) each terminal had to be connected to another terminal through a single transmission line. As the number of terminals increased to N, we see that the number of links increased as $N(N-1)/2$. This was not viable because these links remained idle most of the time and were only used during a short period of time when a call was generated, hence a central switch was introduced and each terminal only needed a pair of connections to it (figure 1(b)). These local switches were connected to toll switches that enable long distance communications (see figure 1(c))[2].



Source: *"Switching and Traffic Theory for integrated Broadband Networks,"* by J. Hui, Springer US 1990.

Figure 1: *Role of a switching function in telephony networks*

The general architecture of a switching node is shown in figure 2. A switching system

basically receives signals such as control signals, message signals and forwards them to a desired destination through a central module known as a *switching fabric*. These switch modules are controlled by a control unit that processes signaling links. (see figure 2).

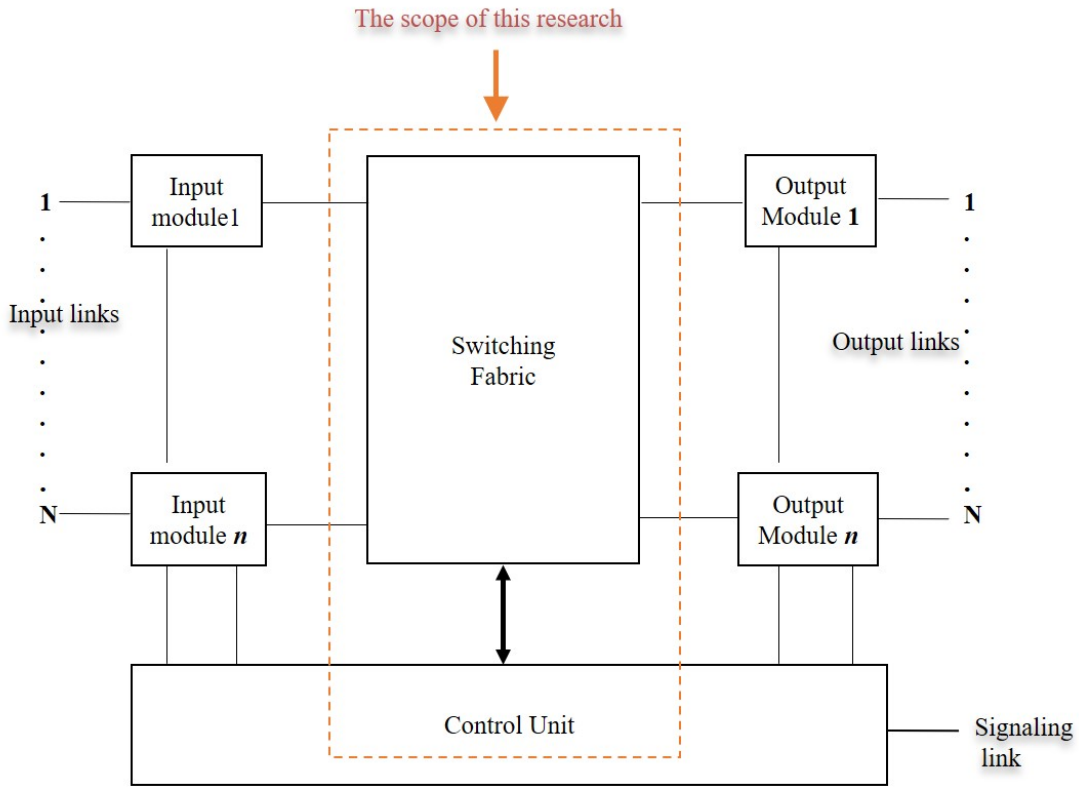


Figure 2: *General architecture of a switching node*

In order to meet high capacity and fast switching, lots of research has been devoted to the design and implementation of the switching techniques [3]. Switching has evolved over many years since Charles Clos initiated his classical circuit switching theory in 1953 [2]. An increasing trend towards mega data centers in which the switching architecture proposed by Clos are used has been observed and this has been necessitated in order to support the ever increasing demand for data [4]. The growth in communication brought about the need for interconnecting many transmission links. This was because traffic on these links needed to be multiplexed onto fewer links going to the same location for effective utilization of resources. Secondly, the traffic on these links needed to be demultiplexed to different transmission links going to different locations. A space switch was a candidate for interconnecting such links [2]. A space division switch can be implemented by either an electromechanical or electronic switch. Prior to the advent of time division switching, all telephone and telegraph switching machines were implemented using a variety of space division switching techniques, particularly Strowger (step-by-step) switches and crossbar switches [5].

1.2 Switching fabrics

At the core of any switch, is a switching fabric. The switching fabric is made up of basic switching elements (BSE) arranged in a regular and planar structure forming what is known as a crossbar switch (XBS) as shown in figure 3(a). BSEs have two input ports and two output ports. Each BSE can be in either of the two states at a given time; cross state with a switch control bit (SCB) of 1 (figure 3(b)) or bar state with an SCB of 0 (figure 3(c)), hence the name XBS. In the bar state, inputs port 1 and port 2 connects with outputs port 1 and port 2 respectively. This means that the BSE forwards its two input signals from the input ports straight to the two opposite output ports (figure 3 (a)). In cross state, input port 1 connects with output port 2 and input port 2 connects with output port 1 (see figure 3 (b)). BSE are arranged in a square matrix configuration to form larger crossbar switches with multiple inputs n and multiple outputs m [2]. When $m > n$, the XBS becomes a rectangular switch and if $n = m$, the switch is a square XBS. The XBS in figure 3(c) has $n = m = N$ and is therefore an $N \times N$ XBS.

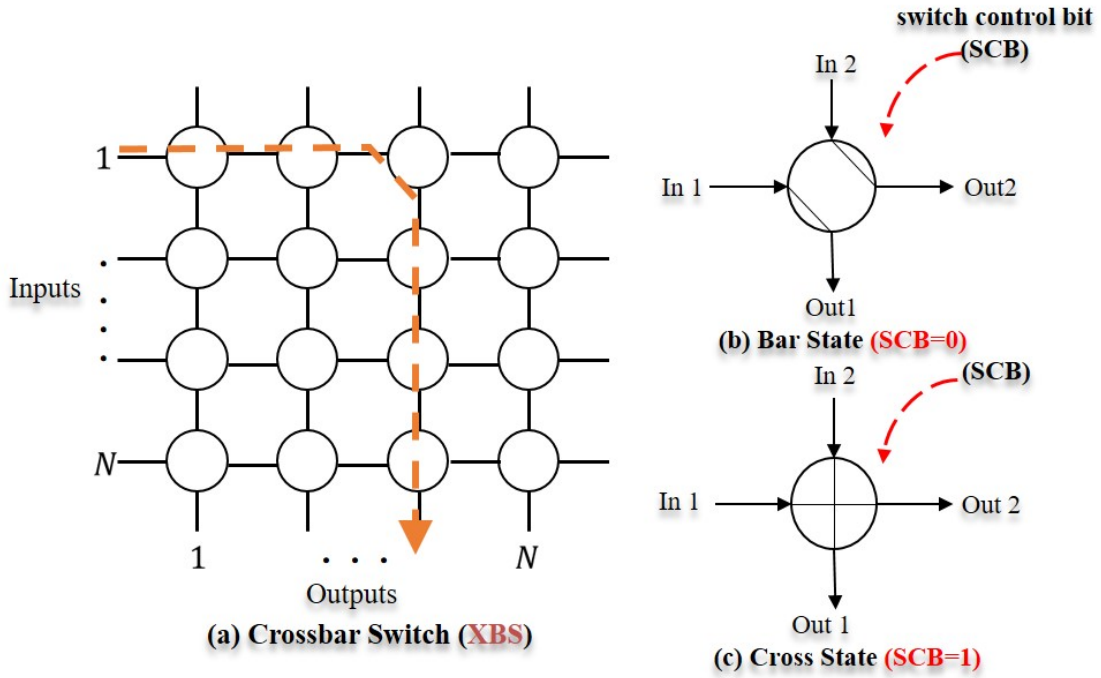


Figure 3: *Single stage architecture of the XBS with BSE*

The XBS takes its inputs and connects them to any of the m outputs. XBSs are represented as $n \times m$ where n is the number of input ports and m is the number of output ports (figure 3(a)). An XBS is considered to be strictly non-blocking (SNB) switches, meaning that any signal at any input port can be transferred to any output port without colliding or blocking with any existing signals in the switch and its control is very simple. A switching fabric is said to be blocking if there are path assignments which cannot be realized in the switching network. With an increase in the number of ports, the XBS becomes impractical to implement because the cross point requirement increases as nm and when $n = m = N$, then the cross point count becomes N^2.

1.3 Multi-staged Switches

During switching in an XBS, a single cross point in a row or a column is activated. To route a group of signals, several cross points are activated in an XBS. When several XBSs are connected between each other to form larger switches, such switches are referred to as multi-staged switching fabrics. XBSs are multi-staged to provide more routes at a reduced cross point count. Multi-staging has proved to be a good solution to the scalability issue of the XBS and has helped build large switching architectures using small BSEs (see figure 4). These play an important role in any switching devices such as routers and switches. Multistage interconnection networks (MIN) are used in building larger IP routers with higher capacities.

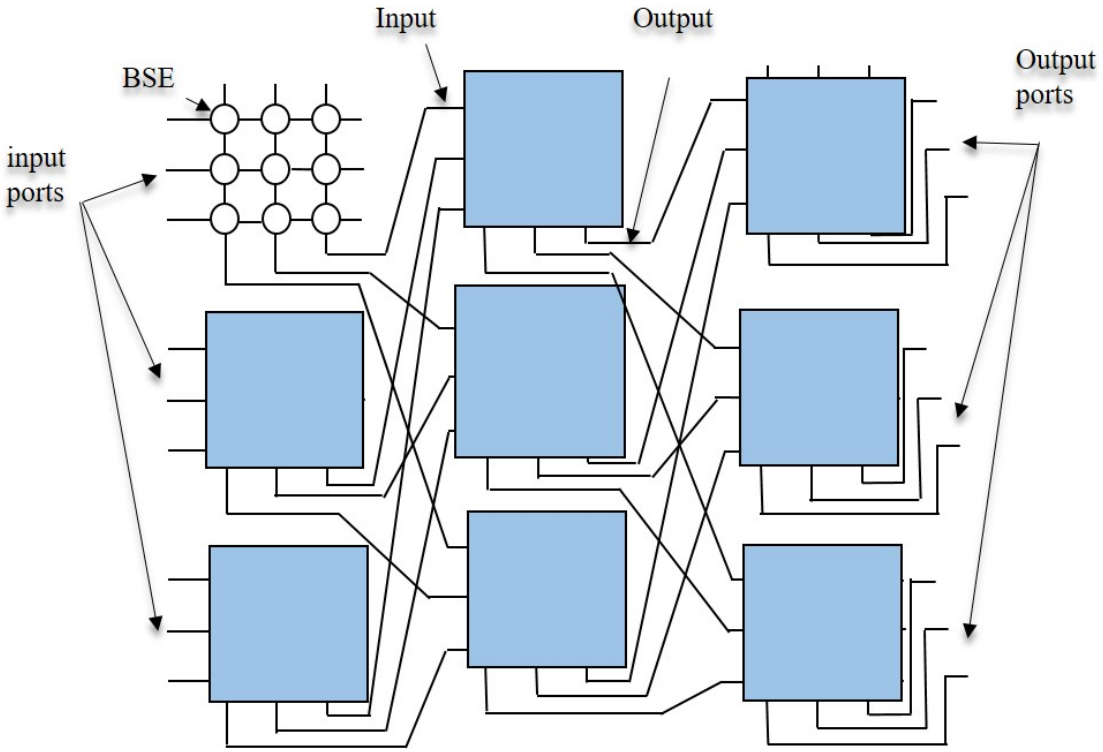


Figure 4: Multi staged architecture of XBS

Charles Clos in 1953 proposed a three stage network which is now known as a Three Stage Clos network (TSCN). The TSCN consists of r, $n \times m$ XBS in the first stage connected to each m, $r \times r$ XBS in the second stage. Each m XBS in the second stage is connected to a set of r, $m \times n$ XBSs in the third stage. A TSCN $C(n, m, r)$ shown in figure 5 is made up of three stages, where n is the number of input ports and output ports in the first and third stage XBSs respectively. m is the number of middle stage XBSs and r is the number of input/output XBSs in the first and third stages. Each of the first stage XBS is connected to the XBSs in the second stage only through a single link. Similarly, each of the second stage XBSs is connected to each of the third stage XBS through a single link. A multi-staged switch scales better than the XBS. An $N \times N$ TSCN has a much lesser cross point count of $N^{1.5}$ than the N^2 cross point count in a single $N \times N$ XBS [2].
The TSCN has some important properties that make it to be widely used in many

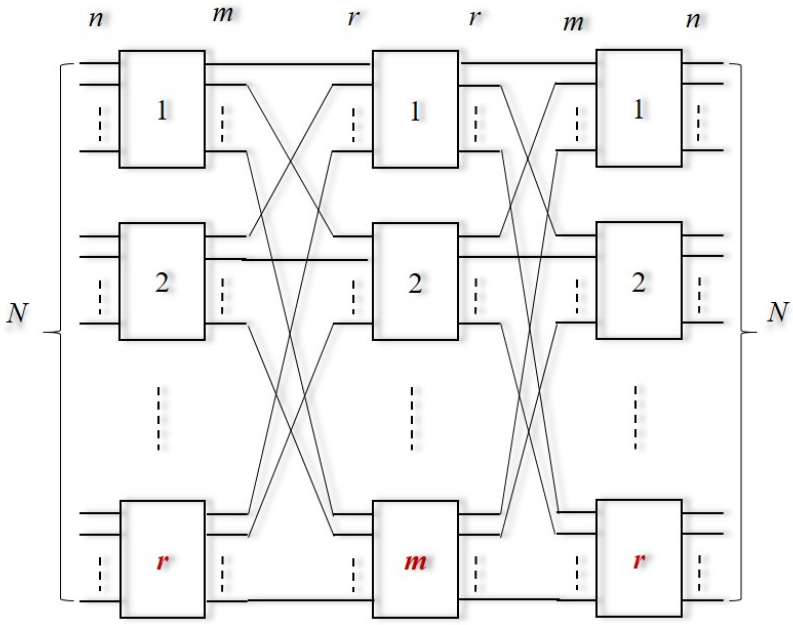


Figure 5: *An $N \times N$ three stage Clos network represented by $C(n, m, r)$*

communication networks. These properties can be fine-tuned by properly setting the values of m and n. A summary of these properties and their values is provided in table 1 below. To obtain the SNB property, set $m \geq 2n - 1$. This is proven as follows; in a

Table 1: Summary of the Properties of the TSCN

Number of middle switches, m	**property**
$m < n$	Blocking
$m \geq n$	Rearrangeably non-blocking (RNB)
$m \geq \lfloor 2n - n/F_{2r-1} \rfloor$ $F_k : k - th\ Fibonacci\ number$	Wide-sense non-blocking (WSNB)
$m \geq 2n - 1$	Strictly non-blocking (SNB)

worst case scenario, if a free port exists at the input port of an input XBS A, we have $n - 1$ existing connections that correspond to a set of unreachable number of middle stage switches. Similarly, a free output port exists at the output XBS B, we have the other $n - 1$ connections connected through another set of $n - 1$ unreachable middle stage switches. To provide a free connection path, Clos showed that an extra middle stage switch (shown in green in figure 6) must be added, hence there must be at least $2(n-1)+1$ middle stage switches. When $m \geq n$ the TSCN becomes RNB. A TSCN can also be wide sense non-blocking (WSNB) if a route can be provided in a network but some rule must be used when setting up a path otherwise blocking may occur later. If the number of middle stage switches is less than n the switch is referred to as a blocking switch since some connection paths cannot be provided in the switch.
Another multistage switching network is the Benes network (BNW). It is a special type of TSCN with $m = n = 2$ and is composed of 2×2 BSEs as building blocks. It has $2\log_2 N - 1$ stages of which each is made up of $N/2$ 2×2 switches. The number of inputs and outputs is given by $N = r \times r = 2^n$. The central stage is composed of two

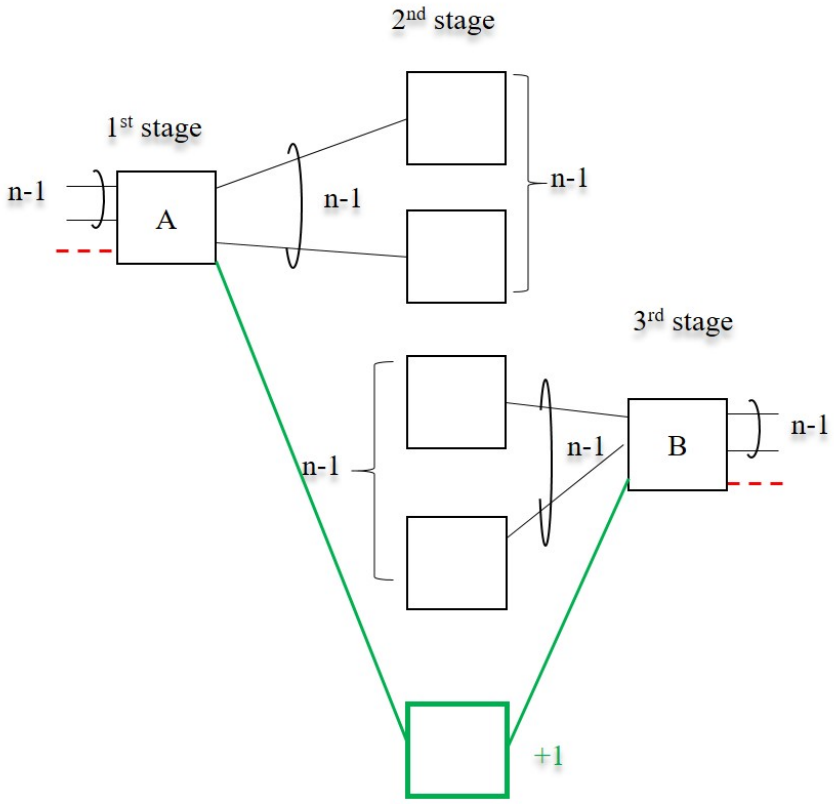


Figure 6: *Strictly non-blocking condition for a worst case scenario with minimum number of middle stage switches required*

$N/2 \times N/2$. Each of the $N/2 \times N/2$ switches are made up of $\log_2 N - 1$ 2×2 BSE (see Fig. 7). In the case where $N = 4$, the middle stage switch is composed of 2 2×2 BSE. When N is increased, the center stage can be repeatedly replaced by $N/2 \times N/2$ to form a rearrangeably non-blocking (RNB) switch. The total number of switching elements required is $N \log_2 N - N/2$.

In an RNB switch, a new path can always be set up between an existing free input port to an existing free output port but already setup paths must be rerouted to accommodate the new path.

Benes and Clos switches with 5,7,9, or more stages can be realized.

1.4 Control algorithms

In order to set up connection paths in non-blocking multistage switches, special routing algorithms are required. These are important because they determine the path setup time and the system reliability of the network which are important factors in determining the performance of a switching network. In setting up paths from an input i to and output j in the network, sequential search algorithm with complexity of $O(m)$ per connection is used resulting in a total of $O(Nm)$ for a switch size of inputs N as shown in figure 8. These sequential algorithms have long processing time. Therefore, parallel algorithms which have lower processing time have been introduced. It is a well known fact that

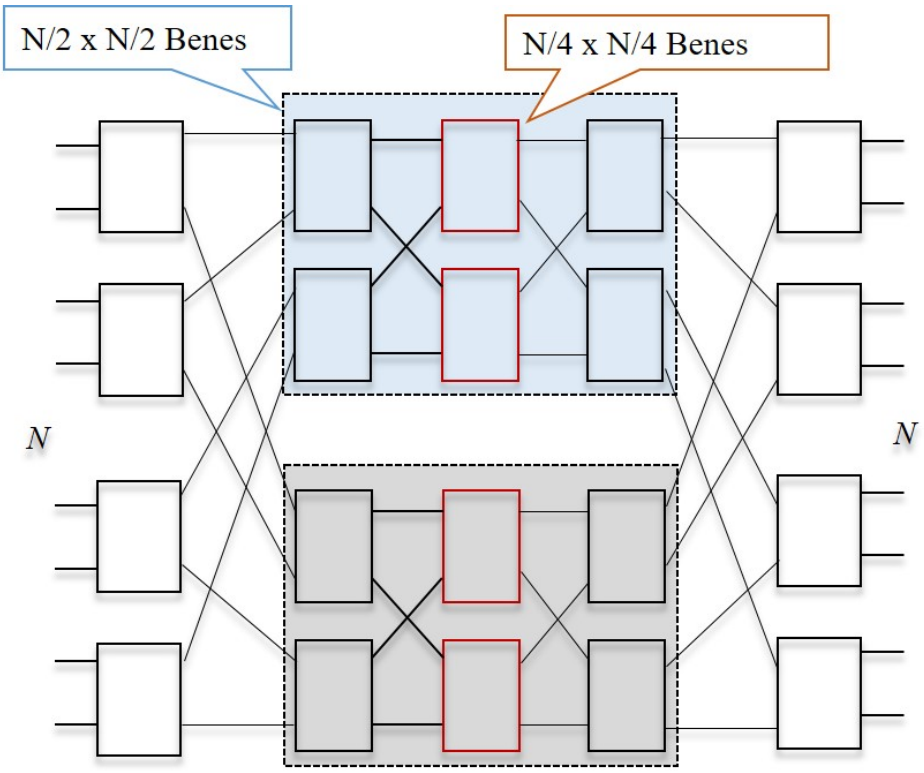


Figure 7: *Recursive construction of the Benes network*

parallel processing reduces the processing time of a given task [6]. This work seeks to improve the time for path setup in the Benes and Clos network.

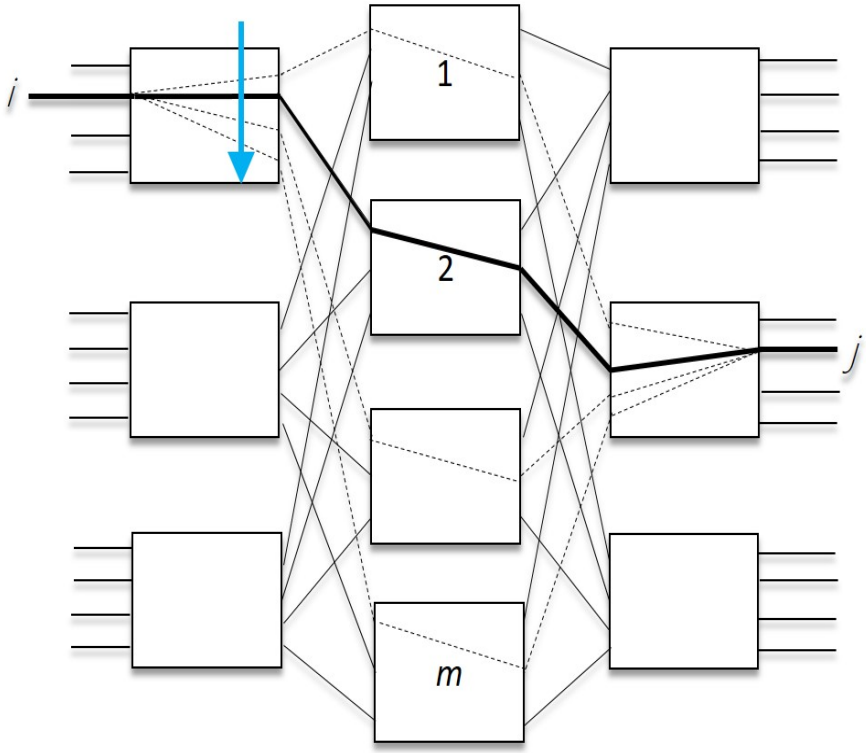


Figure 8: *Sequential route setup in the TSCN*

1.5 Outline of this thesis

This dissertation is organized as follows; chapter 2 begins by reviewing the conventional structure and properties of the TSCN. It presents some related works that have been done in the area of structure and properties of three stage clos networks. It also covers the recent works done in routing control algorithms for Benes and Clos networks. Chapter 3 introduces the design and operation of our proposed Clos architecture. It introduces a new design in which conventional XBS are modified in two different ways and applied to designing the new TSCN architecture. The first is the unidirectional modified XBS and the second is the bidirectional modified XBS. Chapter 4 presents the simulation and experimental call setup in the new architecture. Performance results of the new proposed architectures are discussed and reported. Chapter 5 introduces a new emerging design principle options for relatively small-capacity switches. The non-blocking properties of unfolded two stage networks are discussed. In Chapter 6, the design of parallel control algorithms for the Benes and clos networks are presented. The design of parallel and distributed PEs for processing full and partial permutations in BNW is described.
The hardware implementation of the algorithms performed on an FPGA to realize higher speed than the clock rate using asynchronous operation is also reported. Performances of these algorithms are evaluated through experiments. Experimental results of the performances are highlighted in this section. Finally, chapter 7 concludes the dissertation and suggests future works.

Chapter 2

Related works/Literature review

2.1 Previous clos structure and properties

The clos architecture has been the most practical and cross point–efficient design principle for large switching networks [7]. It has been applied to various types of switches such as space switches [8], time division multiplexed switches [9], packet switches [10] and optical switches [11]. This is because of the non-blocking properties that the TSCN has. Clos networks were at the beginning invented for application in telephone exchange systems. However, recently it has been used in many applications ranging from electrical and optical cross connects which are essential for building communication networks that are economical with highly multiplexed signal links [12]. Clos networks have been classified based on their connecting capabilities. Some have been classified as strictly non-blocking (SNB) network [13], rearrangeably non-blocking (RNB) [14], wide sense non-blocking (WSNB) [15, 16, 17] and repackaged network [18]. In each of the classes, the relationship of the parameters n, m and r have been analyzed. By setting these parameters appropriately, the above properties of the clos network can be realized. The SNB network can be obtained when m is set equal to $2n - 1$. This means that any connection request can be established from the input side of the switch to the output without a specified routing algorithm. In an RNB, a connection request can be made, but existing calls must be rerouted or rearranged to pave way for the new connection request. In WSNB network, a new connection request can be made but a specified routing algorithm must be used in order to avoid blocking in the switch. In a repackable network, any connection is achieved by a repacking algorithm which must ensure that calls are concentrated to some middle stage switch and reduce the load on some other middle stage switches before adding another call to the network [18]. Most of these designs have sought to achieve lower blocking probabilities, increased port count and a lower cross point count within the switch network.

The structure of the TSCN is built based on each of these properties. Since the XBS has been the building block in MIN, it has been studied carefully. In [19], a component efficient design for a parallel optical XBS which utilizes the idle ports as extra input ports and/or output ports was proposed. It was observed that conventional parallel XBSs composed of 2×2 BSE have a significant number of idle internal routes between its idle ports. By utilizing its internal idle routes, it reported a one-quarter decreased switch count from N^2 to $\frac{3}{4}N^2$. Here, the modified architecture was only applied to single and two stage architectures. Because single stage crossbar switch architectures do not fit dilation as their cross point size increases proportionally to the switch size as N^2, we focus on applying this to the three stage switch architecture. Also two new design examples were proposed which are; unidirectional and bidirectional. If both are utilized in a proper manner, an architecture with better performing non-blocking properties can be yielded. The concept of idle ports was also applied in [20]. Idle ports of a PILOSS switch were used in bidirectional mode. An Si-wire thermosoptic 4×4 PILOSS switch in which a bidirectional use of a single path-independent-insertion-loss switch for polarized

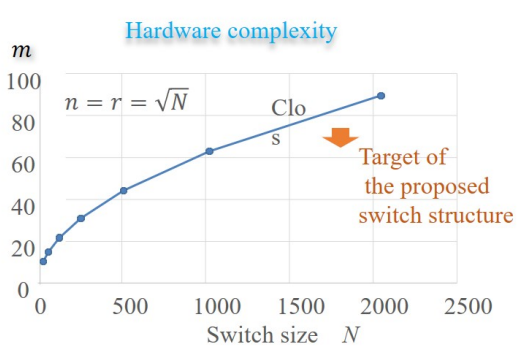
diversity was proposed. The polarized insensitive switching was realized with a single PILOSS switch consisting of N^2 as opposed to $2N^2$ element switches.

Two principle two-stage switch architectures both composed of two identical optical XBSs with an extra set of input and outputs were proposed in [21]. The cascaded switches based on the utilization of idle ports and idle internal routes in the XBS were reported. The number of cross points, switch control complexities and key optical performances were compared to known cascaded conventional XBS. In [22, 23] a packet based switch which focused on traffic balancing and congestion management was proposed. The switch is called a Clos Unidirectional Network on chip (Clos-UDN). In comparison to the conventional Clos switch, the whole XBS was replaced by a crossbar-like network on chip (NoC) which operates in a unidirectional mode. The proposed multi hop NoC operated as a crossbar with shorter inter router wires than those in a single-hop conventional crossbar. F. Hassen and L. Mhamdi et.al, [24, 25] suggested a three stage clos network which was based on Multi-directional NoC (MDN central modules). Both of these works replaced the conventional crossbars with MDN modules. MDN modules are modified types of UDN switches with their input and output ports operating in multi-directional mode. The NoC based crossbar modules are expensive. To increase the port count, means having larger NoC modules which ultimately increase the cost of designing them. In a white paper of mentor Graphics [26], the creation and simulation of a 2 $\times$ 2 optical switch is illustrated. The switch studied employed a double sided mirror whose movement is controlled by an electrostatic, comb drive actuator. A set of folded springs controls the actuator movement. The mirror slides out to the intersection of two perpendicular alignment grooves and then retracts when actuated. Two pairs of optical fiber emitters sits in the alignment groves. In the cross state, the comb drive is actuated and the mirror sits in the groove at the intersection. The mirror reflects the light beam from left input to direct it to the lower output and from the top to the right sides of the switch. In the through state, the comb drive actuates and the mirror retracts. The light beam from left goes straight and is received on the by the output on the right side of the switch. The top input emits the beam to the bottom output. These works report the utilization of the idle ports in a 2 $\times$ 2 optical switch. Simulation of this switch was done in S-Edit software to allow for quick assembly.

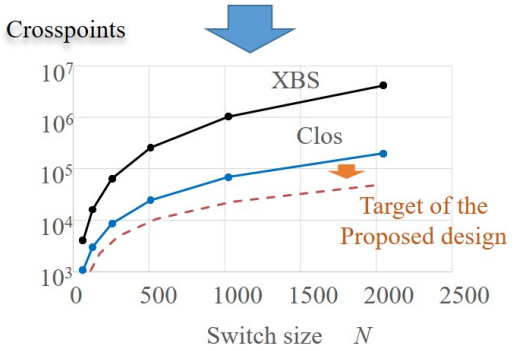
2.1.1 Summary of Contribution

This thesis reports an improved architecture of the TSCN that uses bidirectional XBS. The XBS is modified by utilizing the idle ports. The modified XBSs are applied to the TSCN to form a new architecture. By utilizing the idle ports in the modified XBSs, the non-blocking properties of the new proposed TSCN are investigated through theoretical and computer simulations through examination of additional routes. The total cross point count and the number of rearrangements are also analyzed. This research reports an improved architecture of the TSCN with a reduced number of middle stage switches (figure 9 (a)), reduced cross point count (figure 9(b)) and a reduced number of rearrangements in an RNB clos network. Parallel processing in BNW is investigated and reported with a reduced processing time(figure 9(c)). The reported technique is then applied to a TSCN to come up with a fast parallel algorithm for setting up paths in a TSCN with a component switch size of a power of two. The algorithms for Benes and Clos networks

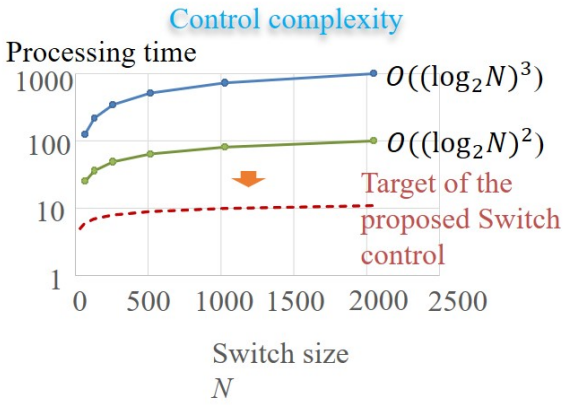
are both implemented in FPGA. The experimental results of their complexities which highlight the performance of the design in comparison with conventional methods are reported.



(a) *hardware complexity*



(b) *Control complexity*



(c) *Crosspoint complexity reduction*

Figure 9: *Summary of objectives of proposed target (a) Hardware complexity (b) crosspoint complexity reduction (c) Control complexity*

2.2 Recent Benes and clos routing control algorithms

2.2.1 Control algorithm in BNW

The BNW is an example of a TSCN. As can be seen in figure 7, the basic building blocks are the 2×2 BSEs. It corresponds to the TSCN with $n = m = r = 2$ and is a RNB network. The BNW provides a unicast connection between N inputs and N outputs where $N = 2^n$. The BNW comprises $2n - 1$ stages each of which comprises $N/2$ 2×2 switch elements (SE) [27]. There is a total of $O(N(\log_2 N))$ SEs, which satisfies the theoretical lower bound of the non-blocking switch complexity [28]. Recently, several parallel algorithms have been investigated to apply the BNWs to high-speed time-division multiplexed systems of the agile responsibility to arbitrary permutations [29]. In 1981, Lev et al. formulated parallel algorithms for Benes and Clos networks based on a mathematical graph theory approach [30]. In 1982, Nassimi and Sahni developed a parallel

algorithm for BNWs based on an engineering (parallel computing) approach [31]. These works were performed concurrently and independently in these two different approaches. Both achieved a time complexity of $O((\log_2 N)^2)$ with completely interconnected parallel computers for full permutations with each input corresponding to a unique output. In several practical applications, some inputs and outputs may not have any connection requests (idle state) on them resulting in what is known as a *partial permutation*. In this scenario, original parallel algorithms for BNWs suspend processing when an idle connection is encountered. Parallel algorithms for BNW are required to handle such partial permutations efficiently and effectively[32]. To address this issue, two approaches have been suggested. Firstly, in 1995, Lee and Oruc introduced *quadruple datasets* to match the idle inputs with idle outputs to act as dummy destinations [33]. Once each input is assigned to a unique output, conventional parallel algorithms are then applied and can therefore work effectively. This approach requires a complicated data structure and a considerable pre-processing time. Secondly, in 2002, Lee and Liew proposed an additional *merging* process which fits in efficiently with the original algorithms [34]. In 2017, the Lee's algorithm was implemented in Field programmable gate array (FPGA) for switch size of $N = 8$ to 32 by Jiang and Yang [35]. They reported a well-functioning algorithm but it incurred significant overhead time. For the first stage processing of the parallel control unit (PCU) of switch size $N = 16$, which only required four clocks for completion, they reported that it required up to 17 clocks. Their approach which was based on the crossbar-like centralized architecture resulted in an increased PCU hardware requirement of $O(N^2)$ [36]. Prior to their implementation of the Lee's algorithm in FPGA, in 2009 Kai et al designed a PCU to configure BNWs. The design only focused on the first stage of the PCU for BNW of switch size 16×16 [37]. The approach however employed a distributed architecture rather than a centralized one. This reduced the hardware complexity from $O(N^2)$ to $O((\log_2 N)^2)$. They further suggested that the time complexity can be reduced from $O((\log_2 N)^2)$ to $O(\log_2 N)$ using a pipeline architecture suggested by Lee and Oruc. In as much as this algorithm has these advantages, it could not handle partial permutations.

2.2.2 Summary of contributions

In this thesis, a parallel algorithm that sets up both full and partial permutations with less overhead time and less additional hardware cost is reported. A PCU with distributed PEs which generates SCB in a pipelined and in part asynchronous manner is constructed in an FPGA.

2.2.3 Control algorithms in TSCN

The TSCN requires a setting time of $O(N \log_2 N)$ under a sequential algorithm [38]. This processing time is too long for its application in data centers and other communication networks. In order to improve the switching time of the TSCN, several parallel algorithms have been proposed. Lev et al. proposed an $O((\log_2 N)^2)$ algorithm based on the graph theory [30]. Zheng et al. came up with a $O(\sqrt{N})$ algorithm which was based on a distributed pipeline routing architecture [39]. Most of these algorithms have no reports of implementation in hardware. Besides, the full advantage of the switch having the switch

size of the power of two had not been fully taken advantage of in these algorithms. Several parallel routing algorithms for BNW in which the switch size is given by $N = 2^k$ have already been developed and their hardware implementations in FPGA already reported [37].

2.2.4 Summary of contributions

This thesis reports a new parallel routing algorithm for setting up TSCN in which the component switch sizes have a power of two using the above developed BNW for the first time. The algorithm is implemented on an FPGA. The performance of the new algorithm is analyzed and compared with the latest conventional methods.

Chapter 3

Design and operation of the proposed structure

3.1 Crossbar modification

The XBS is made up of 2 × 2 BSE which are either in bar or cross state. To route a call in an XBS as shown in fig. 10, the BSE in the second row and third column must change its state to bar. Recently, a modified XBS switch using idle ports to provide an extra set of input and output reported in [40, 41]. The reported modified XBS single-stage architecture has a drastically reduced cross point count and a simple routing algorithm. The modified XBS is derived from a Conventional XBS which has idle ports on the top and the right side as shown in figure 10.

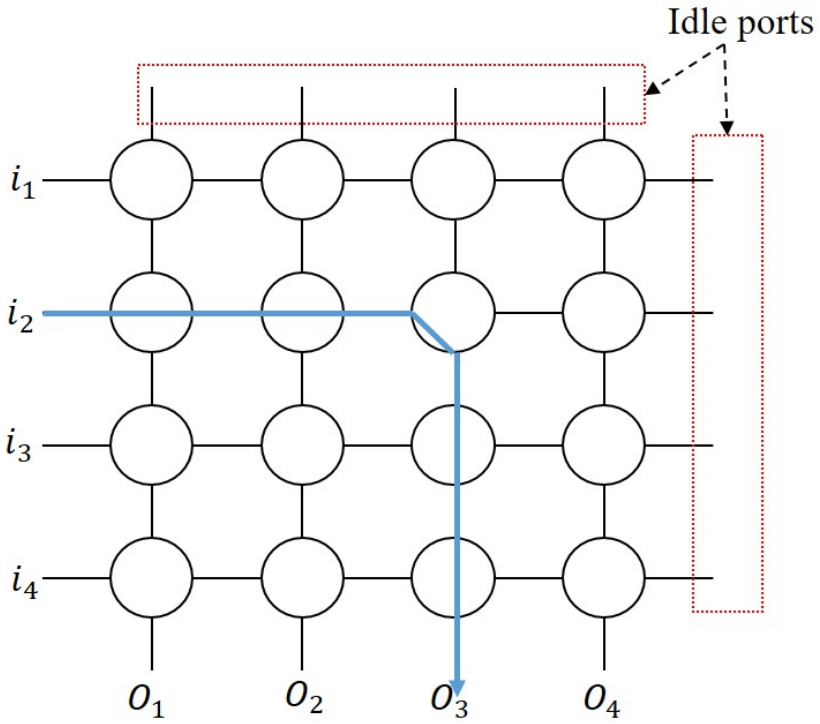
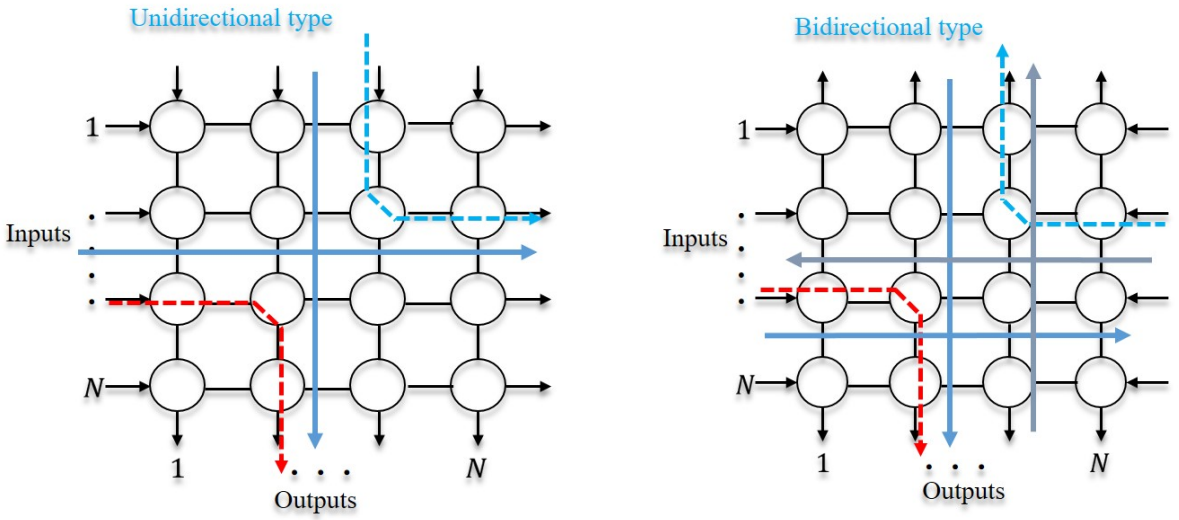


Figure 10: *Conventional crossbar (XBS) Switch with idle ports*

The idea is to utilize these idle ports to provide additional routes in the XBS without increasing the cross point count. Two types of modified XBS one with unidirectional and the other with a bidirectional mode were reported in [21]. These are shown in fig.11.

The unidirectional mode of figure 11 (a) is modified in such a way that the top ports are used as extra inlets and the idle ports on the right are used as extra outlets. The direction of travel of the signal in this switch is such that it moves from left inlets to the right outlets and/or left inlets to right outlets turning to bottom outlets. The signal can also travel from either top to bottom outlets or from top towards bottom and turning to the right outlets. This is considered as unidirectional switch because its either left to



(a) *Unidirectionally modified XBS* (b) *bidirectionally modified XBS*

Figure 11: *Two possible types of XBS modification (a) Unidirectional modified (b) bidirectionally modified*

right or top to bottom or left to right going bottom or top to bottom going right. The bidirectional one shown in figure 11 (b) is modified by using the right side idle ports of the XBS as inlets and the top ones as outlets. Signals can travel from left towards right turning to the lower outlets and can also travel from the right side going towards left turning to the upper outlets. Because of this direction of travel in both right and left directions by two different signals, this switch is referred to as *bidirectional switch*. These two types of XBS are applied to the TSCN in order to provide additional internal routes without increasing the size of XBS which, in turn yields a smaller number of XBSs, reduced cross points and a reduced number of rearrangements in RNB switches.

3.2 Application of XBS to TSCN

The modified XBSs are used to design a new TSCN architecture. Firstly, a special case of a bidirectional modified TSCN is described in which the number of rearrangements are reduced to only one regardless of n, m and r at a negligible cost of extra crosspoints. The non-blocking properties of the TSCN such as the SNB, WSNB and RNB depend on the number of middle stage switches m. It has been observed that by adjusting this parameter appropriately, the properties of the TSCN are also changed. The proposed design reported in this thesis focuses mainly on the middle stage XBSs.

3.2.1 Bidirectional TSCN

The proposed general architecture of the modified TSCN using modified XBS is as shown in figure 12 below. The proposed architecture has three stages similar to the conventional TSCN. Each stage is composed of XBSs represented by I_x , M_y and O_z with ($1 \leq x \leq r$, $1 \leq y \leq m$,$1 \leq z \leq r$). The input stage XBS I_x ($1 \leq x \leq r$) are $n \times 2m$ ordinary XBS

and its outputs $2m-1$ and $2m$ are connected to the left and right side of the middle stage XBS respectively. Similarly, the third stage also is composed of $2m \times n$ O_z XBS. Their inputs $2m-1$ and $2m$ may simultaneously receive signals from the middle stage switches. The total number of cross points in the input and output stages become twice as large as the conventional TSCN. However, the number of middle stage switches is reduced drastically. The middle stage has $r \times r$ bidirectional XBS with extra sets of r inlets and r outlets. It should be stressed here that each of the first stage I_x XBS provides two links to each of the rows of every M_y middle stage XBS. The columns of a modified XBS are used as outputs and feed to the inputs of the XBS in the next stage. The proposed structure also provided two links from each of the M_y to each O_z. For example, two signals from I_r can both be routed through the $M1$ XBS to any two third stage XBSs. The two signals may simultaneously coexist in a single row of middle stage XBS. In a conventional TSCN, only a single link is provided between each switch in each stage.

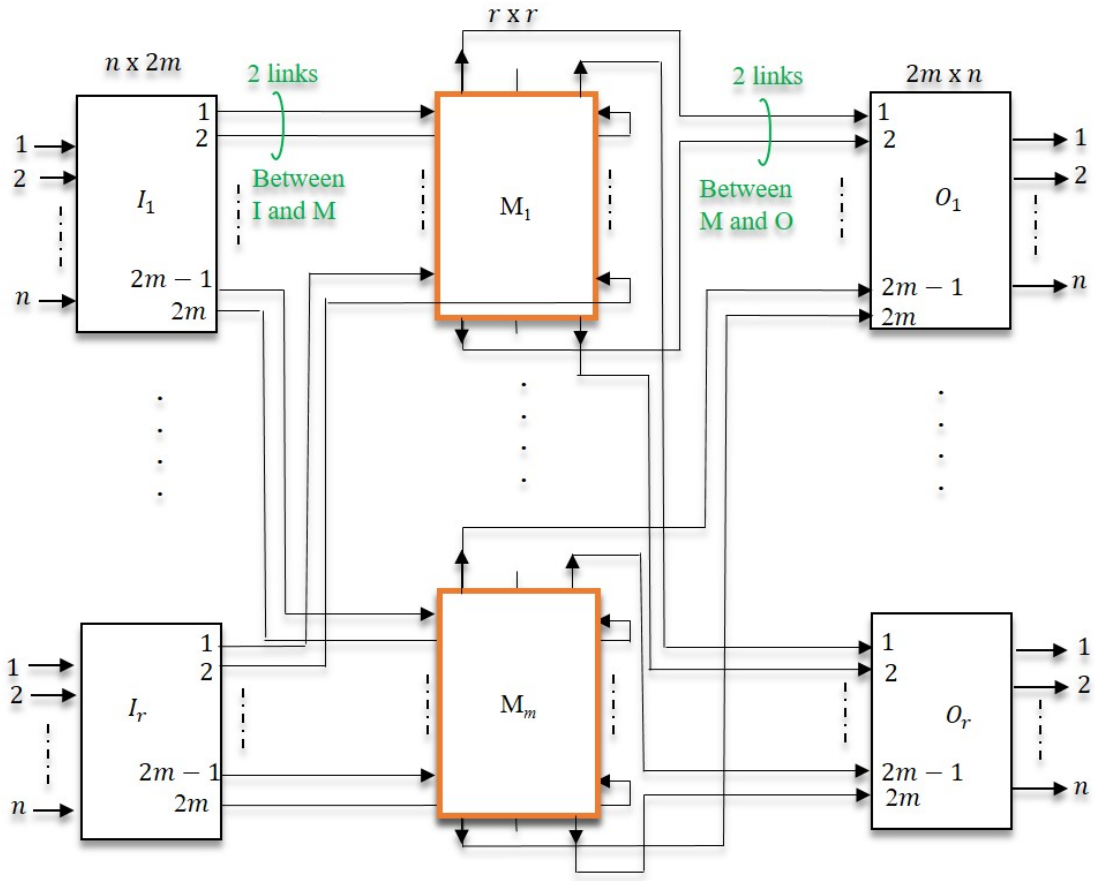


Figure 12: *Two possible types of XBS modification*

The utilization of idle ports introduces additional routes and therefore more than one signal from each XBS in one stage can be routed to another XBS in another stage. The other special design is as shown in figure 13. In this design the middle stage is composed of $m = n$ modified XBS used in bidirectional mode. Each first stage is composed of ordinary r $n \times n$ XBSs. Each of the output from the first stage is fed either to the left or right inlet of the middle stage switch through a
1×2 switch. At the second stage, it is switched to the third stage from the top outlets of the switch. The wiring pattern of the inlets at the left side and outlets at the bottom

sides is the same as that in a conventional XBS of the TSCN.

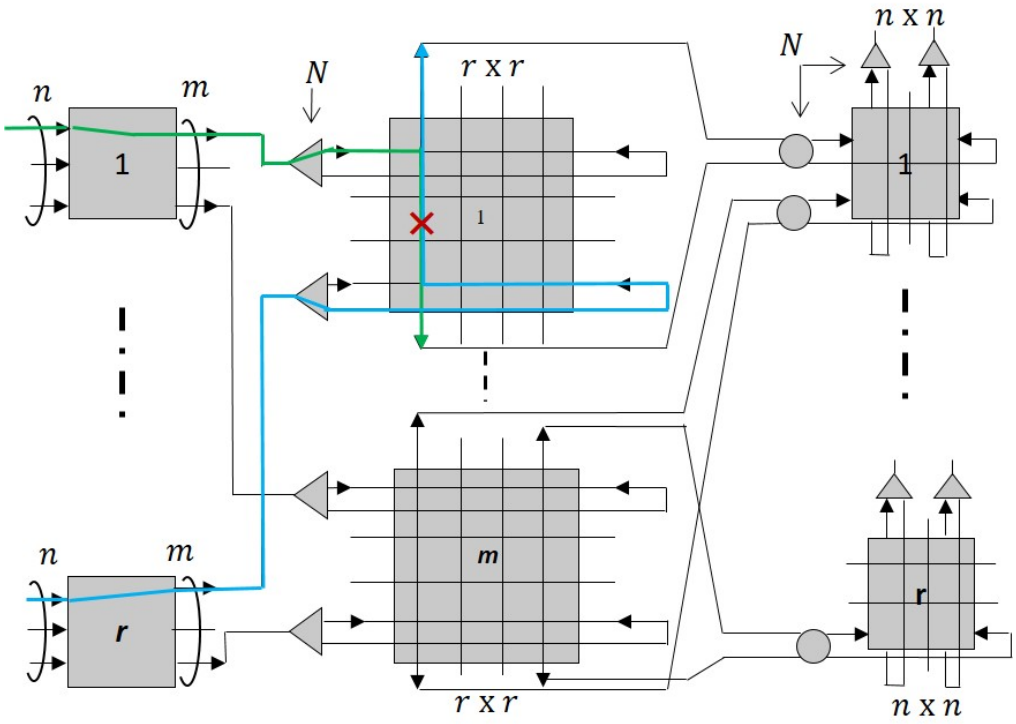
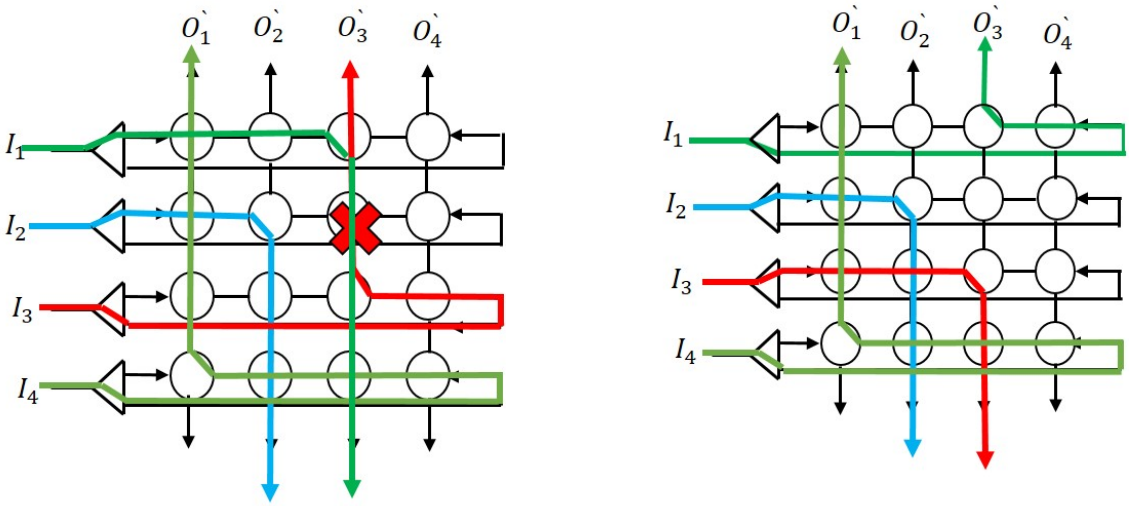


Figure 13: *Special configuration with $m = n$ middle stage switches*

The signals are then fed to the third stage switches through a 2 × 2 BSE. As a result of the 1 × 2 switch in the first stage, only a single connection request from the first switch in the first stage to the first middle stage switch occupies a row of the middle stage switch although it has two inlets at the end of each of its rows. Therefore, only a single signal is allowed to travel along the row of the middle stage switch either from the left side going right or entering to the right side going left. This prevents row blocking from occurring in any row of the second stage of the TSCN. However, blocking can occur when two input signals from different input switches are destined for an identical switch. As can be seen when the signal indicated in blue seeks to connect to the upper side in the first column, blocking may occur with the green signal which is already destined for the bottom output in the same column. This is further illustrated in figure 14a, i.e., if a signal from a lower input switch on inlet $I3$ is fed to the right side, and its destination is $O_3^{'}$, we see that blocking can occur in the column since another signal from input switch on inlet $I1$ is already occupying the column. This is known as column blocking. As can be seen in figure 14b, this kind of blocking can easily be resolved by exchanging a pair of routes to other switches after a conventional rearrangement control. The blocking that occurs here is resolved by diverting the existing connection to an alternate route through the same column swapping only a single pair of routes. This limits the number of rearrangements in the proposed $C_{\mathrm{B}}(n, n, r)$ to only a single rearrangement. Therefore it requires at most a single rearrangement regardless of n or r. therefore the number of rearrangements $R_B(n, n, r)$ is expressed as

$$R_B(n, n, r) = 1 \tag{1}$$



(a) *column blocking*

(b) *Blocking resolution by exchanging a pair of connections*

Figure 14: *Column Blocking and blocking resolution*

The second case of the bidirectional TSCN has a reduced number of middle stage switches. In this design (figure 15), Each input to the first stage-switches I_x has a 1×2 switch to route the signal either to the left or to the right side of the switch. The extra unused ports on the right side of the switch are used as extra inlets and the top ports as extra outlets. The output signal of the first stage switches is tapped from either the top or the bottom depending on the side the signal entered from. The signal entering through the left side of the XBS can travel to the bottom outlet while the signal entering through the right side can travel to the upper outlet. A pair of column outlets of the first stage are fed to the same row of each of the second stage switches. The upper outlet is fed to the left side of the middle stage XBS and the bottom outlet to the right side of the middle stage switch. Similarly, the column outlets of the second stage switches are fed to the third stage switches through 2×2 BSE which distributes and exchanges the signal in the third stage. In the third stage, the output ports are equipped with 2×1 BSE to select the appropriate route from where the signal is emanating. The use of idle ports in this proposed design allows each of the first stage switches to route at most two signals to the same middle stage switch.

Consider a conventional TSCN $C(n, m, r)$ with $n = 4$, $m = 4$ and $r = 4$. In this switch, we require a minimum of $m = n = 4$ middle stage switches to connect all $n = 4$ inputs from a single XBS, i.e. for the switch to maintain RNB properties. Since blocking has been avoided in the first and third stages, we focus mainly on the routing of calls in the middle stage switches only. This premise is also based on a well-known fact that the properties of the TSCN is mainly dependent on the middle stage switches. In a conventional TSCN, two calls from the same first stage switch cannot be switched to the third stage through the same middle stage XBS. We also see that two calls originating from two different first stage switches need at least two rows and two columns in a single middle stage XBS to be switched to the third stage. This means a total of 2 rows and 2 column resources are used up in a single XBS to route two calls as can be seen in Fig. 16(a). The bidirectional TSCN represented by $C_B(n, m, r)$ allows for two calls coming from the same input stage XBS to be routed to the third stage through the same middle

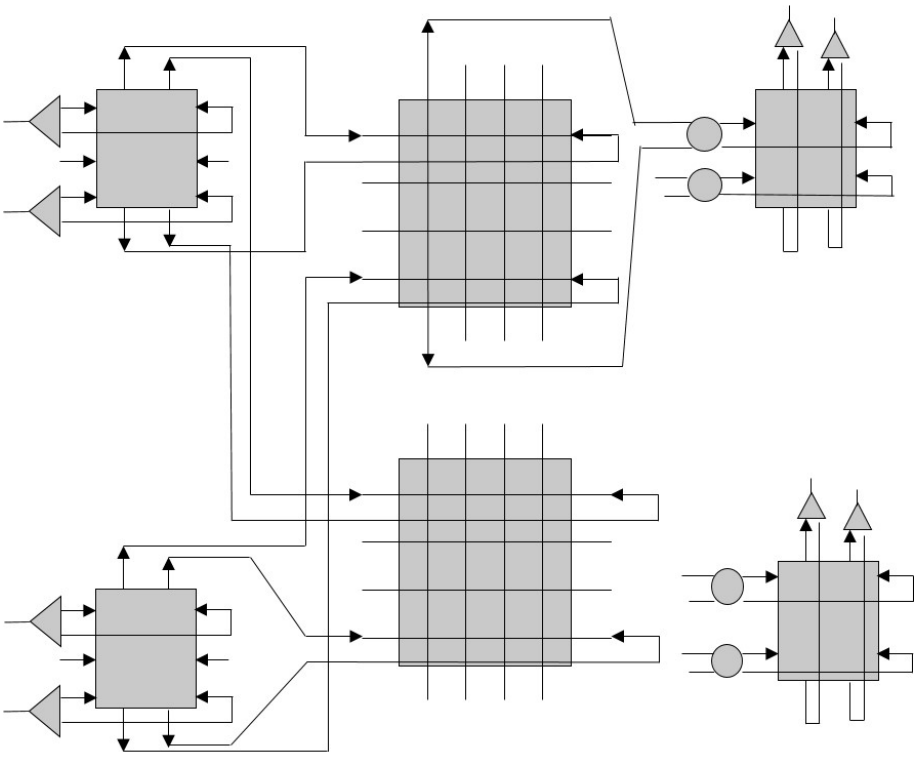


Figure 15: *bidirectional modified TSCN with* 1^{st} *and* 3^{rd} *stage modified*

stage XBS because it has additional inlets and outlets. Two calls from two different first stage switches headed for the same destination switch in the third stage can use two different rows but will share a column. This will use up 2 rows and a single column (3 resources) (see Fig. 16b). Similarly, two calls coming from the same first stage switch can share a row to reduce the resource requirement to 3 resources (see Fig. 16 c). A special case where two calls from the same first stage switch headed for the same third stage switch will share a row and a column resulting only in using 2 switch resources (see Fig. 16d).

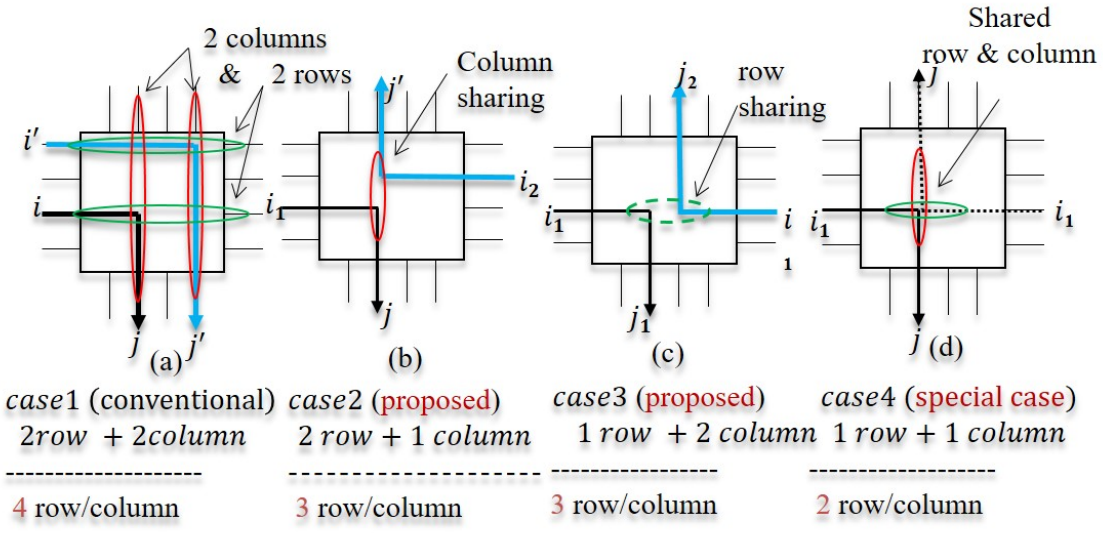


Figure 16: *Sharing of rows and columns by a pair of connections in XBS*

Let us see how the sharing of rows and columns in the proposed TSCN $C_B(n, m, r)$ allows routing $n = 4$ inputs through $m < n$. Consider routing all calls from the first stage switch to the last switch of the last stage i.e. input SW $1, 2, 3, 4$ to output SW $4, 3, 2, 1$ respectively.

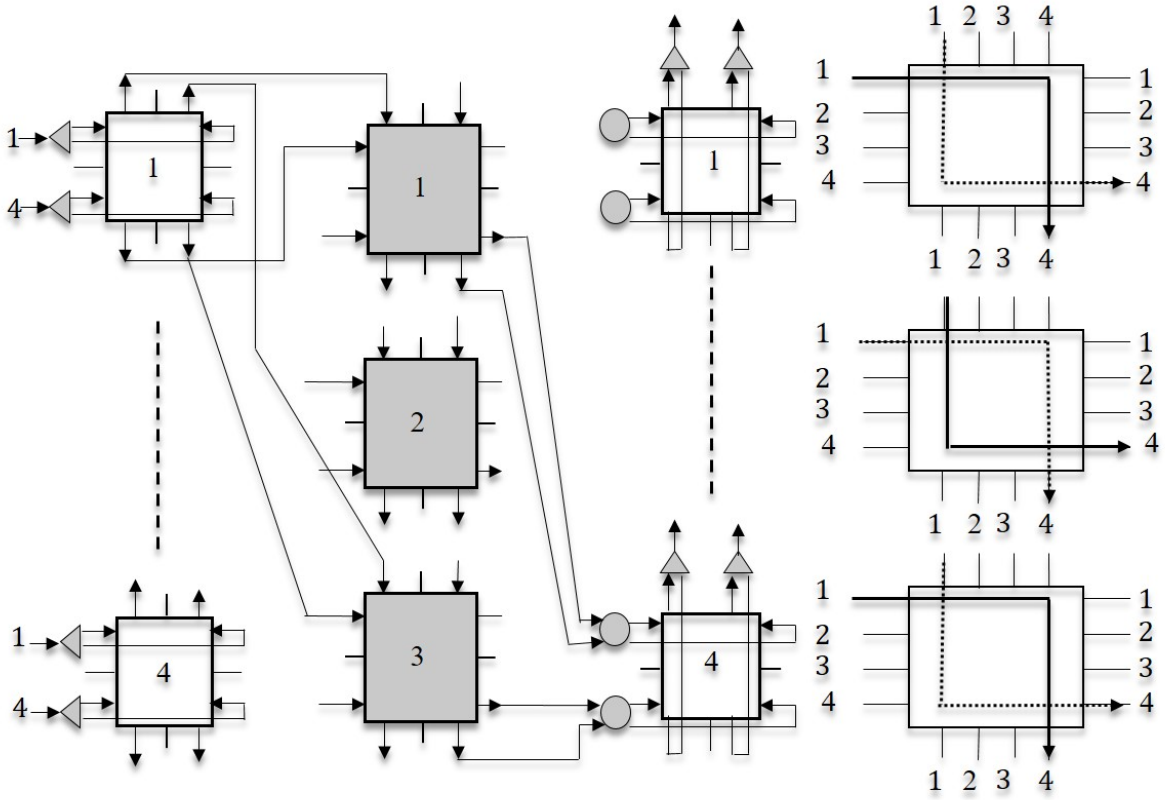


Figure 17: *Routing n calls using $m < n$ middle stage switches*

As can be seen in figure 17, the first $n - 1$ calls can be routed through the first three middle switches. The remaining call can be routed through any of the available routes indicated by dotted lines in any of the three middle stage switches shown on the right side. The above demonstrates that to route $n = 4$ inputs from any first stage switches, a minimum of 3 middle stage switches is required. The conventional TSCN requires a minimum of 4 middle stage switches to route $n = 4$ from each of the first stage XBS. It has also been demonstrated that the row/column resource usage reduces to three quarters. Though the conventional TSCN requires 4 row/column resources, based on the above, we hereby make a proposition that the minimum number m_L of middle stage switch resources required to route n inputs of each of the first stage XBS becomes as follows:

$$m = \left\lceil \frac{3n}{4} \right\rceil \tag{2}$$

for the bidirectional TSCN. The $\lceil x \rceil$ denotes the ceiling function which is to say the value of the least integer greater than or equal to x.

3.3 Proposed Unidirectional TSCN

The proposed structure of the unidirectional TSCN $C_u(n, m, r)$, just like the bidirectional TSCN is composed of three stages. The first stage XBSs are represented by I_x, middle stage switches M_y and third stage switches by O_z with x, y and z as described earlier. The first stage XBS have been modified to have inputs at both the left and the right side. Each input to the XBS is first fed through a 1×2 SE which can divert the input signal to either the left or the right side. This avoids row blocking from occurring in the first stage. The outlets are taken from the top and bottom of the switch. Each top outlet is then fed to the top inlet of the middle stage XBS and the bottom outlet is fed to the left side of the middle XBS switch as shown in Figure 18. At the second stage, each of the $r \times r$ XBS uses the left and bottom side of the switch as inlets and outlets respectively. The extra top inlets and right outlets are used as additional inlets and outlets respectively. The top inlets and right outlets provide additional routes in the XBS without additional cross points. The direction of travel of signals in this switch is from left to right (forming a straight through connection) or (left to bottom forming a rectangular route) and top to bottom or top to right (rectangular route). The general direction of signal travel is one way from top to bottom or left to right and/or top bottom going right or left to right going bottom. This is referred to as unidirectional as was described above in section 3.1 figure 11(a).

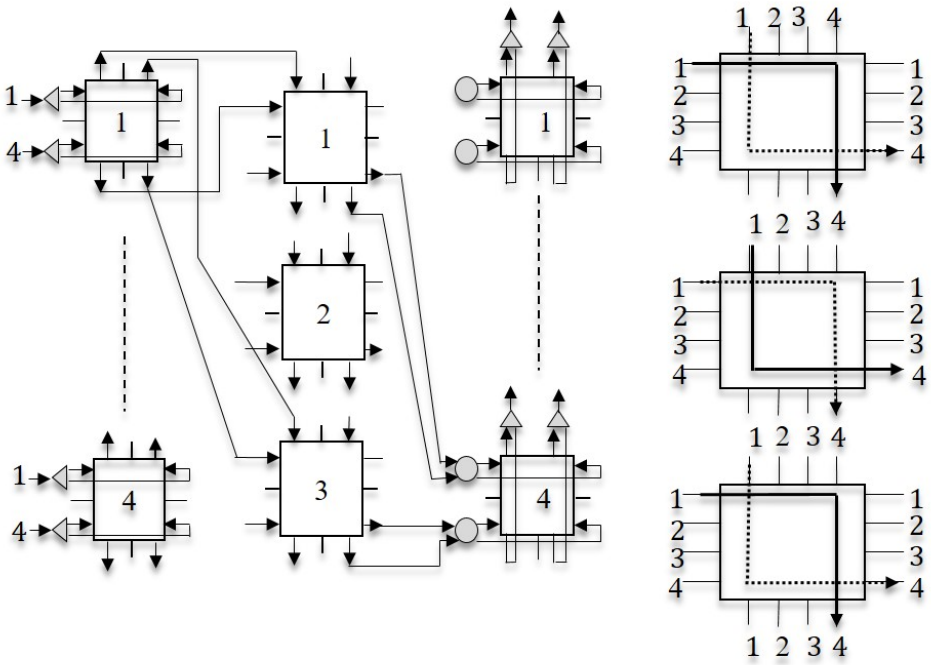


Figure 18: *Proposed Unidirectional TSCN*

The XBS in the third stage are bi-directionally modified with inlets at the left and right sides and outputs at the bottom and top side of the switch. The outlets of the second stage XBSs are fed to the third stage XBS via a 2×2 BSE. These 2×2 BSE allows the two signals to access either sides of the XBS in the third stage. The outputs to the third stage are connected through 2×1 BSEs at the top to select the signal either from the top or from the bottom part of the XBS. Similar to the bidirectional TSCN,

the extra routes provided in the unidirectional switch allows two connection requests to share a row or a column in one XBS. If we assume two connection requests coming from two different switches. If the destination of the request from the left side is equal to the source port of the second request from the top, and provided the destination of the request from the top is less than the source port of the request on the left side, the two requests will share a column. If the destination switch of the request from the top is the same as the source switch of the request from the left, provided the destination of the request on the left side is less than the source switch of the top request, the two requests will share a row. If the destination of the request on the left is the same as the source port, and the output on the left is free, then a straight through connection is made. The straight through case only requires a single row and a single column for the two calls.

Similar to the bi-directional switch, as can be seen on the right side in figure 18, 4 inputs from the first stage XBS can be routed using only 3 middle stage switches. 3 of the inputs will be routed using the solid lines while the remaining fourth connection will be routed through any of the three switches using any of the dotted routes. There is a resource reduction in the number of middle stage switches required to route n inputs to use only $\frac{3n}{4}$ compared to that of $m = n$ in conventional Clos. In a conventional TSCN, when the minimum number of middle stage switches $m = n$, it becomes an RNB network and the number of rearrangements is represented as $r-1$ in a worst case scenario [42].These rearrangements are confined within a pair of middle stage switches [43]. In the bidirectional TSCN with $m_L < n$ (where L stands for lower), every middle stage switch can accommodate up to $\frac{4}{3}r$ connections. Due to additional routes in the modified TSCN, during rearrangement, the call rearrangement is not confined to only a side of a pair of middle stage switches but has options of being routed to the other side of inlets in any other middle stage switch which has free idle ports and internal routes. Therefore, the number or rearrangements in the proposed $C_B(n, m, r)$ is expressed as follows:

$$R_B(n, m_L, r) = \frac{r}{2} - 1 \tag{3}$$

The detailed derivation of equation 3 is given in section 4.2 of summary of simulation results.

3.4 Wide-sense non blocking

As observed above, the sharing of row/column resources reduces the minimum value of m to maintain RNB properties for both unidirectional and bidirectional TSCNs. By increasing m, just like in conventional TSCN, both the WSNB and SNB TSCN types can be realized. In order to maintain the SNB property in the proposed TSCN, the row/column pairing must be followed. Because of this,the minimum m that requires no rearrangement becomes WSNB. Based on the conventional SNB TSCN, the minimum value of m for the proposed WSNB is derived. As was discussed in figure 6, conventional SNB TSCN require a minimum number of middle stage switches to maintain SNB properties given by the equation 4 below;

$$m = 2n - 1 \tag{4}$$

From fig.19 below,it can be seen that $n - 1$ inputs of switch A are connected to the

third stage through $n-1$ middle stage switches. Similarly, $n-1$ outputs of switch B in the third stage are connected through a different set of $n-1$ middle stage switches. In order to route a connection request from the idle port of the first input switch A to the idle output of the third stage output switch B, an extra middle stage switch is necessary for maintaining SNB property.

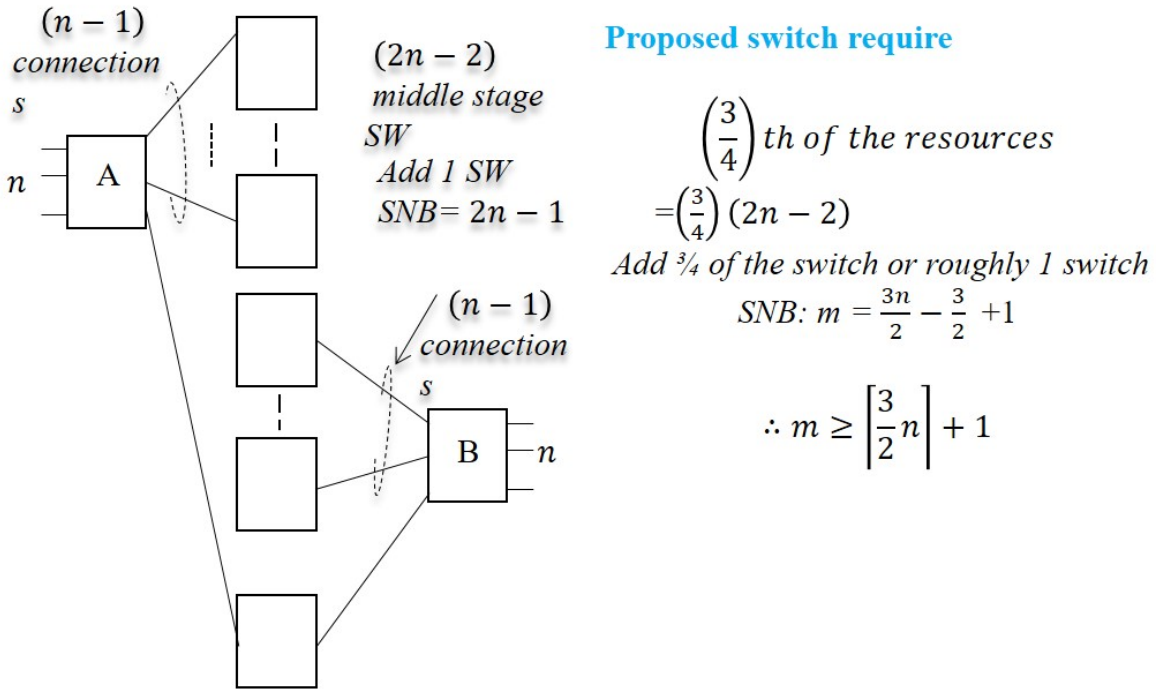


Figure 19: *Derivation of the proposed lower limit*

The proposed structure reduces each set of the middle stage switches to m_L as in eqn. 2. Therefore, to route all the calls, the number of middle stage resources reduces to three quarters. Therefore, we need $\frac{3}{4}$ of the total resources used in a conventional TSCN. Therefore, the WSNB condition for the proposed structure becomes

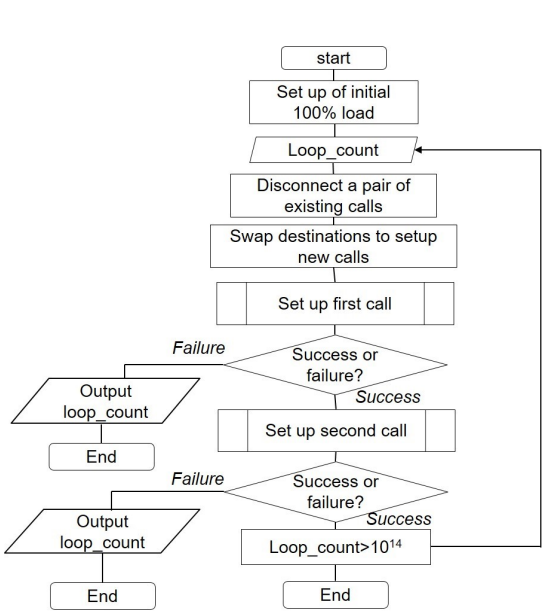
$$m = 2\left\lceil \frac{3}{4}(n-1)+1 \right\rceil => m = \left\lceil \frac{3n}{2}+1 \right\rceil \tag{5}$$

where the equality holds when $n = 2d(d = 1, 2, 3, ...)$. the m_{WSNB} is smaller by about 25% in comparison to the m_{SNB}. The proposed architectures require a certain control algorithm to maintain the row/column sharing of call set up. The details of the algorithm are discussed in the next chapter.

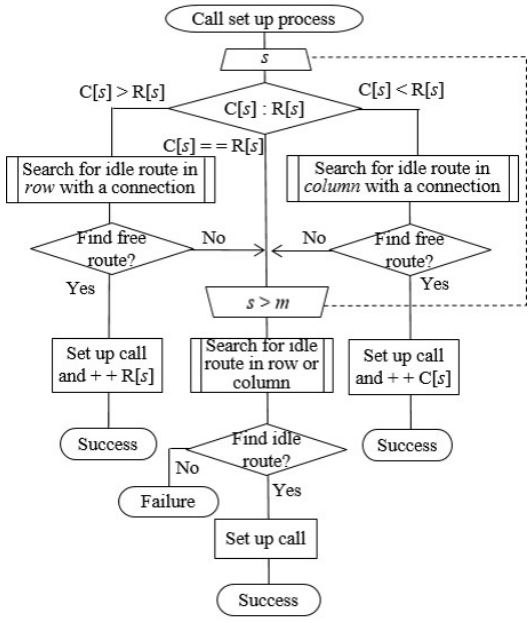
Chapter 4

Simulation setup and experimental results

In order to test the non-blocking performance of the proposed TSCN switches, simulations were run using c programming. The methodology employed in this study involves building simulation models of the proposed switch using C programming. The non-blocking performance of bidirectional TSCN through simulation of call connection and disconnection is examined. The number of middle stage switches that determine the non-blocking properties was observed in relation to blocking that occurred in the simulation. Since no blocking can occur in first and third stages as was described in section 3.2.1, the simulation focused on the connection requests arriving at the middle stage switches. Two dimensional data arrays were used in representing the connections at every inlet as $I[s][k]$ and outlet as $O[s][k]$ of each XBS. The first dimension represents the XBS number of middle stage switches s, $(1 \leq s \leq m)$, while the second dimension represents the inlet or outlet number of the XBS k, $(1 \leq k \leq r)$. The second dimension was set to twice the number of inlets and outlets meaning that for each XBS, there are twice the number of inlets coming from the first stage switches and twice the number of outlets going to the third stage switches. This was implemented by using four two dimensional arrays, two for the inlet side of the switch, $I1$ and $I2$ and two for the outlet side of the same switch, $O1$ and $O2$. The details of this simulation implementation are described in the next section and are summarized in the flow chart of figure 20.



(a) *Flow chart of the simulation process*



(b) *Detailed flow chart of the simulation process*

Figure 20: *Flow chart of the simulation process*

4.1 Simulation flow in Bidirectional TSCN

The simulation was performed in Microsoft Visio studio 2010 IDE. The parameters for the bidirectional switch were set as follows; The inlets on the left side of the XBS were represented by the array $I1[s][k]$ and the right side inlets as $I2[s][k]$. The bottom outlets as $O1[s][k]$ and the upper outlets as $O2[s][k]$. This is represented in figure 21 below.

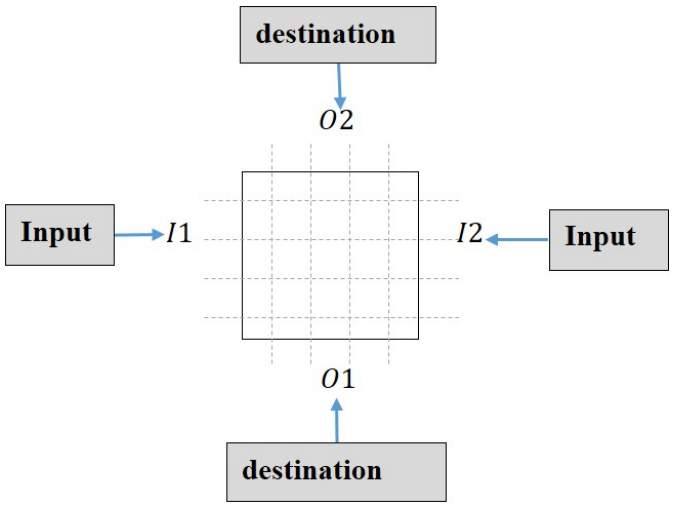


Figure 21: *Simulation of the XBS in C programing*

A connection request was represented by a source port Pt1 to a destination port dest1. Similarly, a second source port as Pt2 and destination port dest2. The number of input ports per switch was set as n which was set equal to the value of r and the number of middle stage switches set as SW. These parameters were manually set before the simulation was compiled and run. The value n was set to an even number for simulation purposes.

(i) Initialization

As shown in the flow chart of figure 20, the procedure starts with initialization of all arrays to show that no initial connections exist in the switch. This was done by setting the idle state of all the inlets and outlets of the XBS to a value of -1. In the simulation, the value -1 means no connection exists at a specific inlet or outlet of the switch, for example, as shown in figure 22, all the ports are empty.

(ii) Call setup

Next, an initial 100% call setup was performed by assigning the inlets of the XBSs with the destination outlets numbers and also linking the outlets with the source inlets. i.e. to say the content of the outlets $O1[s][dest1]$ are set to the inlet port 1 and the contents of the inlets $I1[s][pt1]$ are set to the destination dest1. In this simulation, a 100% load call setup was assumed. This is because the compact connection pattern as discussed earlier in 3.2.1 is suitable for imposing a worst case scenario. The connection requests were set up so as to maximize the row/column sharing as shown in figure 23. Note that only $n/2$ middle switches were used for this.

(iii) Disconnection

Next, an arbitrary pair of connections was selected and disconnected. The random function $rand()$ in c programming was used for this. This was done by first selecting a random switch ($w = rand()\%SW$) from which to make the first disconnection, then a random side of that switch ($d = rand()\%2$) to disconnect the call from and then the actual call

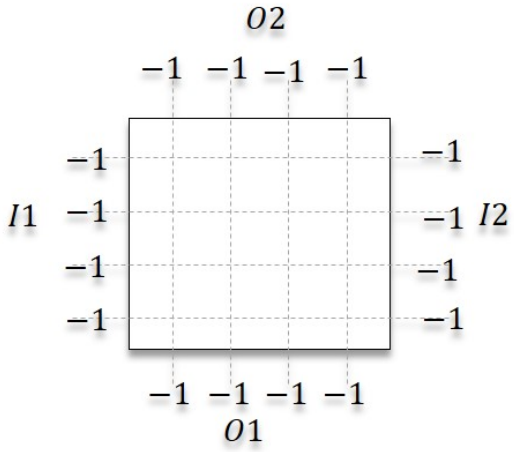


Figure 22: *Initializing the XBS to idle state*

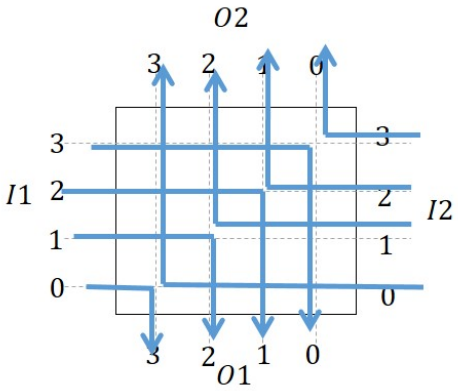


Figure 23: *Simulation of the XBS in C programing*

to disconnect ($t = rand()\%N$), where w is the randomly selected switch, d is the side limited to two sides only and t is the actual connection to disconnect. If a random side $d = 0$ was selected, it meant that the disconnection was performed on the left side of the switch. The content of $I1[w][t]$ must be assigned to $dest1$ as $I1[w][t] = dest1$ with input port as $Pt1 = t$ and side 1 ($sd1 = d = 0$). The corresponding inlets and outlets must be reset to a value of -1 to represent that a disconnection has been performed i.e. $I1[w][pt1] = -1$ and $O1[w][dest1] = -1$ as shown in figure 24. This procedure was repeated for the second disconnection assigning the respective values accordingly, with $sd2 = 1$, $Pt2 = t$ and $dest2 = I2[w][pt2]$. In the second disconnection, a random side $sd = 1$ corresponding to a right side was generated.

(iv) Reconnection

After disconnections, the destinations were swapped and an attempt to reconnect the two new call requests was made one after the other. The algorithm began first by searching for an available route for the first request and if it successfully found the route, it sets it up and went on to search for the available route for the second request. The algorithm searches for the route for the second request in the same manner as the first. Once successful, the second request completes and the procedure repeats for a different

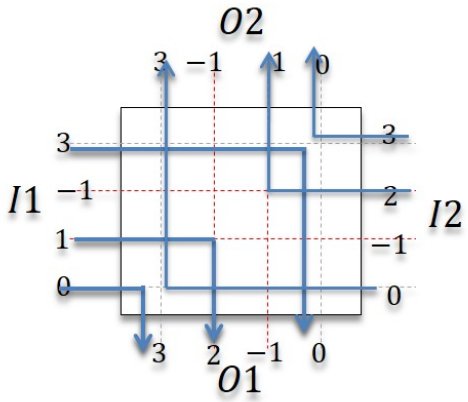


Figure 24: *Disconnection of a random pair in an XBS*

random pair. Several route conditions to avoid blocking were set carefully. During call setup, the highest priority in the bidirectional switch was to search for routes that sought to pair calls in either rows or columns so as to use the switch resources efficiently. Two parameters were defined $C[s]$ and $R[s]$ to represent the number of pairs formed in columns and row directions respectively in the s-th XBS. An idle route is sought in sequence. At the middle stage switch s, $C[s]$ and $R[s]$ are compared. If $R[s] > C[s]$, an idle route is sought in the column direction to achieve a balance between $R[s]$ and $C[s]$. If $C[s] > R[s]$, an idle route is sought in the row direction to increase $R[s]$. If $C[s] = R[s]$, the algorithm skips s to $s + 1$. If the routes cannot be found in which a row or a column pair cannot be formed, then and only then should a search for other available routes with no row/column pair condition can be performed. These conditions are used in searching for routes on both the left and the right sides.If there is a row occupied and the connection request's destination does not interfere or cause blocking with an existing call in the row, then set up the call through this row and form a row pair. This is illustrated in figure 25.

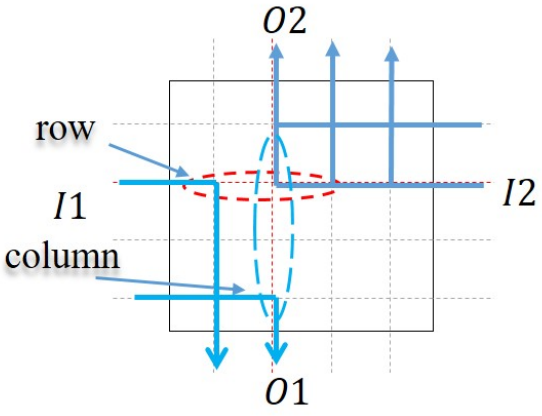


Figure 25: *Initializing the XBS to idle state*

If a row pair cannot be achieved, search for a column pair in a similar manner. If there are no free rows or columns to form row/column pairing, then the call is set up using a default search, which does not necessarily search for row/column pairs. If all the conditions to search for the routes are not met, it was concluded that no route is found

in the switch and blocking happened and the route setup procedure ends in a failure and it displays the loop at which blocking occurred. Once blocking is detected, the number of middle stage switches was increased and the procedure repeated. The detailed path search for setting up the first and second calls is summarized in a more detailed flow chart of figure 20(b). Although the path search algorithm requires several runs through a search for an available route through each middle switch, the complexity for setting up a connection is the same as that of a conventional SNB TSCN of $O(mr)$. In the simulation, m was varied for several pairs of fixed n and r to check the point at which blocking in the bidirectional switch stopped occurring. The maximum number of loops used was set to 1×10^{14} for each setting of n. This value is sufficient to test all the necessary call attempts for non-blocking because it was observed that blocking for the conventional TSCN $C(n, m, r)$ was detected within the loop-counts of 1×10^{10} as shown in figure 26. The dotted lines in the figure represents the point at which blocking was no longer detected in the corresponding switch. Figure 26 shows the loop count at which blocking was detected vs number of middle stage switches. The vertical dotted lines correspond to a point at which blocking stopped in the respective curves representing different switch sizes.m was increased further until no blocking occurred for a maximum number of loops.

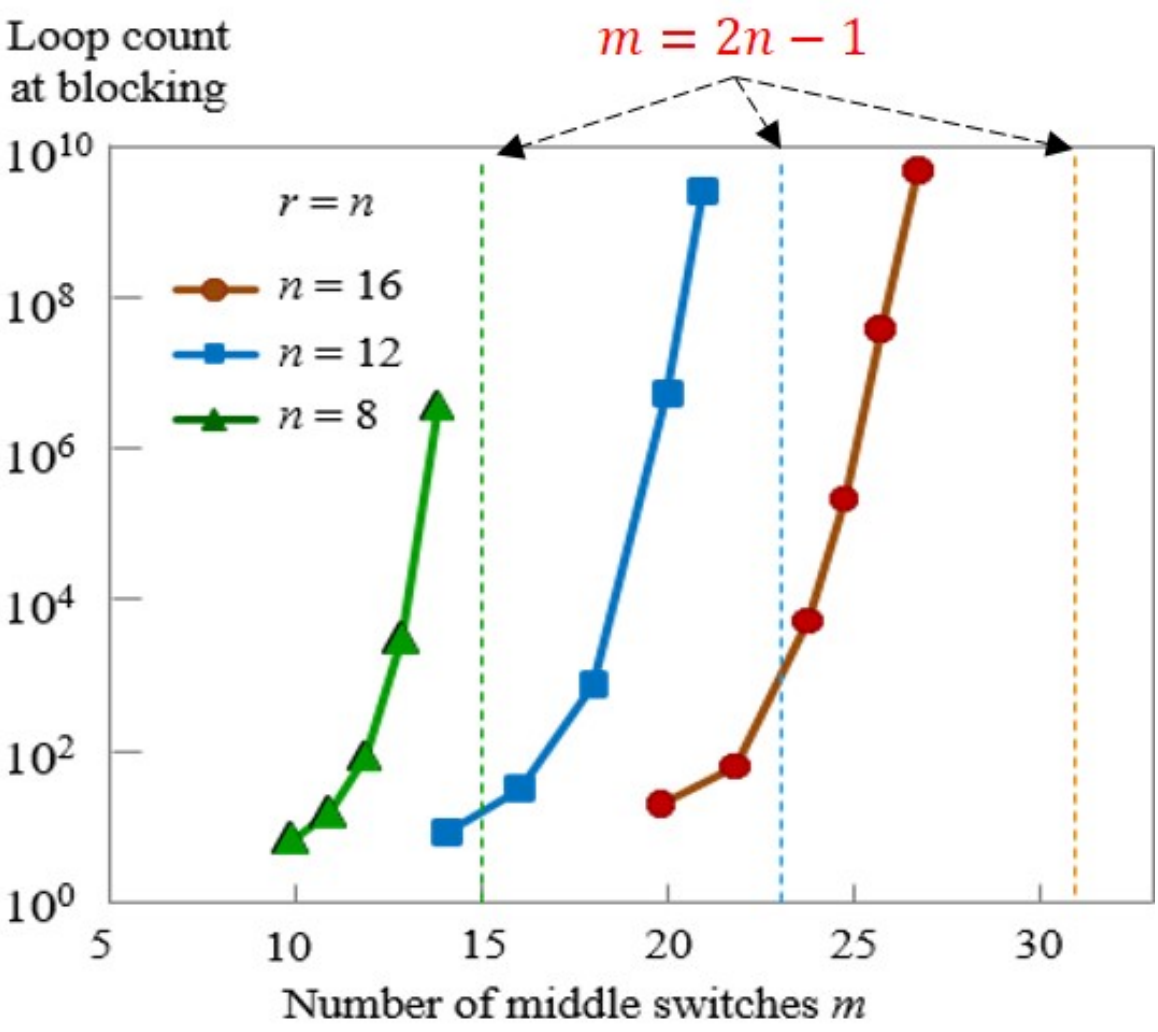


Figure 26: *Simulation results of the conventional Clos network*

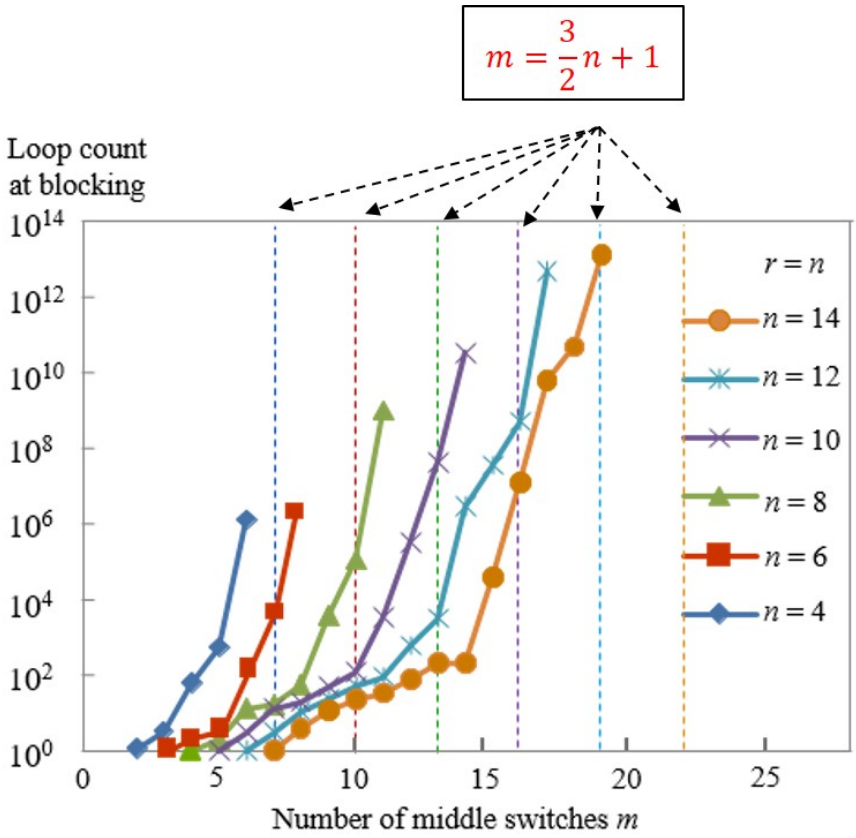


Figure 27: *Simulation results of the proposed architecture*

4.2 Simulation and experimental results

The results in figures 26 and 27 shows a summary of the simulations performed. As can be seen in figure 26, blocking in the conventional TSCN occurred all the way until $m = 2n - 2$. When the number of middle stage switches reached $2n - 1$, the dotted lines, blocking ceased. For example, when $n = 8$, blocking occurred up until $m = 14$. However, at $m = 15$, no blocking occurred. In the proposed TSCN, blocking continued to occur up until the number of middle stage switches m was $\left\lceil \frac{3}{2}n \right\rceil$. Blocking stopped when the number of middle stage switches reached $\left\lceil \frac{3}{2}n+1 \right\rceil$. For example, when $n = 6$ and $m = 10$, no blocking occurred. However, blocking occurred at all points lower than $m = 10$. Each dotted line corresponds to a point at which blocking ceased for the respective values of n. In summary, equation 5 holds as a test for the WSNB condition. Equation 2 was also indirectly verified because equation 5 was derived from equation 2 which was based on the row/column sharing scheme. The performance of the conventional, bidirectional TSCN is summarized in the table 2 with respect to m, the number of rearrangements, total number of cross points and the differential number of cross points to $c(n, m, r)$. The number of cross points are calculated in an $n \times m$ XBS as nm. The number of crosspoints in the first, second and third stages in $C(n, m, r)$ are calculated as n^2r, r^2n and n^2r respectively. The resulting total number of cross points in $C(n, m, r)$ is $2n^2r + r^2n$ which is $N(2n + r)$ as shown in table 2, here $N = nr$. Several values in the table 2 were approximated in order to clearly observe the difference in performance

at a glance. There is a clear performance trade off relationship between the number of rearrangements and the total number of crosspoints. For example, the conventional $C(n, m, r)$ has $r - 1$ rearrangements but has a minimum number of crosspoints, while the special case of the $C_B(n, m, r)$ and $C_U(n, m, r)$ only has a single rearrangement at an increased crosspoint count. The bidirectional $C_B(n, m, r)$ and unidirectional $C_U(n, m, r)$ achieves no rearrangements at an increased crosspoint count at the first and third stages while they have a smaller number of crosspoints in the middle stage switches. It is clear from the graph in figure 27 that the number of middle stage switches are reduced by 25% in the proposed TSCN in comparison to those in the conventional TSCN due to row/column sharing. A new lower bound of $m = 3n/4$ for maintaining rearrangeably non-blocking property capability has been proposed. This is much smaller than the conventional TSCN.

In terms of rearrangements, consider a fully loaded conventional RNB TSCN as shown in figure 28. If a pair of calls in M_{j1} and M_{j2} disconnect as shown in dotted lines, and an attempt to establish a new connection from I_r to O_k is made as indicated in blue line, blocking occurs. It is therefore necessary to perform some rearrangements in order

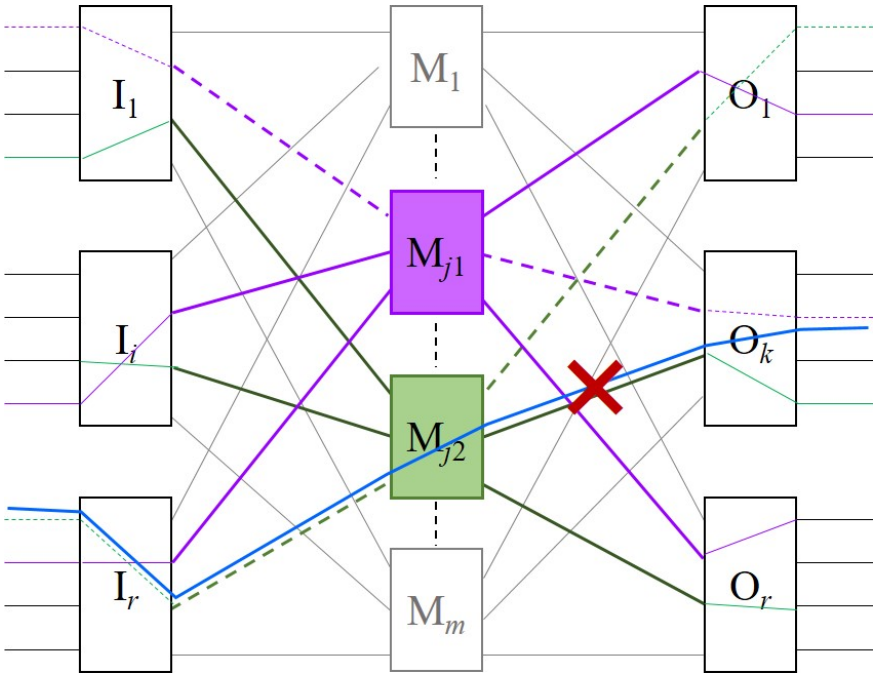


Figure 28: *Worst case blocking and need to rearrangement*

to provide a route for this connection. These rearrangements are restricted to the two middle stage switches. The conventional XBSs only use a single side as inputs. The rearrangement can be performed by sending the call to be rerouted to the other switch and if blocking occurs in the second switch, then the existing blocked call is rerouted to the initial switch and the procedure repeats until all connections are established. As can be seen in figure 29, the call tries to set up (in red) in sw1, the existing call is sent to sw2 in which blocking occurs with another call with a similar input. The call is sent to the other switch and there is no blocking that occurs. We see that the total number of connections are given by $2r - 1$ and the number of rearrangements is less than or equal to the number of existing calls and is given by the equations with the figures below. The total number of connections is equal to $2r - 1$ Therefore, the number of rearrangements

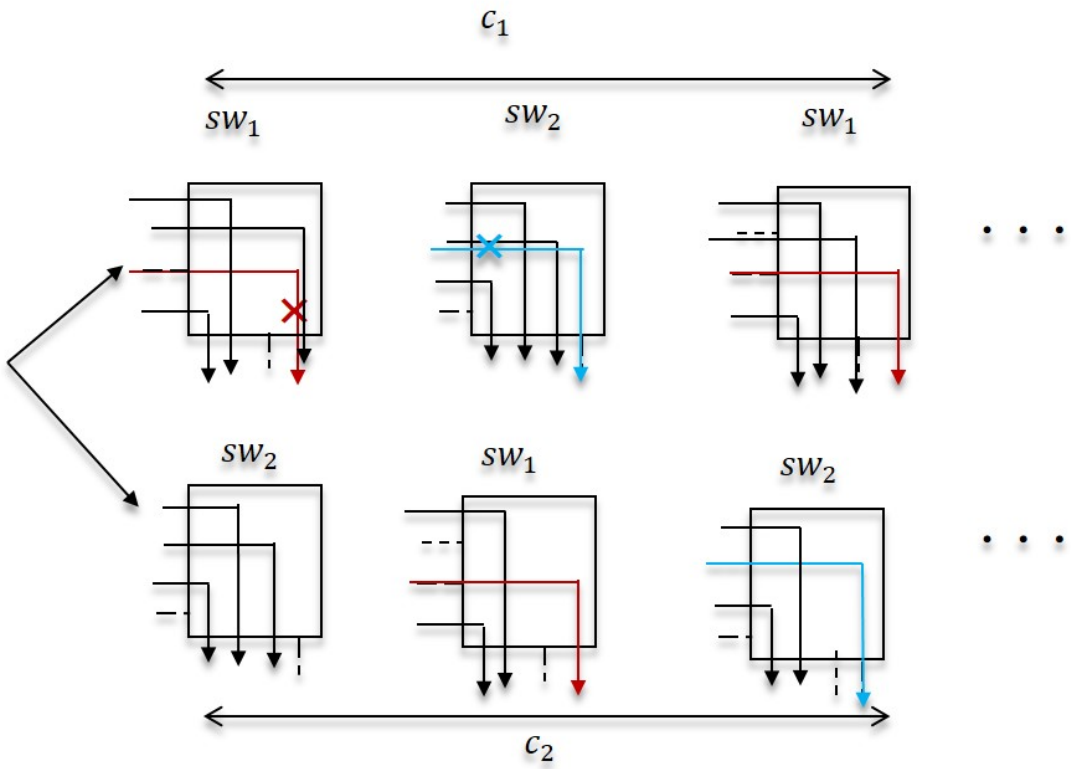


Figure 29: *Rearrangement process in conventional TSCN*

needed becomes

$$\begin{aligned} C_1 + C_2 &\leq 2r - 1 \\ 2\min(c_1, c_2) &\leq 2r - 1 \\ \min(c_1, c_2) &\leq r - 1 \end{aligned} \tag{6}$$

In the proposed architecture, the modified XBSs have extra inputs on the right (in the bidirectional) and at the top (in the unidirectional) SW. So during a rearrangement, calls can begin rearranging within the first switch before they are taken to the second switch. For example in a bidirectional sw, if blocking happens on the left side in one switch, existing connections can be rerouted to the right side of the same switch. And if blocking continues, the existing blocked call can be taken to the second switch which also has two sides. This procedure is repeated until all the calls are rearranged and reconnected successfully. This procedure is represented in figure 30.
As can be seen in figure 30, the total number of rearrangements can be derived and becomes;

$$\begin{aligned} c_1 + c_2 + c_3 + c_4 &\leq 2r - 1 \\ = 4\min(c_1, c_2, c_3, c_4) &\leq 2r - 1 \\ = \min(c_1, c_2, c_3, c_4) &\leq r/2 - 1/4 \\ \cong \min(c_1, c_2, c_3, c_4) &\leq r/2 - 1 \end{aligned} \tag{7}$$

Figure 31 show the number of rearrangements in the conventional and proposed method. Based on the column and row sharing of the method, the proposed architecture has a reduced number of rearrangements. As can be seen in the orange curve, the reduction

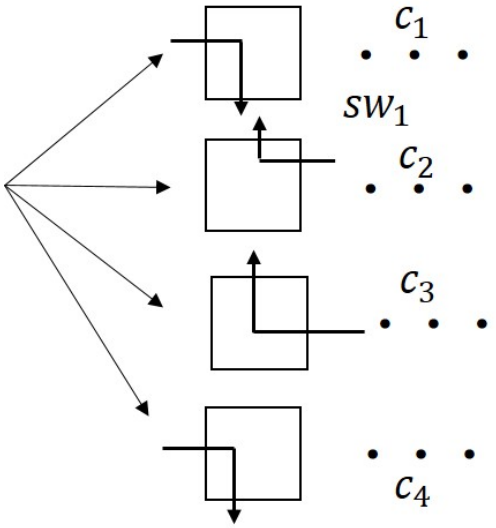


Figure 30: *Rearrangement process in proposed architecture*

in the number of rearrangements is about 50% compared to those in the conventional architecture.

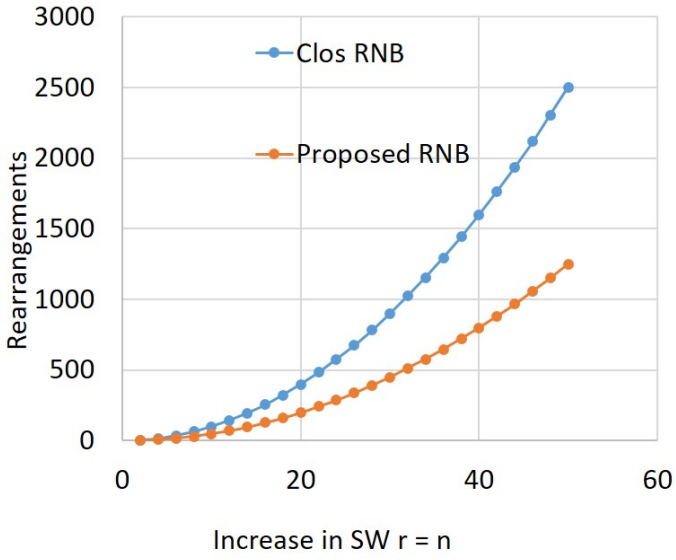


Figure 31: *comparison of the number of rearrangements in conventional and proposed architecture*

Table 2 shows a summary of the performance of the conventional three-stage Clos and the proposed architecture. The table compares the number of middle stage switches required in the SNB, RNB conventional and proposed three-stage Clos network. The table also shows the number of rearrangements in the RNB switches of the proposed and three stage architecture. A generalisation of the total cross points count and a reduction in the crosspoint counts in the SNB and RNB and also a shown.
The graph in figure 32 shows the crosspoint comparison of the conventional three stage Clos network and the proposed architecture. The decrease in the crosspoint count be-

Table 2: Summary of switch performances for conventional and proposed unidirectional and bidirectional TSCN

Type of TSCs	**Middle stage SWs**	**Re - arrange ments**	**Total cross points**	**Reduction in cross points from conventional TSCN**
Clos SNB	$2n-1$	0	$2N(n+r)$	0
Prop. Clos SNB	$\lceil \frac{3}{2}n \rceil + 1$	0	$N\left(2n+\frac{3}{2}r+3\right)+r^2$	$\frac{r}{2}N-3N+r^2$
Prop. Clos SNB With $n \times 2n$	$\lceil \frac{3}{2}n \rceil + 1$	0	$N\left(4n+\frac{3}{2}r\right)+r^2$	$\frac{Nr}{2}-2Nn-r^2$
Clos RNB	n	$r-1$	$N(2n+r)$	0
Prop. RNB	$\frac{3}{4}n$	$\frac{r}{2}-1$	$N\left(2n+\frac{3}{4}r+3\right)$	$N\left(\frac{r}{4}-3\right)$
Prop. RNB	n	1	$N(4n+r)$	$-2Nr$
Prop. Clos RNB With $n \times 2n$	n	1	$N\left(2n+r+3\right)$	$-3N$

comes significant as the switch size increases. At lower values of switch size n, the crosspoint reduction is minimal but as the switch size increases, it saturates at 25%. Figure 33 shows a percentage change in the resource requirements for the proposed architecture. For a switch size at $(\log n = 10)$ on the x-axis, the corresponding percentage change about 10%. At $(\log n = 1000)$ the difference saturates at 25%. As a percentage, the required number of middle stages required in the proposed architecture shows a decrease that saturates at 25%. When the switch size is smaller, the number of middle stage required in the proposed architecture are more and thereby showing a negative reduction. As the switch size increases, the difference becomes more pronounced and it stabilises at one-quarter.

4.3 Summary

The major results of this chapter is that the number of middle stage switches required to maintain the non blocking properties is drastically reduced. This is brought about by the use of idle ports in the proposed architecture. As a result, cross points are reduced drastically by about 25%. It is also observed that for rearrangeably non blocking switches, the number of rearrangements are reduced to 50%.

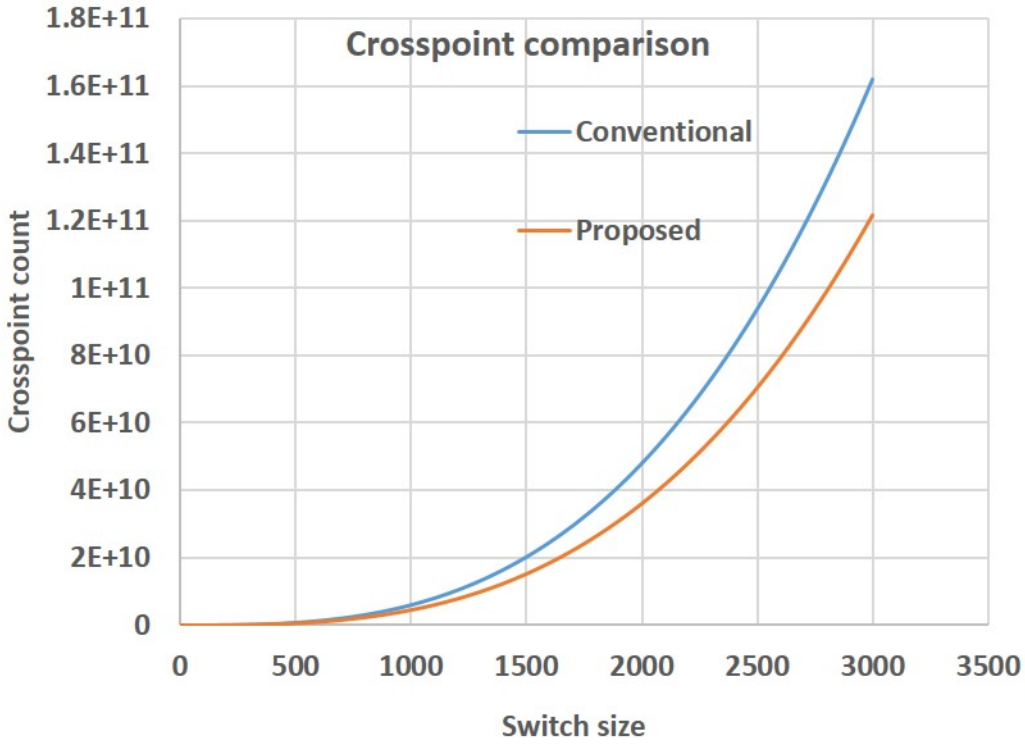


Figure 32: *Cross point comparison in conventional and proposed architecture*

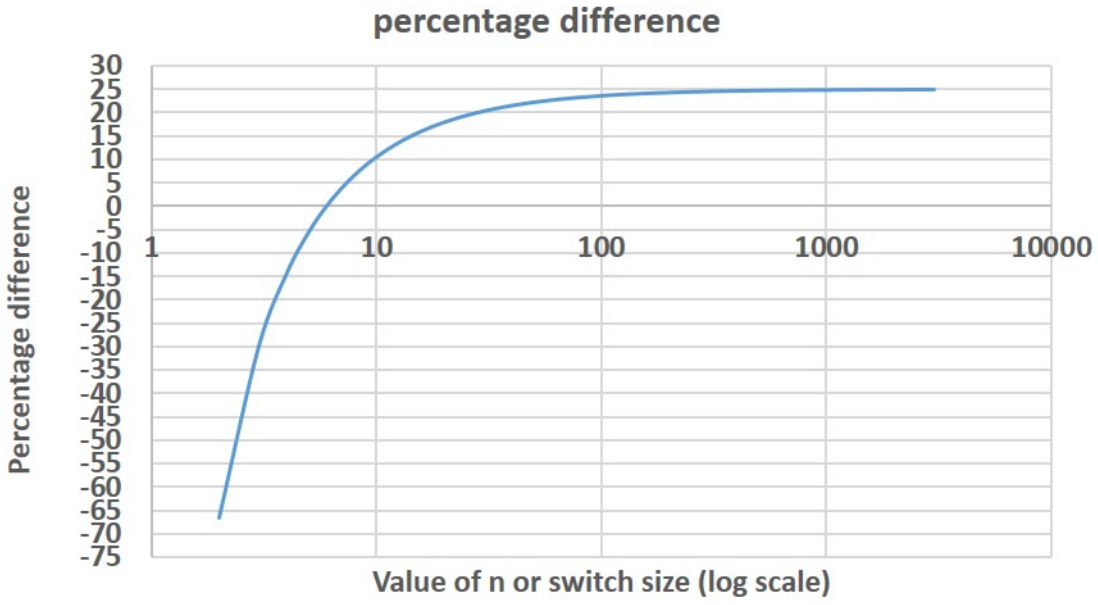


Figure 33: *Overall percentage reduction in the number of middle stage switches required*

Chapter 5

Structure and non-blocking properties bidirectional two stage switches

5.1 Brief introduction

Space-division multiplexing technology for scaling optical network capacity has received serious attention recently [44]. Multistage space switch networks have become a key component in creating high-port-count optical interconnects and cross-connects[45, 46]. While the three-stage Clos architecture remains to be a well-established and highly practical design principle for scalable space switches [47], two stage networks (TSNs) are emerging as a new design option for relatively small-capacity switches[48]. TSNs are classified into two, folded and unfolded types. The folded is equivalent to a three stage Clos network and its structure and non-blocking properties are well known [43]. On the other hand the unfolded TSNs (UTSNs) are completely understood. A few types of UTSNs exists all of which are RNB [21]. In this study, for the first time, a strictly non-blocking UTSN is considered. A new design principle of UTSN which consists of input and output switch modules (ISMs and OSMs) with bidirectional switching capabilities and represented by $B(n, m, r)$ where n,m and r are denote the number of input ports to the ISM, the OSMs and the number of ISMs respectively, is firstly presented. Secondly,the maximum number of rearrangements and minimum value of m to satisfy the SNB condition is formulated. Finally the switch hardware complexity is estimated.

5.2 Proposed structure of UTSN

The proposed UTSN $B(n, m, r)$ with r ISMs and m OSMs each denoted by I_p, $1 \leq p \leq r$ and O_q, $1 \leq q \leq m$ is shown in figure 33.
All the ISMs have n inputs and a the total number of inputs is given by $N = Nr$, while every OSM has N outputs, of which the first and second halves are provided at the top and bottom edges. every pair of an ISM and an OSM is interconnected with a pair of internal links, i.e.each link between and outlet on the top of the ISM and an inlet on the left of left side of the OSM. The other link between the bottom of the ISM and the right side of the OSM. An ISM can be implemented by an $n \times m$ bidirectional crossbar switch(BXS) and $n \times 2$ switches as shown in figure 34.
Every input signal to the ISM may be switched to its outlet at either end of the column. Let (i, k) be a connection between an input i, $1 \leq i \leq n$, and an outlet k, $1 \leq k \leq 2m$, in the ISM. Similarly, let (l, j) be a connection between an inlet l, $1 \leq l \leq 2r$, and an outlet j, $1 \leq j \leq N$,in the OSM, which can be implemented by an $r \times N/2$ BXS, as shown in Figure 35.

Every jth outlet in the OSM is coupled in the jth output via a passive coupler, which is shown as a dashed triangle in Figure 33. The passive coupler can be substituted by an $m \times 1$ switch at the cost of extra cross-points. Here, only one outlet of all the m OSMs

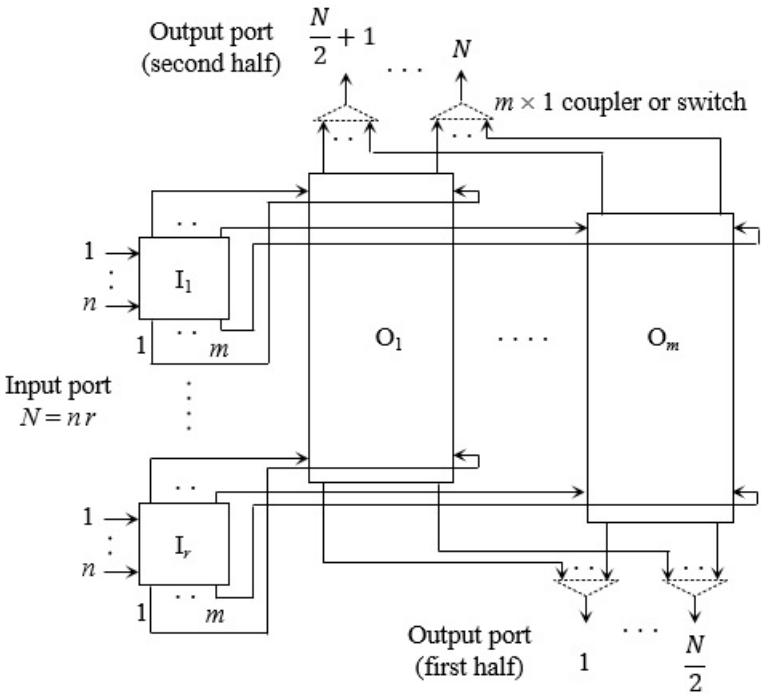


Figure 34: *Structure of* $B(n, m, r)$

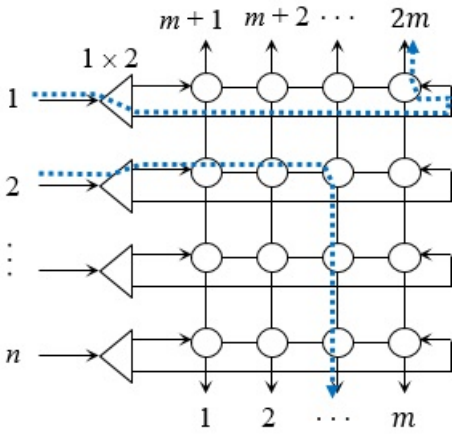


Figure 35: *Design Example of ISM using bidirectional switching*

may have an output signal at a time with the others remaining idle. As can be noted in the figure 34, every input signal is routed to either a left or right inlet through a 1×2 switch, every row of ISMs takes only a single at most. It is clearly shown that every column of ISMs and OSMs is shared by at most by two signals, i.e. one signal headed for the top and the other down for the bottom.

5.3 Non-blocking properties of $B(n, m, r)$

Even though the main objective here is to derive the SNB condition for $B(n, m, r)$, the reaarangement process for $B(n, m, r)$ which provides insights into the SNB condition is first investigated. As was observed from figure 34, no blocking occurs in the ISMs

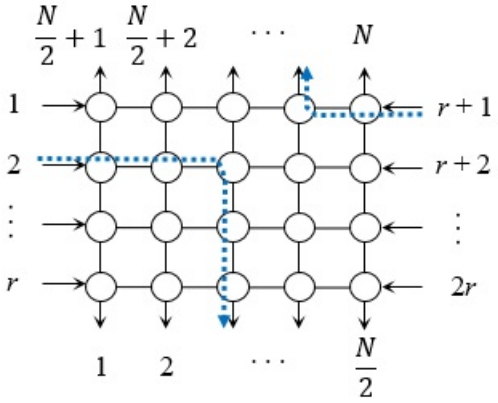


Figure 36: *Design Example of OSM using bidirectional switching*

because the ISMs functions as incomplete $n \times 2m$ switch. Blocking can occur in the OSM in the following worst case scenario: assume that $n-1$ inputs from an ISM I_{p1}, $1 \leq p1 \leq r$ are already connected with $n-1$ outputs. Also assume that these $n-1$ inputs enter the OSMs on their left side inlets. Then the last connection request in ISM I_{p1}, which is denoted by the dashed line in figure 36 is issued.

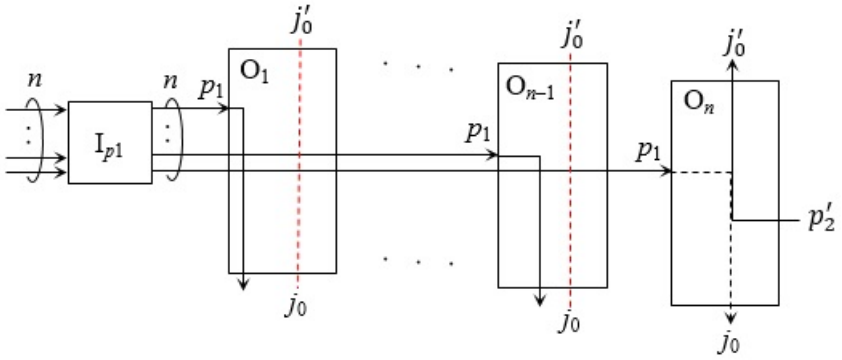


Figure 37: *Worst-case scenario for rearrangement in* $B(n, n, r)$

Here assume that the request has a destination output of j_0, $1 \leq p_2 \leq N/2$, while $j_0 = j_0 + N/2$ corresponds to I_{p2}, $1 \leq P_2 \leq r$ as shown in figure 36 with an assumption that $p^{'} = p_2 + r$ and all of the n inputs to the I_{p2} are already in use. Note that the primes mark denotes the inlets on the right side and the outlets on the top edge. Blocking occurs in the j_0th column in O_n. It can be seen that every j_0 th column of OSMs except O_n, is idle. If the new connection (p_1, j_0) is rerouted to the $j0$ th column in the OSMs O_b, $1 \leq b \leq n-1$, which is connected to I_{p_1}, an existing connection denoted by (p_1, j_1) in O_b should be moved to O_n, only where inlet p_1 is idle. However,this rearrangement can cause blocking if the j_1 th column in O_n is already used. Note that $j_1 = j_0$ holds because the blocking in the j_0th column has already been addressed. In other words, each existing connection in O_n experiences a rearrangement only once. Because there are r–1 existing connections in O_n, the rearrangement process will last r–1 times at most between O_n and O_b. Let R_1 be the number of rearrangements in this case.

Now, consider another rearrangement process that begins with p_2 in O_n. If the new connection (p_1, j_0) is provided in O_n, the existing connection (p_2, j_0) should be rerouted to the j_0 th column in O_b. However, blocking will occur in the OSM because there is

another existing connection (p_2, j_2) or (p_2, j_2), with $j_2 \neq j_0$ and $j_2 \neq j_0$, which should be moved to O_n, only where inlets p_2 and p_2 are idle. The second rearrangement process also lasted r–1 times at most. Let $R2$ be the number of rearrangements in this case. because the connections involved in R_1 and R_2 are different from each other, the following relation holds:

$$R_1 + R_2 \leq r - 1 \tag{8}$$

Both R_1 and R_2 are integers and the minimum number of rearrangements is expressed as :

$$\min(R_1, R_2) \leq \lfloor (r-1)/2 \rfloor \tag{9}$$

where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. Note that O_b was fixed in the above discussion. By examining the number of rearrangements for every O_b, the minimum number of rearrangements at large, denoted as R_0, can be derived as follows in a similar manner to the above derived equation (9) as:

$$R_0 \leq \Big\lfloor \frac{r-1}{2(n-1)} \Big\rfloor \tag{10}$$

where $n \geq r$, there is more freedom to exchange the blocked connections (p_1, J_0) with other existing connections. As it has been shown in figure 36, I_{p1} and I_{p2} have $n-1$ and n connections under the worst case scenario. These assumptions allows for a choice of an outlet, which is not included in the r–1 existing connections in O_n, out of the n existing connections in I_{p2}. Accordingly, when the connection (p_2, j_0) can be moved to an appropriate OSM, of which the j_3 th column is occupied by an existing connection, that is, (p_2, j_3) or (p_2, j_3), whereas the j_3th column in O_n is idle. As a result, the number of rearrangements is reduced to two at most.

Based on the above discussions, an SNB condition can be readily derived as follows: When $m = n$, the blocked connection needs to be moved to another OSM, where blocking can occur. However, if m is set such that $m = n+1$, the last connection request may be provided in the $(n+1)$-th OSM, where both p_1 and j_0 are idle. Consequently, the SNB condition for $B(n, m, r)$ becomes

$$m \geq n + 1 \tag{11}$$

5.4 Hardware complexity of $B(n, m, r)$

The total number of cross-points becomes large when $m = n + 1$. It is expressed as a function of n as follows:

$$C_p(n) = \frac{N^2}{2}\Big(1 + \frac{1}{n}\Big) + N\,(n+2) \tag{12}$$

where the first and second terms in equation (12) corresponds to cross-points of the OSMs and ISMs, respectively. $C_p(n)$ is minimum at $n_{opt} = \sqrt{N/2}$ as follows:

$$C_p(n_{opt}) = \frac{N^2}{2} + N\Big(\sqrt{2N} + 2\Big) \tag{13}$$

$C_p(n_{opt})$ converges to $N^2/2$ as $N \to \infty$. Some RBN UTSNs also have $N^2/2$ cross-points[21]. This bidirectional UTSN has achieved SNB properties with approximately the same cross-points as RNB UTSNs.

5.5 Summary

The new design principle for UTSNs using bidirectional switches has been unveiled herein. The non-blocking properties have been investigated starting from the RNB properties. The number of maximum arrangements have been studied from which the number of OSMs required to obtain SNB have been derived. The hardware complexity has also been investigated and it has been shown that the hardware complexity becomes minimal at $n = \sqrt{N/2}$ and saturates at $N^2/2$ as $N \to \infty$. The two stage switch obtained in this chapter can be applied to the first and third stages of the proposed bidirectional three stage network shown in figure 12 in Chapter 3. The obtained Clos network becomes a 5 stage Clos network. The proposed TSCN can be scaled to increase the number of stages by recursively replacing the first and last stages with the UTSN.In this case, the number of stages will be scaled as $2k-1$ where k is an integer starting from $k = (2, 3, 4, ...)$. This is illustrated in figure 37 below. The cross-points of the architectures obtained have not been discussed here and have been left for future studies.

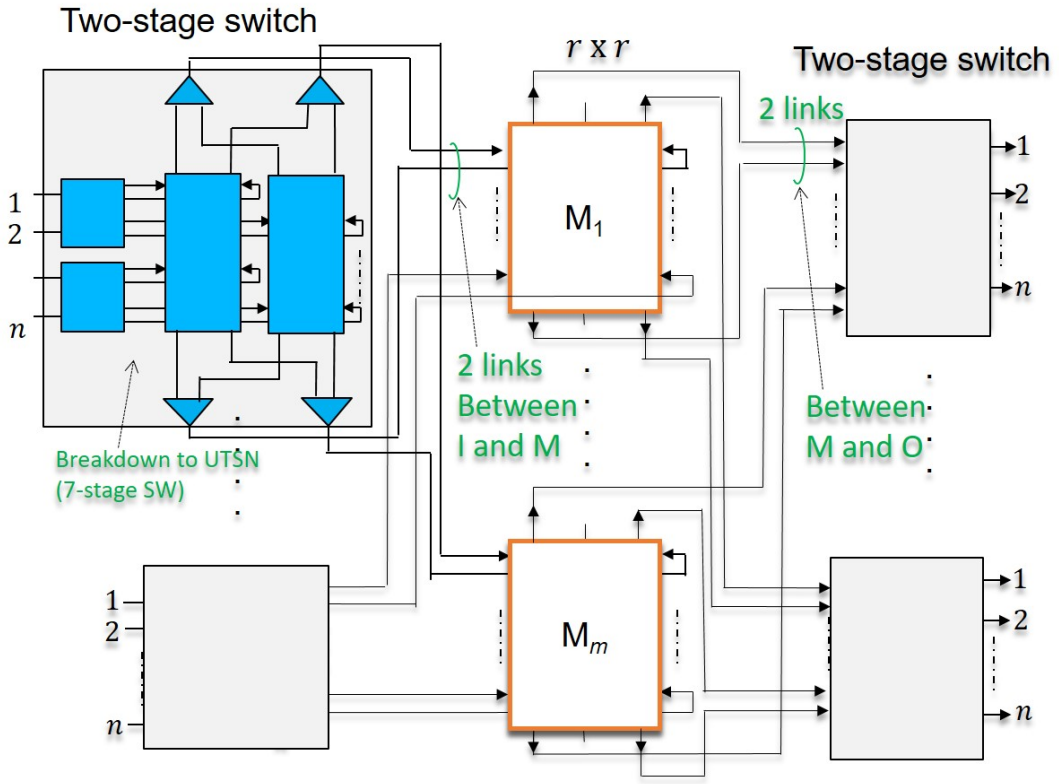


Figure 38: *Modified TSCN with first and third stages replaced by two-stage switches*

Chapter 6

Design of parallel control algorithm

In setting up paths in RNB switching networks, one of the challenges has been to find and set paths that are non-conflicting with any inputs and outputs in the switch. Owing to the increased demand in the amount of traffic in communication networks, Control algorithms that seek to set up non-conflicting paths at a fast rate are required in modern switches. Control algorithms vary depending on the type of connection to be established. These types could be unicast, multicast or broadcast. A unicast connection is when one input terminal connects to one output terminal. A multicast connection is a connection from one input terminal to several output terminals. Broadcast involves an input terminal sending messages to all or most of the output terminals. In the past few decades, several switching algorithms have been developed. These include the sequential and parallel algorithms. Sequential algorithms have consecutive steps of instructions that are executed in a chronological order to set up paths in the switching network. On the other hand, parallel algorithms divide the process of setting up routes into smaller sub tasks and these are executed in parallel to obtain individual outputs which are combined to obtain a final desired output. Relative to sequential algorithms, parallel algorithms have a faster processing time. Parallel control algorithm is the key to improving the performance of switching networks. This section focuses on the design of a parallel switch setting algorithms in Benes and Clos networks.

6.1 Benes Network

The BNW has been used in many areas such as interconnections of parallel computers and networks-on-chip (NoC). Several connecting paths exist in switching fabrics. A random connection can be made through any of these arbitrary paths. However, to preserve the properties of the Benes switching fabric such as non-blocking property, a specific path must be used to connect an input to an output. Algorithms to set up such connections are referred to as path searching algorithms. Let's consider the parallel construction of the BNW. Fig.38 shows a primitive model of an $N \times N$ BNW in a three stage structure where a parallel pair of half-sized (i.e. $N/2 \times N/2$) matrices are in between two sets of $N/2$ SEs. Each $N/2 \times N/2$ matrix may be replaced in a three-stage BNW with a pair of $N/4 \times N/4$ matrices. The reduction proceeds until the minimum size matrices, i.e. 2×2 SEs appears in the centre stage. As a result, a complete BNW can be obtained as shown in Fig.39, where each SE has two connection modes, i.e. bar (=) and cross ($\times$), flipped by a SCB. Note that $S_{p,q}$ represents the q-th top SE at the p-th stage from the left, where $0 \leq p \leq 2n-2$ and $0 \leq q \leq N/2-1$. Assume $s_{p,q} \in \{0,1\}$ indicates its status, and $s_{p,q}$ of 0 and 1 makes $S_{p,q}$ bar and cross. As can be seen in figure 39, the BNW is equivalent to a back-to-back concatenation of two baseline networks. The routing algorithm proposed in this study is broken down into two parts similar to the BNW structure. The first part is the division of the permutations recursively into half while satisfying the SDR constraints as shown in figure 39. The division is the same as that of the classical looping algorithm [2],[49].

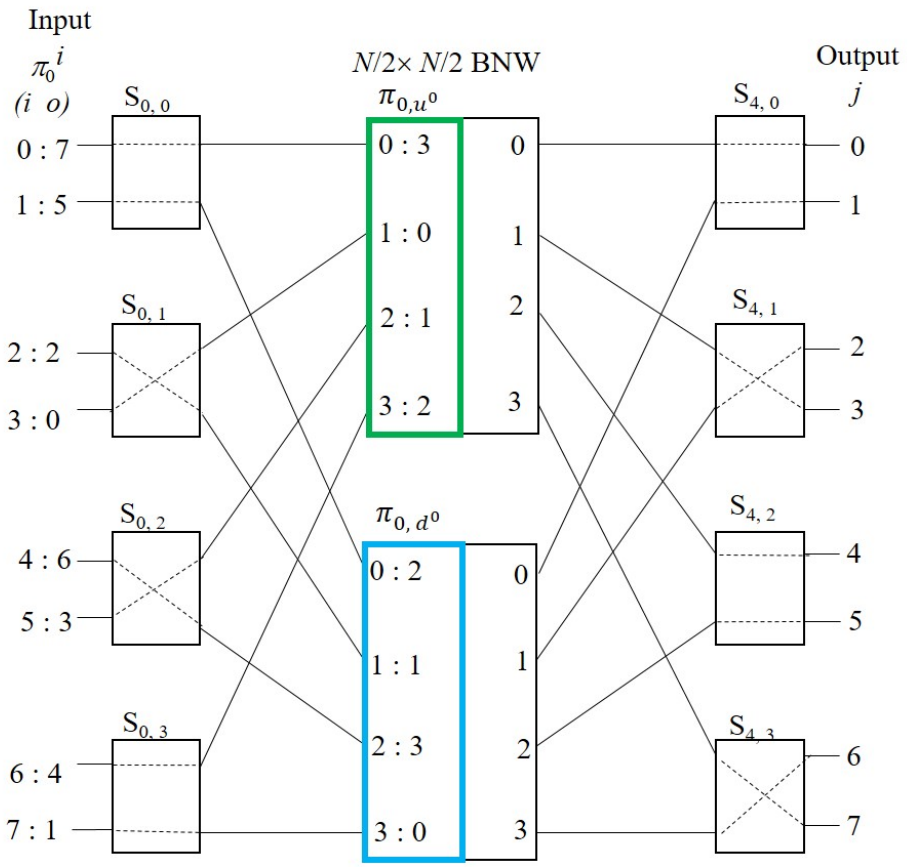


Figure 39: *Primitive three–stage structure of $N \times N$ BNW ($N = 8$)*

Consider a set of permutations (i, o) as shown in figure 38, where i is the input port and o represents the output port to which a specific connection needs to be established. Initially all inputs and outputs are unconnected. The looping algorithm is performed as follows;
1). Select an unconnected set (i, o) and set the connection from input to output switch. If no such input exists, the algorithm ends since all the connections are set.
2). The selected unconnected input i is connected to the unconnected output o through the upper subnetwork of the center stage.
3). Then connect the other corresponding output o at the output stage to a corresponding input in the input stage through the lower subnetwork of the center stage.
4). If the corresponding input is not connected to an output, then repeat the procedure by going back to step 2.
The looping algorithm works by traversing the network from the inputs to the outputs and back until it reaches its starting point thereby forming a loop. This is shown in figure 40 below.

The looping algorithm is a sequential algorithm and therefore it takes $O(N)$ time since N calls must be set up. Let i and j, where $0 \leq i \leq N-1$ and $0 \leq j \leq N-1$, be the input and output port numbers in an $N \times N$ BNW. Firstly, a full permutation π_0 is considered as follows:

$$\pi_0 = \begin{pmatrix} 0, 1, ...i, ...N-1 \\ 0, j_1, ...j_i, ...j_{N-1} \end{pmatrix} \tag{14}$$

where j_i is the designated output at input i.

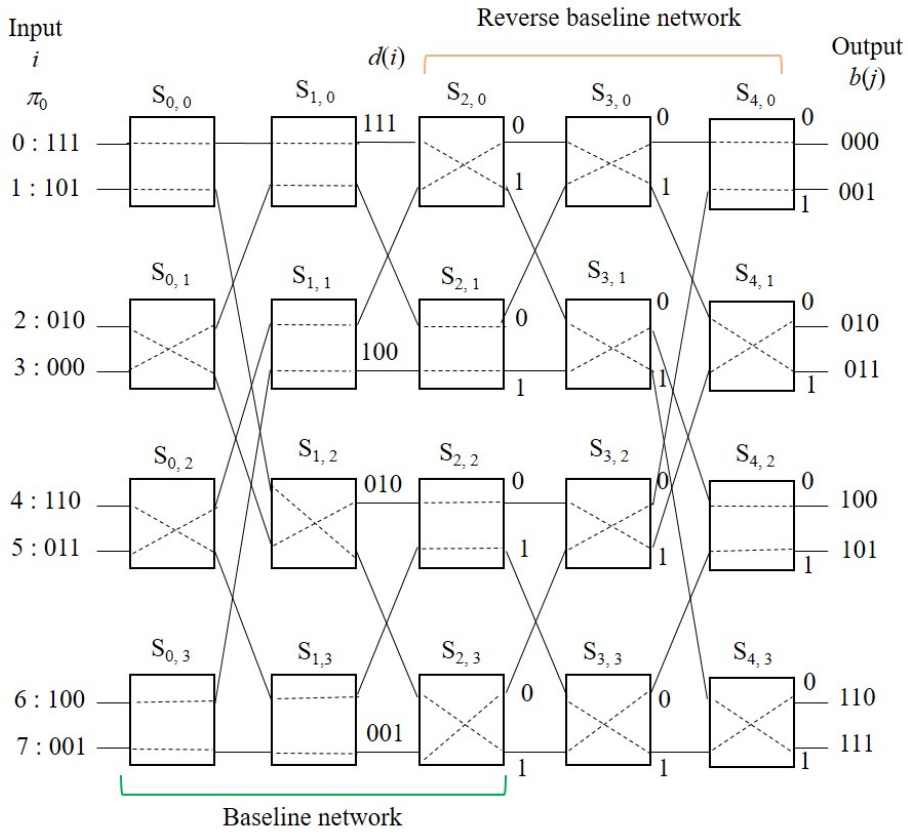


Figure 40: *Full picture of* $N \times N$ *BNW* ($N = 8$)

As an example, the following permutation π_0 is assumed in figure 38 & 39.

$$\pi_0 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 7 & 5 & 2 & 0 & 6 & 3 & 4 & 1 \end{pmatrix} \tag{15}$$

A duplet is expressed as $(i : j)$ corresponding to a pair of input i and output j. Let $b(j)$ be the binary address of output j, which is defined by

$$b(j) = (b_j^{n-1}, b_j^{n-2}, ..., b_j^0) \tag{16}$$

where b_j^t, $0 \leq t \leq n-1$, with

$$\begin{aligned} b(j)_{MSB} &= b_j^{n-1}, \\ b(j)_{LSB} &= b_j^0. \end{aligned} \tag{17}$$

All SEs in the last stage in figure 39 has a pair of outputs j and j_c, of which $b(j)_{LSB}$ and $b(j_c)_{LSB}$ are different and herein referred to as complementary. $m(j)$ is defined as the output SE number with j, where $0 \leq m(j) \leq N/2 - 1$, and

$$m(j) = m(j_c) \tag{18}$$

Let $\underline{b(j)}$ be the truncated binary address of $b(j)$, defined by

$$\underline{b(j)} = (b_j^{n-1}, b_j^{n-2}, ..., b_j^1) \tag{19}$$

The following expression is obtained;

$$\underline{b(j)} = \underline{(bj_c)} \tag{20}$$

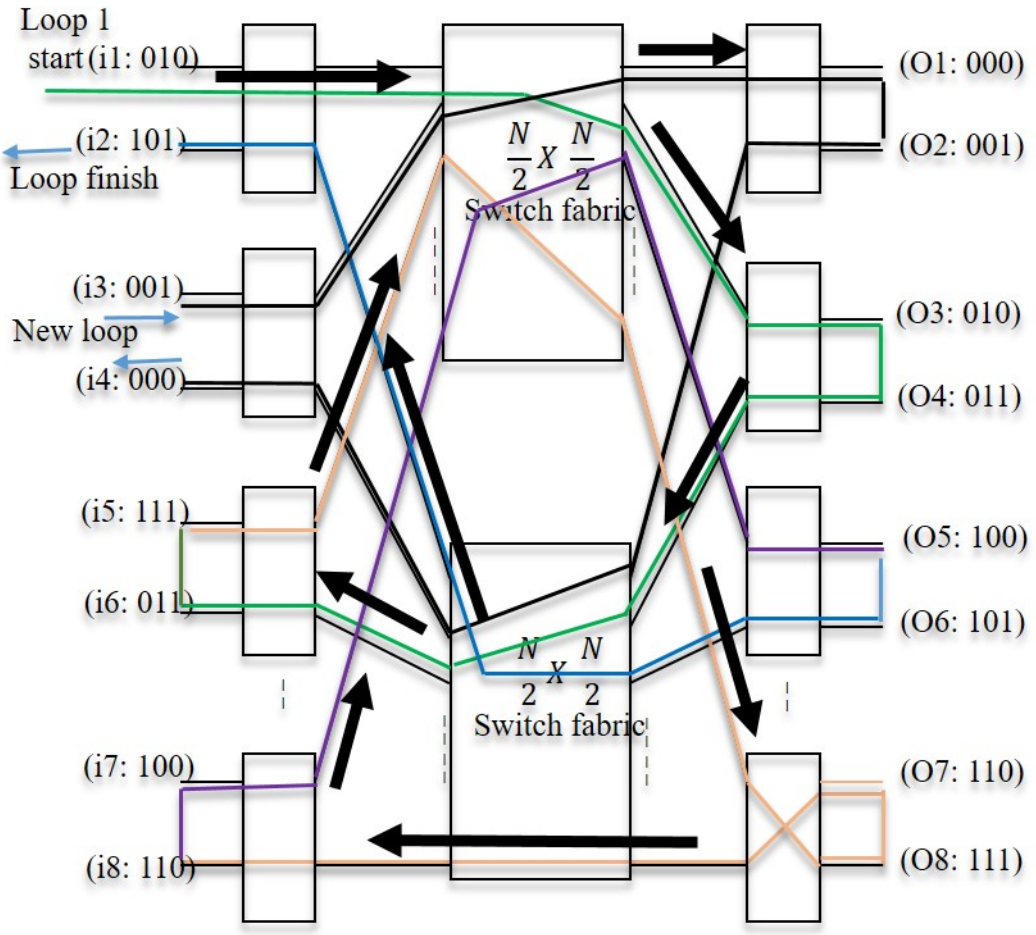


Figure 41: *Conventional looping algorithm in Benes network*

Similar relationships hold between i and i_c in an input SE, which has a pair of duplets $i : j$ and $i_c : j_a$, where j_a is the associated destination output in the input SE. Note that j and j_c together with (i and i_c) are used interchangeably. The original permutation π_0 is divided into a pair of sub-permutations as follows:

$$\begin{aligned} \pi_{0,u^0} &= \{u_0^0, u_1^0, ..., u_{N/2-1}^0\}, \\ \pi_{0,u_d} &= \{d_0^0, d_1^0, ..., d_{N/2-1}^0\}. \end{aligned} \tag{21}$$

where u_l^0 and d_l^0 ($0 \leq u_l^0 \leq N/2-1$ and $0 \leq d_l^0 \leq N/2-1$ for $0 \leq l \leq N/2-1$) are output SE numbers to which a designated output j belongs, as shown in figure 38. Previous research referred to a sub-permutation as a complete residue system [50] or as an equivalent class [31]. In this study, sub-permutations are referred to as SDR because they constitute *systems of distinct representative* (SDR) [51, 52].

Assume that the three-stage structure shown in figure 38 is the 0th reduction. Generally, at the k-th reduction, with $0 \leq k \leq n-2$, there are 2^k sub-permutation pairs $(\pi_{k,u^0}, \pi_{k,d^0}), ..., (\pi_{k,u^{2^k-1}}, \pi_{k,d^{2^k-1}})$, where each sub-permutation has $N/2^{k+1}$ elements. The width of the binary address of these elements is given by $n-k-1$. Note that, an output address of n bits is transferred to the next sub-permutation as it is, while the region of interest in the address at the k-th reduction, denoted by $R(b(j), k)$, is reduced

to $n-k-1$ bits, truncated by one bit at each reduction, and is expressed as follows:

$$R(b(j),k) = (b_j^{n-1}, b_j^{n-2}, ..., b_j^{n-(n-k-1)}) \tag{22}$$

The division process is the same as that of the looping algorithm. It is briefly described here so as to provide an introduction to the parallel algorithm. Because the algorithm is recursive in nature, the primary focus is only on the first stage so as to illustrate how the permutation π_0 is divided into π_{0,u^0} and π_{0,d^0}.
Consider the permutation in equation (15) above.

i. Select an arbitrary SE that is not set yet and set it to bar state by default. In the initial state, $S_{0,0}$ is selected and set as $S_{0,0} = 0$. Let the upper duplet be $i : j$ ($0 : 7$ in figure 38). At this point, $\pi_{0,u^0} = \{3\}$ and $\pi_{0,d^0} = \{2\}$ are obtained.

ii. Search for an SE to which j_c belongs ($j_c = 6$ in figure 38) and find j_c in $S_{0,2}$. Here a pair of outputs $j = 7$ and $j_c = 6$ shares $S_{4,3}$, and therefore must be distributed to different sub-permutations due to the SDR constraints. As $m(j) = 3$ has already been included in the π_{0,u^0} at step i, $m(jc) = 3$ must be in π_{0,d^0}. As a result, $S_{0,2}$ is set equal to 1. At this point, $\pi_{0,u^0} = \{3, , 1\}$ and $\pi_{0,d^0} = \{2, , 3\}$ are obtained.

iii. Next, identify $j_a = 3$ in $S_{0,2}$ as the associate of $j_c = 6$. Because $j_c = 6$ was set in the lower sub-permutation, j_a is taken to the upper permutation. A pair of outputs $j = 3$ and $j_c = 2$ share $S_{4,1}$, these must also be distributed to different sub-permutations. This process repeats until the cycle terminates. The following permutations are obtained; $\pi_{0,u^0} = \{3, 0, 1, 2\}$ and $\pi_{0,d^0} = \{2, 1, 3, 0\}$. It is worth noting that the status of some SEs like $s_{0,0}$ and $s_{0,3}$ are equal because the relative input positions of the pair of duplets including $j = 5$ and $j_c = 4$ differ [31], i.e., one sits at the upper inlet in one SE and the other on the lower in the other SE.

The second part of the proposed routing algorithm is the destination-tag routing (DTR). This correlates to a superposed binary tree structure of the reverse baseline network (RBN) [53] shown in figure 39. Here, assume $d(i)$ to be a binary destination output imposed at the i-th input of the RBN as follows.

$$d(i) = (d_i^{n-1}, d_i^{n-2}, ..., d_i^0) \tag{23}$$

where $d_i^t \in \{0, 1\}$, $0 \leq i \leq N-1$ and $0 \leq t \leq n-1$. Each pair of outlets of SEs in the RBN is labelled with 0 and 1, as can be seen in figure 39. A route can be traced from i to $d(i)$ as follows; When $d(i)$ appears at an input of $S_{p,q}$, it is forwarded to either outlet 0 or 1 of the $S_{p,q}$ depending on whether d_i^{2n-2-p}= 0 or 1. This operation is realized in time of order $O(1)$. The detailed description of the DTR processing and its hardware implementation have not been included in this thesis.

6.2 Design of parallel processing elements

6.2.1 Design overview

In this study, a design of parallel distributed architecture of processing elements (PEs) is proposed. As can be seen in figure 41, the upper half is a switch body of size $N \times N$ BNW and the lower half is the proposed Parallel control Unit (PCU). The PCU accepts address information represented by A_{in} slot by slot, then it computes routes based on these addresses, then it generates switch control bits (SCBs) stage by stage in parallel. The design of the PEs is such that a PE at a stage is connected to a pair of other PEs in the next stage in the same manner as the counterpart SEs in the upper BNW switch body. A PE receives a pair of addresses and transfers them to the next PEs in the same mode (i.e. bar or cross) as the SE. The PCU has upto $O(N \log_2 N)$ PEs and is similar to a massively paralleled system. The PEs can comprise several kinds of primitive logic circuits and memories such as comparators, multiplexers and registers unlike [35] and [39] in which the shared memories and arithmetic units were required.

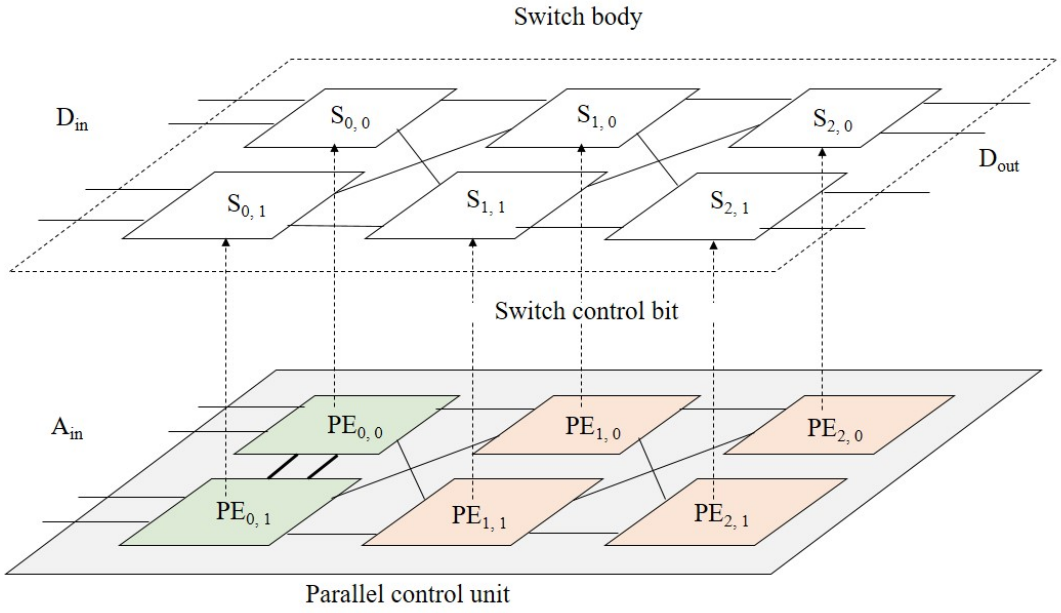


Figure 42: *Proposed Design overview of the parallel control unit with the switch body*

As can be seen in figure 41, there are two types of PEs, i.e. $PE_{k,h}$ in the first part, with $(0 \leq k \leq n-2)$ and $0 \leq h \leq N/2-1$ and $PE_{k',h}$ in the second part, where $(n-1 \leq k' \leq 2n-2)$ and $(0 \leq h \leq N/2-1)$, according to the earlier mentioned two parts of routing algorithm. The first stage of the PCU corresponds to the index of $k = 0$. Note that PEs in the first k-th $(0 \leq k \leq n-2)$ stage are divided into 2^k independent groups. Therefore, a PE group in the k-th stage has a total of $N/2^{k+1}$ PEs and accepts a total of $N/2^k$ addresses, of which the bit width of the region of interest becomes $n-k-1$. All the PEs within a group are interconnected with $N/2^{k+1}$ buses to communicate within the group. These are shown as bold lines in the first stage of the PCU in figure 41. In the first part of the proposed routing algorithm, the division process of sub-permutations is common to all the groups, although the number of PEs in

a group (i.e. size of sub-permutations) decreases with stage. In the second part, all the PEs have uniform, low complexity. Because of this, the PEs in the first stage of the PCU have the most complex hardware. It takes $O(\log_2(N/2^k))$ processing time for a PE group in the k-th reduction, where $(0 \leq k \leq n-2)$. All the PEs in the second part require small constant time of $O(1)$. As a result, the processing time and hardware complexity are most critical in the first stage of the PCU. In the next section, the proposed parallel algorithm is described in detail. This study focuses on the PEs in the first stage as in Ref. [37].

6.2.2 Design of PEs for Full permutation

The first stage of the PCU has a single group of $N/2$ PEs interconnected to each other via a number of buses $B_{b(r)}$ (e.g. B_{00}), where $(0 \leq r \leq N/2-1)$ as shown in figure 42 (a). Binary suffixes are assigned to the $N/2$ PEs as $PE_{b(p)}$ (e.g PE_{00}), where $(0 \leq p \leq N/2-1)$. Note that the status of $N/2$ SEs are also re-assigned with binary suffixes (e.g. S_{00}).

We assume a full permutation in figure 42 (a) as follows:

$$\pi_0 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 4 & 5 & 0 & 2 & 6 & 7 & 1 \end{pmatrix} \tag{24}$$

Figure 43 shows a simplified functional block diagram of the PEs. The function of each block is described as follows. During the division process,each $PE_{b(p)}$ sends out its own data, e.g. $\underline{b(j)}$, $\underline{b(j_c)}$ and $\overline{b(p)}$, to the bus $B_{b(p)}$. Here, $\overline{b(p)}$ is a PE suffix that is identical to $b(p)$ for full permutations and will later be redefined in the partial permutations section. Each bus comprises the data information of $n-1$ bits and a data valid flag (f_v) of 1 bit. At most one pair of eligible data is accepted by each PE through multiplexers in the bus interface (BIF) of figure 43 even though each PE is capable of receiving multiple data simultaneously. Thus, the internal processing speed of the PEs remains constant, independent of the PE group size. In the full permutation, the division process has three phases, which are described in the following section.

i). First phase

The first phase of the process is the parallel process for neighbour search. The $PE_{b(p)}$ is assumed to have two destination addresses $b(j)$ and $b(j_a)$, as shown in Fig. 43. The neighbour search is performed to find PEs with truncated addresses $\underline{b(j)}$ and $\underline{b(j_a)}$. If $\underline{b(j)} = \underline{b(j_a)}$ holds in a PE, the neighbour PE is identical to the original PE, and no addresses are launched onto the bus. Otherwise, $\underline{b(j)}$ and $\underline{b(j_a)}$ are broadcast in series onto $B_{b(p)}$. The other PEs compare their own truncated addresses with the incoming $\underline{b(j)}$ and $\underline{b(j_a)}$. This is done by using the comparators in the BIF. As shown in fig.42(b), $\overline{PE_{00}}$ first broadcasts $\underline{01}$ from its upper inlet to bus bar B_{00}, the other PEs compare their truncated addresses to $\underline{01}$. PE_{10} realizes PE_{00} as a neighbour because it has $\underline{01}0$ at its upper inlet. Here, both $\underline{01}1$ and $\underline{01}0$ are at upper inlets of the two neighbour PEs. This means that they must be in different sub-permutations; and therefore, the status of PE_{10} and corresponding S_{10} is not equal to PE_{00} which also corresponds to s_{00} (Fig. 42 (c)). Similarly, when PE_{00} launches $\underline{10}$ from its lower inlet, $PE01$ discovers PE_{00} as a neighbour, the status of PE_{01} (S_{11}) is set equal to PE_{00} (s_{00}) because PE_{01} has $\underline{10}1$ at its upper inlet.

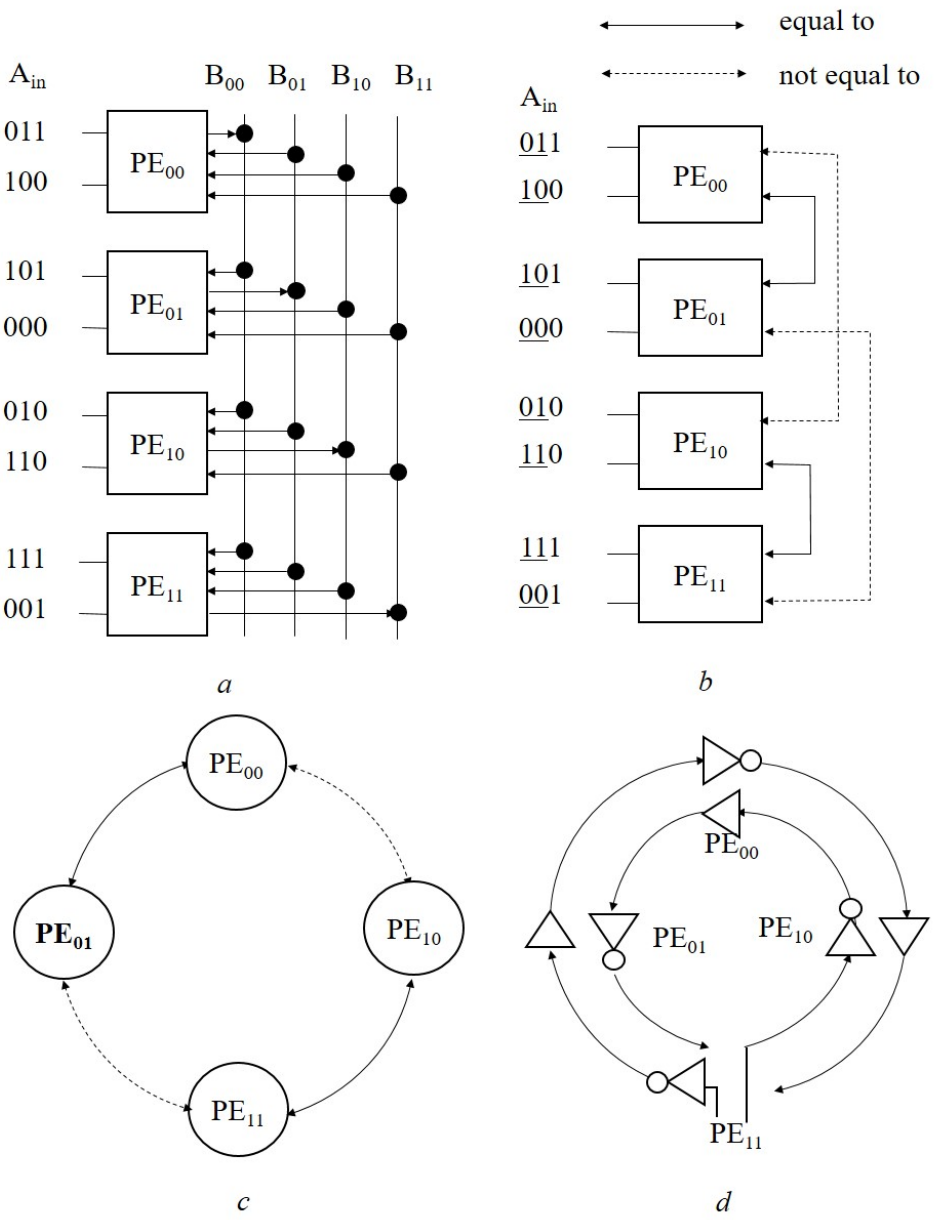


Figure 43: *Bus-connected PE array and its operation: (a) each PE accepts address information and interchanges switch control data via multiple buses in parallel; (b) link status after first phase; (c) link diagram in a cycle; (d) implementing link diagram in FPGA using inverting and non-inverting gates*

The *'equal to'* and *'not equal to'* relations are shown in figs. 42 (b) and 42 (c) by solid and dashed links, respectively. Here, we refer to both relations as link status collectively. The status of each PE corresponds to the SE status. Link status flag (f_s) and neighbour PE suffixes are saved in registers, shown in fig. 43. In practice, a pair of counter-rotating links is implemented with a portion of each bus in the third phase. Inverting and noninverting gates shown in fig. 42 (d) are used to implement the link status relationship between two adjacent PEs in FPGA, where the link originates and terminates at PE_{11}, as the result of the following second phase. The neighbor search process is realized in parallel, and the time complexity is $O(1)$.

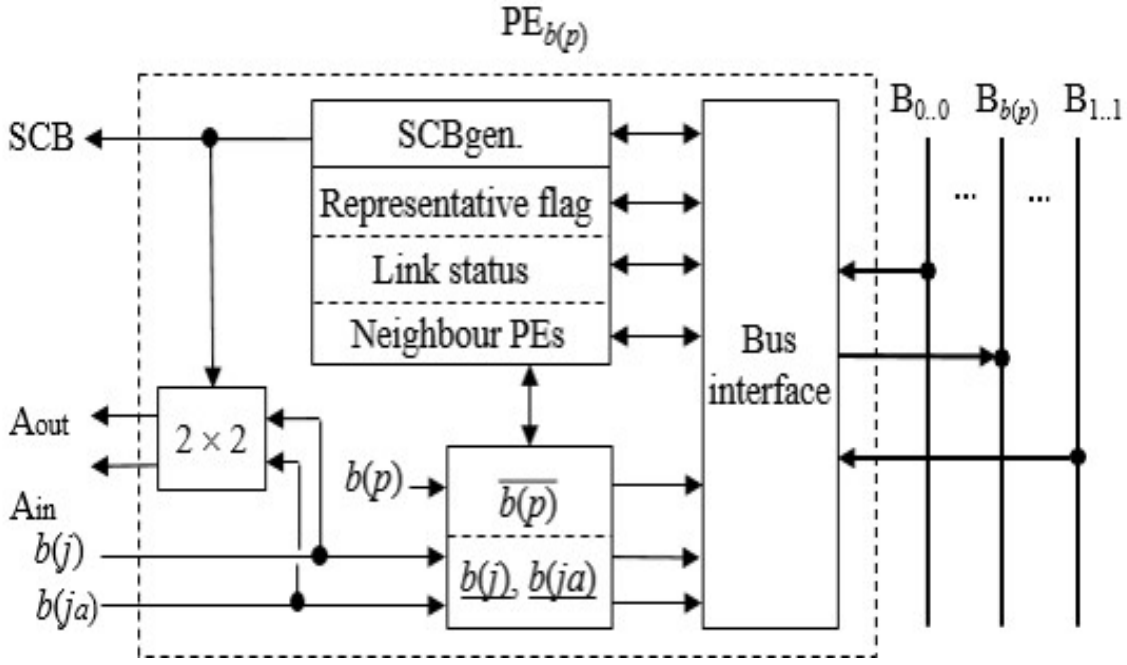


Figure 44: *Simplified functional diagram of PE with binary suffix* $b(p)$

ii). Second Phase
The second phase involves an iterative process to select a representative element PE in each cycle. In a cycle (fig. 42(c)) each PE is correlated with each other and a system of relations is referred to as the initialization equation. The proposed solution for the initialization equation is based on the fact that if we appoint a PE (E.g. PE_{11} in Fig.42 (c) as a representative element in the link diagram and set SE to the default state, the status of the other SEs is generated automatically in the link diagram which cannot be easily implemented with only software. Therefore, it is necessary to have a representative element for determining the status of other SE based on this representative element. The proposed design assumes the representative PE to be the one with the maximum suffix in a cycle (e.g. PE_{11} in the fig. 42(c)). The iterative procedure is as follows; in the initial state, each PE in a cycle is a candidate for a final representative, whose status is indicated as a double circle in Fig. 44(a), where only binary suffixes of PEs, i.e., $b(p)$, are shown for simplicity. In the first step, each PE sends its suffix to two neighbouring PEs, as shown in Fig. 44(a), and each PE compares its own suffix y_0 to the incoming suffixes y_1 and y_2. Note that a similar situation occurs in the subsequent steps, and there are three possible cases.
(i). If $(y_0 < y_1) \cup (y_0 < y_2)$, the PE loses the competition, y_1 and y_2 are transferred to the next neighbour PEs, e.g. PE_{00} and PE_{01} in Fig. 44(b), and PE_{10} in Fig. 44(c), lost competition and are shown by dashed circles. Note that the PEs that loses become transparent in the subsequent steps.
(ii). If $(y_0 > y_1) \cap (y_0 > y_2)$, the PE remains as a potential representative and discards both y_1 and y_2, e.g. PE_{11} and PE_{10} in Fig. 44(b).
(iii). If $y_0 = y_1 = y_2$, the PE becomes a final representative and goes to the third phase, e.g. PE_{11} in Fig. 44(d).

PEs that lost the competition bypass incoming data through simple transfer switches embedded in BIF; thus, the bypass delay is negligible. The comparison result at each

PE is updated in a representative flag (f_r), as shown in fig. 43. Neighbour PE suffixes are referred to by the BIF to specify a bus to receive the suffix data from. Note that each individual cycle has a maximum suffix, and there can be several representatives in a stage, e.g. $S_{0,0}$ and $S_{0,3}$ (each corresponding to PE_{00} and PE_{11} respectively) as shown in Fig. 38. As a result, the number of candidates for a representative is reduced to one-half after each iteration; thus, the second phase for the first stage completes in at most $O(\log_2 N)$ time.

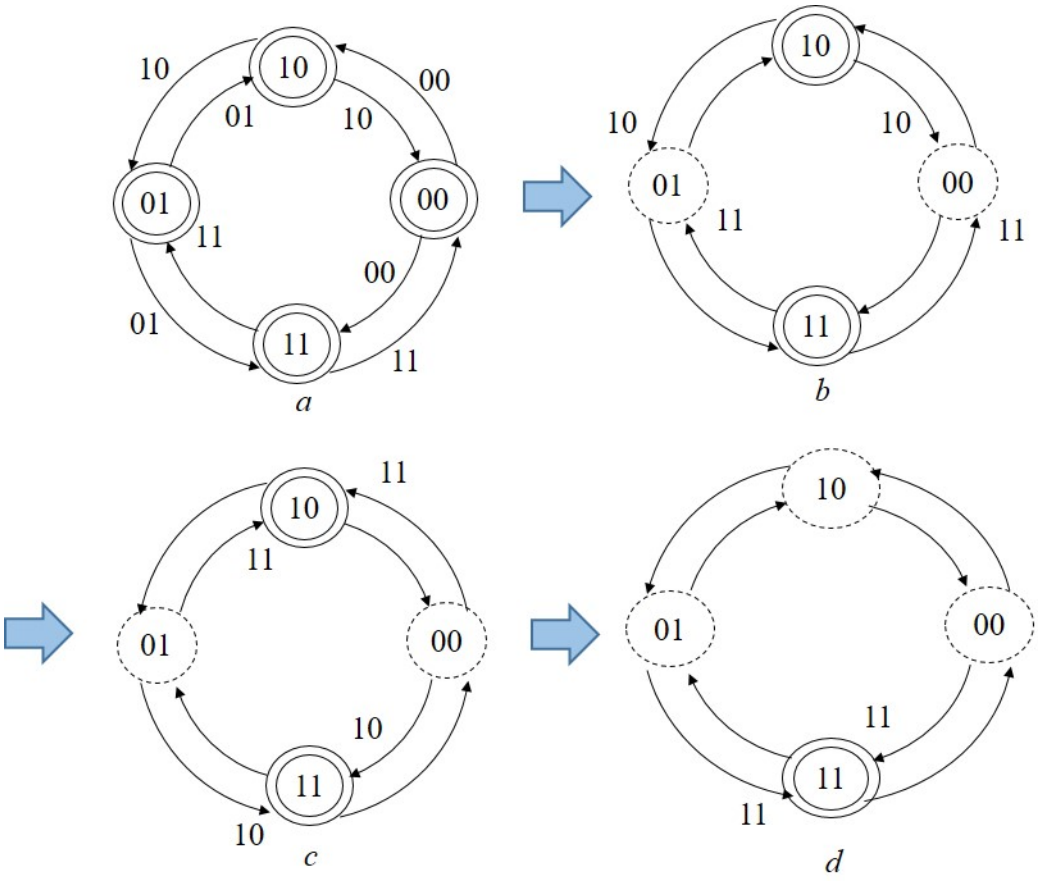


Figure 45: *Iterative steps to determine representative PE in the second phase: (a) each PE is eligible for representative in the initial state; (b) the PE receives two suffixes and compares its suffix to them and surviving PEs are reduced to one-half and lost PEs become transparent and bypass incoming data; (c) the competition process is repeated in subsequent iterations; (d) maximum suffixes return to the originating PE in the final iteration*

(iii). Third Phase

The third phase is performed to determine the status of each SE in each cycle according to the direction of a representative and pass addresses on to the next stage. For example, PE_{11} in Fig. 42(d) sets an initial value of $s_{11} = 0$, which also serves as a trigger to the link diagram. Note that each link between neighbour PEs has already been established over buses in the beginning of the third phase. In addition, inverting and non-inverting gates, which reside in an SCB generator in Fig. 43, have been configured according to f_s. Here, a pair of input addresses $b(j)$ and $b(j_a)$ is transferred to PEs in the next stage through a 2×2 switch (Fig.43), which is set to bar or cross by an SCB. The propagation delay in the link diagram is very small compared to a clock period; therefore, the time complexity

of the third phase is estimated as $O(1)$. To summarize, the total processing time in the first stage is $O(\log_2 N)$ Generally, the total processing time in the k-th reduction, where $0 \leq k \leq n-2$, is $O(\log_2 N/2^k)$, because the size of permutation in the k-th reduction is given by $O(\log_2 N/2^k)$. It is clear that the first stage has a maximum processing time, and it decides the maximum throughput of the pipelined PCU.

6.2.3 Design for partial permutations Unified Parallel algorithm for full and partial permutations

At some instances during the operation of the switch, some inputs may not have a connection request. These inputs without a connection request are referred to as idle connections. In this section, idle connections and their effect on the proposed algorithm are investigated. A partial permutation as follows is assumed;

$$\pi_0 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 4 & 5 & 0 & x & 6 & 7 & 1 \end{pmatrix} \tag{25}$$

where x indicates an idle address. Similar to the full partial permutation in the previous section, the fifth address element has been changed to x.

As can be seen in fig. 45(a) and (b), no links exist between PE_{00} and PE_{10} due the idle connection in PE_{10}. A representative element cannot be determined this way because neighbour prefix information y_1 and y_2 cannot be shared due to missing links. There are two design options to tackle this problem. The first one is a structural adaptation to loop back the disconnected links at the two end PEs so that they can each have a pair of neighbours like in a full permutation. This loopback does not change the comparison operation for the selection of a representative element, however, it almost doubles the length of a cycle thereby increasing the processing time. The other option is to mask the disconnection by blocking the received suffix data at the two end PEs. To do this, $\overline{b(p)}$ is first redefined as follows:
If a PE has a single neighbour for example PE_{00} and PE_{10}, which sits at both ends as shown in fig.45 (b) then the suffix $\overline{b(p)}$ is obtained by modifying it as:

$$\overline{b(p)} = (1, b_p^{n-1}, b_p^{n-2}, ..., b_p^1) \tag{26}$$

If a PE has two neighbours like PE_{01} and PE_{11} which sits in between two end nodes, then $\overline{b(p)}$ is modified as follows:

$$\overline{b(p)} = (0, b_p^{n-1}, b_p^{n-2}, ..., b_p^1) \tag{27}$$

From the two equations above, it is very clear that the suffixes of the two end nodes are greater than those of the intermediate PEs. This immediately makes the two PEs to survive the competition to the final iteration. The two nodes eventually exchange suffix information over the available link and either becomes the final representative. A pair of terminating conditions must be added to step iii of section 6.1 as follows. Note that if $\overline{b(y)}_{MSB} = 1$ it means that $PE_{b(y)}$ is an end node.
iv. If $\overline{b(y)}_{MSB} = 1 \cap (y_0 < y_1)$, one end PE losses the competition and the procedure

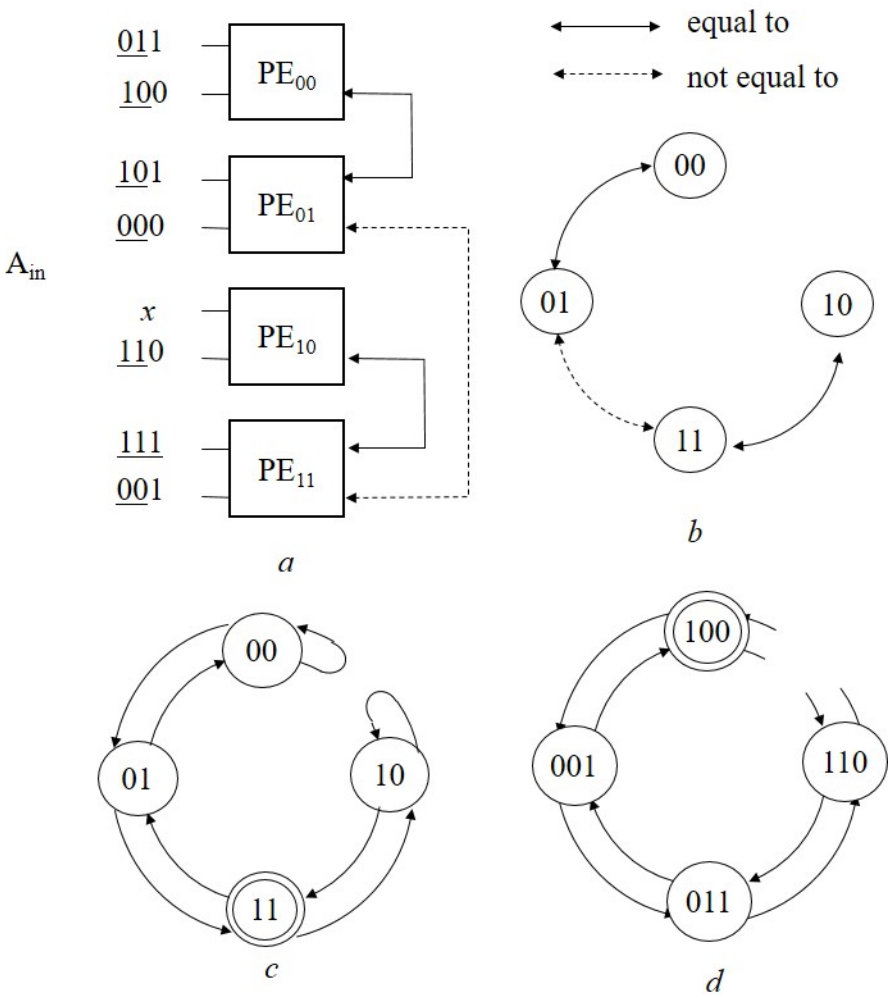


Figure 46: *Example of partial permutation and modified link diagrams against disconnection: (a) input address 010 in Fig. 42 a becomes idle and is represented as x; (b) the link diagram is disconnected between PE_{00} and PE_{10}; (c) the disconnection is protected by the loop-back mechanism (similar to self-healing ring networks); (d) alternatively, each PE is provided with an extended suffix to mask disconnection*

halts, e.g. PE_{100} in Fig. 45(d).
v. If $\overline{b(y_1)}_{MSB} = 1) \cap (y_0 > y_1)$, the other end PE becomes a final representative and goes to the third phase, e.g. PE_{110} in Fig. 45(d). It is worth noting that the second design option incurs very little additional processing time and hardware while maintaining the comparison operation for full permutation. The PCU developed in this thesis, uses the second design option to realise high speed and low hardware complexity.

6.3 FPGA Design and Experimental Results for Benes network

6.3.1 Design environment

FPGAs are an efficient hardware design target for rapid prototyping [54]. In this study, a Xilinx midrange FPGA (XC6SLX45), which has 6,822 configurable logic blocks (CLBs) operating at 100 MHz was used. The ISE Design Suite 14.7 development tool for Windows 10 was used for writing the code for PEs in VHDL with the IEEE Std_Logic_1164ALL package. Synthesis options were set to defaults, i.e. the speed priority mode, normal optimisation effort, etc. A constraint file was used to allow internal signals, e.g., counter outputs, f_r and SCB, to be output to I/O pins to monitor their logic status and measure delay time. In addition, the input addresses were generated within the FPGA. Note that each time the permutations pattern changed, the FPGA needed to be redesigned. For this however, external digital pattern generators were not required. The total hardware amount was estimated according to the number of occupied slices in the summary report in order to analyse device utilisation. Note that each PE in a given stage operates synchronously to a common clock. In the proposed algorithm, three processing steps operate synchronously. The first step is an initializing process, where destination addresses are imported and flags and registers are reset. The second step is composed of the three phases described in Section 6.2.2. The third step is a terminating process, where SCBs and addresses are exported. In order to increase the processing speed, asynchronous hardware operation was introduced in part as follows, without making any change to the parallel algorithm described in Section 6.2.2 and shown in Fig. 46 below. In the first part of the PCU, the iterative procedure to find a representative (Section 6.2.2), can operate asynchronously by referring to the f_v flags and comparison results. Here, an additional set of buses as shown in Fig. 42(a) were added for this purpose, and each pair of input addresses at PEs was processed simultaneously. DTR for the second part was implemented asynchronously by referring to a specified routing bit given by Eq. (17). All these asynchronous operations were described in VHDL code using if-statements, and the synchronous operations were realized using when-statements. The combined synchronous operation is as shown in Fig. 47.

6.3.2 Experimental results and discussion

Figure 48 shows the number of occupied slices in the first and second parts of the PCU for switch size $N = 4$ to $N = 32$ denoted by HW1 and HW2, respectively. As explained in section 6.2.1, the first part accounts for a larger portion of the PCU hardware, whereas the second part requires a significantly less amount of hardware than the first part. The hardware amount of a single PE was estimated in the first part (Fig.41) as $O(\log_2 N)$ because most parts of a PE have a dimension of $\log_2 N$. Remember that a single stage comprises $N/2$ PEs; therefore the hardware complexity per stage is $O(N(\log_2 N))$. This results in the total hardware complexity denoted by HW1, of $O(N(\log_2 N)^2)$ over $\log_2 N - 1$ stage.Similarly, the total hardware complexity of the second part, denoted by HW2, is also given by $O(N(\log_2 N)^2)$. As can be seen from Fig.47, the experiment results agree with the theoretical estimation, i.e., $O(N(\log_2 N)^2)$.

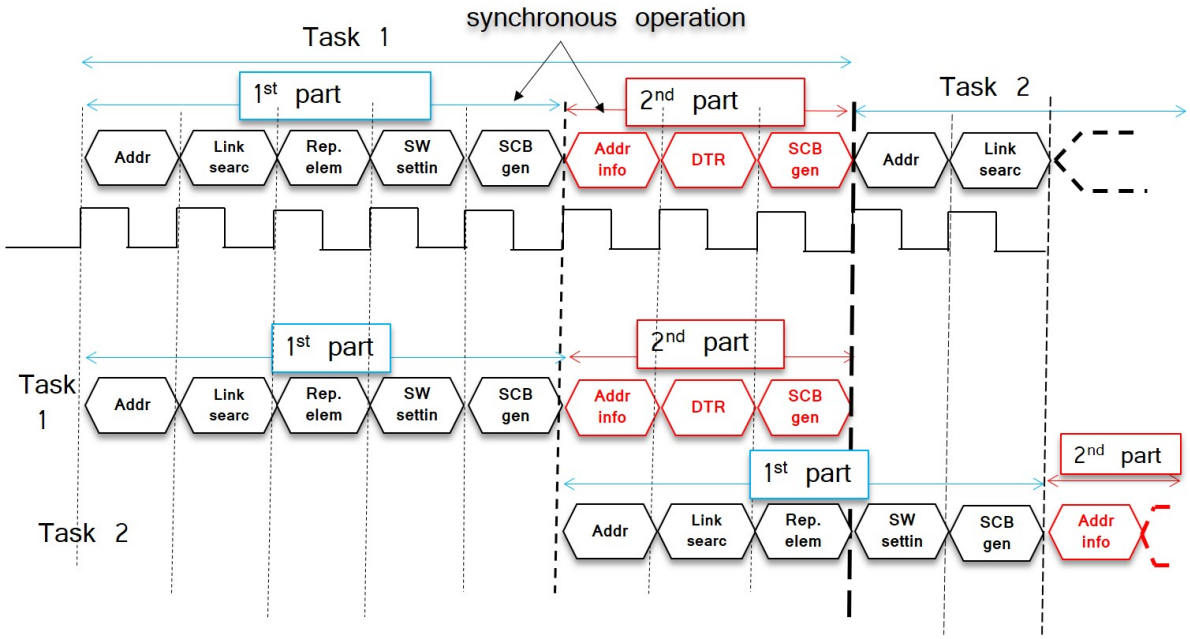


Figure 47: *Asynchronous operation of the proposed structure*

From the experimental results, it has been demonstrated that the occupied CLBs were

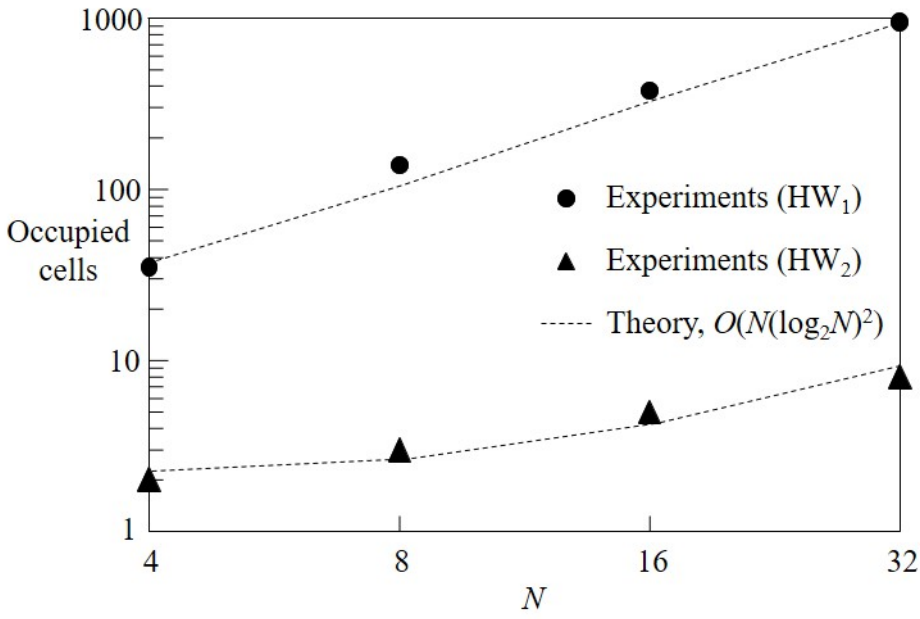


Figure 48: *Number of occupied slices in FPGA vs. switch size*

less than 1,000 when $N = 32$.This is approximately 15% utilisation of CLB which was estimated to be approximately 6,400 cells [55]. Note that the conventional hardware complexity increases as $O(N^2])$ in [35].The number of cells for $N = 32$ is up to 36,200 cells (approximately six times greater than this proposed design). These results indicate that the proposed design is significantly more efficient relative to hardware costs compared to existing methods. As for time complexity, we focus on the processing time in the first stage because it has a largest portion and correlates with the total processing

time as described towards the end of Section 6.2. Specifically, focus was given to the processing time in the second phase (Section 6.2.2), denoted by t_f, which shares most of the processing time in the first stage. Fig.48 shows the processing time t_f vs. switch size N for the first part of the proposed design. From the Fig. 48 t_f remains less than a clock

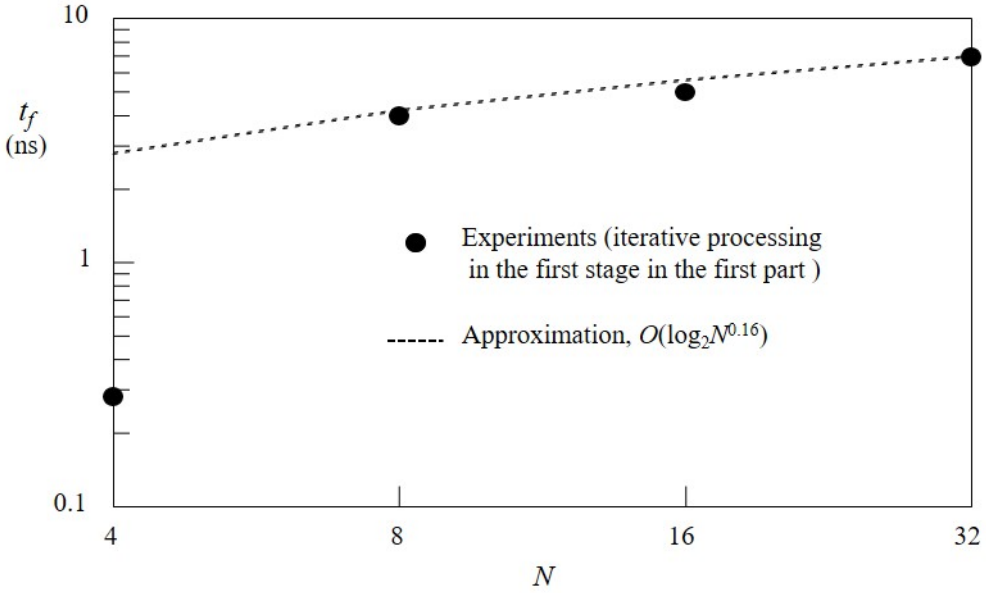


Figure 49: *Processing time t_f vs. switch size N for the first part in the proposed design*

period (10 ns) up to a certain switch size due to the asynchronous operation. It increases in approximately $O(\log_2 N^{0.16})$. It was also found that the asynchronous operation causes a dramatic decrease in processing time. It is worth noting that the attenuating effect given by α depends on FPGA devices and t_f can be generalized as $(\log_2 N^{\alpha})$, $\alpha < 1$. In Fig. 48, it can be observed that t_f for the special case of $N = 4$ indicates a sharp drop. This is because the 4×4 BNW only has two SEs in the first stage, and therefore it is very easy to fix a representative quickly without much time. This tendency to have a sharp drop in the processing time at $N = 4$ was also observed in a previous study [35]. Fig. 49 also shows the processing time for the DTR in the second part of the proposed design, denoted by t_s. As can be seen, t_s is also less than a clock period due to the asynchronous DTR operation. It is significantly less than t_f due to the simplicity of the DTR and is negligible compared with t_f as suggested at the end of Section 6.2. It is worth observing that t_f and t_s have the same approximation curve as $O(\log_2 N^{0.16})$. Recall the time complexity in the first stage is given by $O(\log_2 N)$ as described at the end of Section 6.2.3. The processing for DTR in the second part takes $O(1)$ time per stage as described at the end of Section 6.2. Since the second part has $\log_2 N$ stages, the total processing time complexity is also given by $\log_2 N$. However, both processes of complexity were reduced to $O(\log_2 N^{0.16})$ due to the asynchronous operation, as predicted in Section 6.3.1 and shown in fig. 49. The conventional algorithm [36] for the first stage requires $O(\log_2 N)$ clock times, while the proposed design demonstrated $O(1)$ clock time until $N = 128$, because the most time-consuming process for the proposed algorithm completes within a single clock cycle as shown in Fig. 50. In fact, the proposed algorithm requires 5 clocks, whose breakdown is as follows: one clock for the initializing step, three clocks for the first to third phases, and the last clock for the terminating step described in Section 6.3.1. For example, in the $N = 16$ case, the proposed design is at least three times faster than

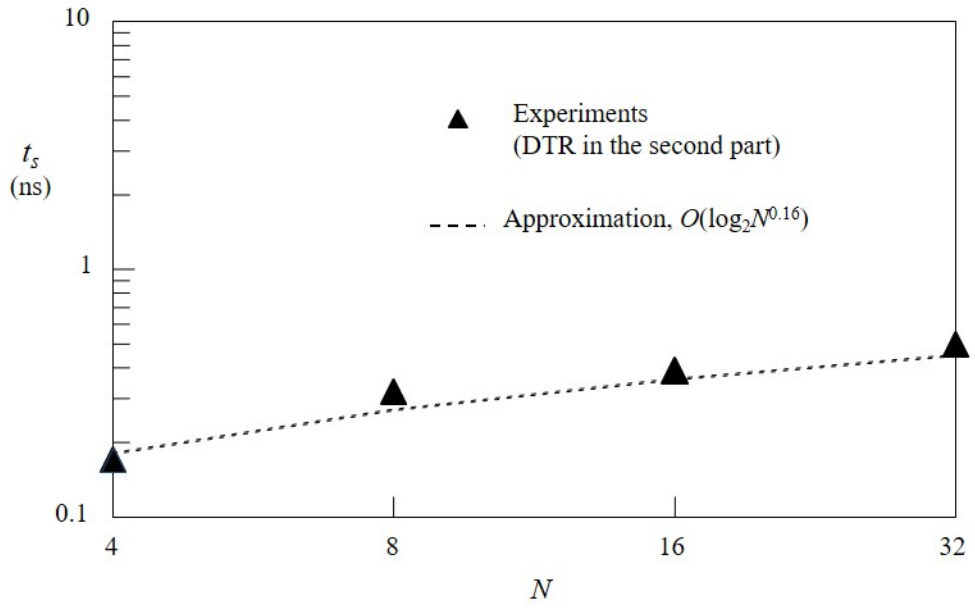


Figure 50: *Processing time (t_s) vs.switch size N for the Second part (DTR) in the proposed design*

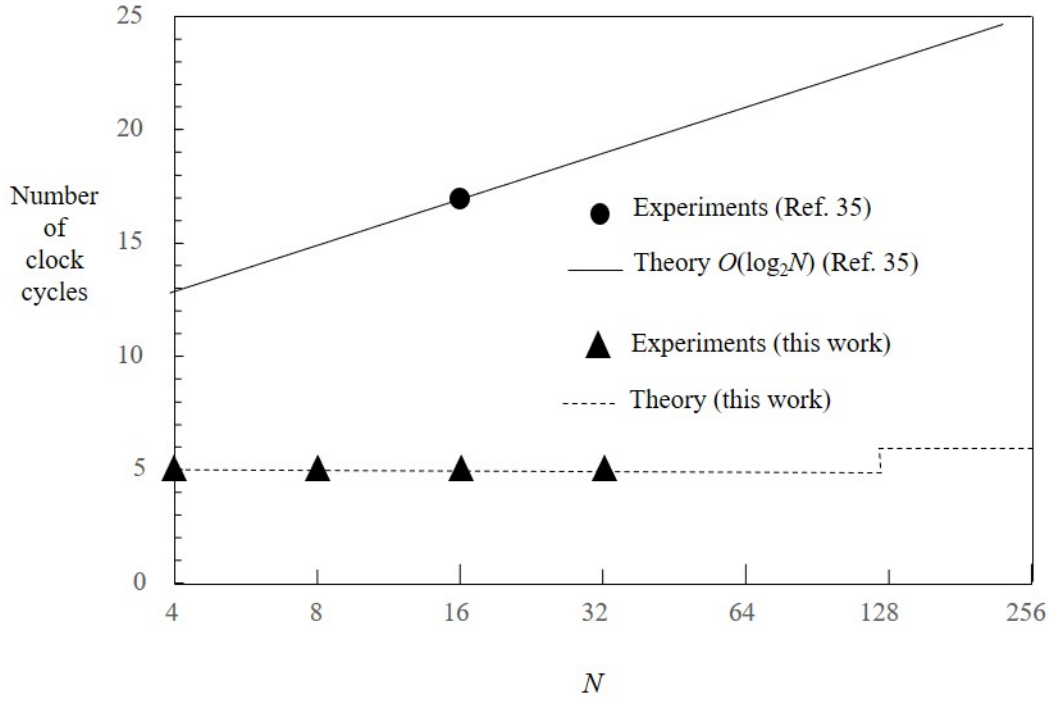


Figure 51: *Total clock cycles vs. switch size for the first stage in the first part*

the previous method, and the performance difference increases with increasing N. It is worth noting that the number of clock cycles for $N > 128$ is estimated to increase by only one clock cycle because of the asynchronous operation introduced. As a result, the proposed design has clock cycles that are several times faster than the previous method [36].

6.3.3 Summary

Table 3 shows a summary of the technique proposed, its implementation, the hardware complexity and the time complexity of the algorithm

Table 3: Summary of the performance of the algorithm

Technique proposed:	Parallel distributed Pipelined architecture of PEs
Implementation:	Synchronous and Partially asynchronous operation of several steps introduced
Hardware complexity	$O(\log_2 N^2)$
Time complexity	1st part requires only one clock per stage as a result total complexity becomes $O(\log_2 N)$

6.4 Clos Networks

6.4.1 Outline of the proposed routing algorithm in TSCN

A three stage Clos network (TSCN) is a fundamental multistage switching network and has been used widely in many communication and interconnection networks. Time-division multiplexed TSCNs used in data centers require high-speed switching capabilities i.e. several nanoseconds to route or set up calls within the switch. Conventional sequential algorithms have longer processing time and therefore a need to implement parallel processing algorithms to speed up routing. There have been no reports of hardware implementation of parallel algorithms developed earlier for TSCN in which takes advantage of the switch size of a power of two. However, parallel algorithms for Benes networks whose switch size is inherently given $N = 2^k$ where k is an integer have already been developed. Here a new parallel algorithm to set up a TSCN using a routing algorithm for Benes network is proposed and implemented in hardware using FPGA. Consider a Clos switch of size $N \times N$ represented by $C(n, n, r)$ shown in fig. 51 below, in which the first and third stage consists of r $n \times n$ switches while the third stage consists of n $r \times r$ switches.

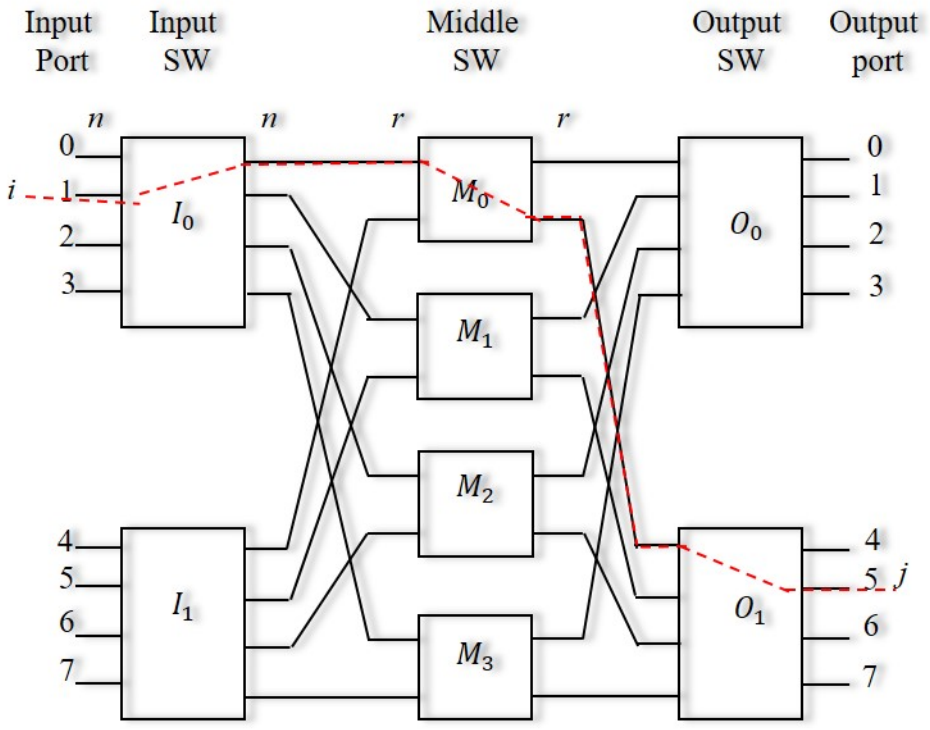


Figure 52: *Three stage Clos network* $C(4, 4, 2)$

Each of the switch size is restricted to be that of a power of two i.e. $n = 2^{k1}$ and $r = 2^{k2}$ with both $k1$ and $k2$ being positive integers. As can be seen in the figure 51, an input i has a total n possible routes to an output j, where $0 \leq i \leq N-1$ and $0 \leq j \leq N-1$ and each is uniquely designated by assigning the route with a middle stage switch as a transit point. The details of the operations are shown in figure 52. The figure shows the first cycle of the proposed algorithm for a clos network of switch size $c(4, 4, 2)$. The output addresses at each of the input ports denote the binary expression of the destination

addresses to which the signal wishes to travel (e.g., $100, 011, 010, 000, 101, 110, 111, 001$). The proposed algorithm for TSCN is as follows: firstly, input switches are virtually divided into $N/2$ 2×2 to obtain a structure similar to Benes network as shown in dotted red line in fig.52. Then a pair of adjacent output addresses are obtained at each virtually divided 2×2 switch. Next, the output addresses are then divided into two sets using the looping algorithm for the Benes network. Note that a pair of addresses with least significant bits (LSB) of the addresses are complimentary while the other bits are identical, e.g.,100 at the first input and 101 at the fifth input are divided into different sets. It is worth noting that every LSB is eliminated after the division. Two sets of truncated addresses $(10, 00, 01, 11)$ and $(01, 10, 00, 11)$ are obtained as shown in figure 53. Each set of these addresses goes through the upper or lower set of two middle stage switches. The second part of the algorithm is that of Applying Destination tag routing (DTR). And in this part, each set of the truncated addresses is processed independently in the same way as the first cycle, however, the number of address elements reduces to one-half and their width reduces by a single bit. It can be seen that the algorithm can be completed

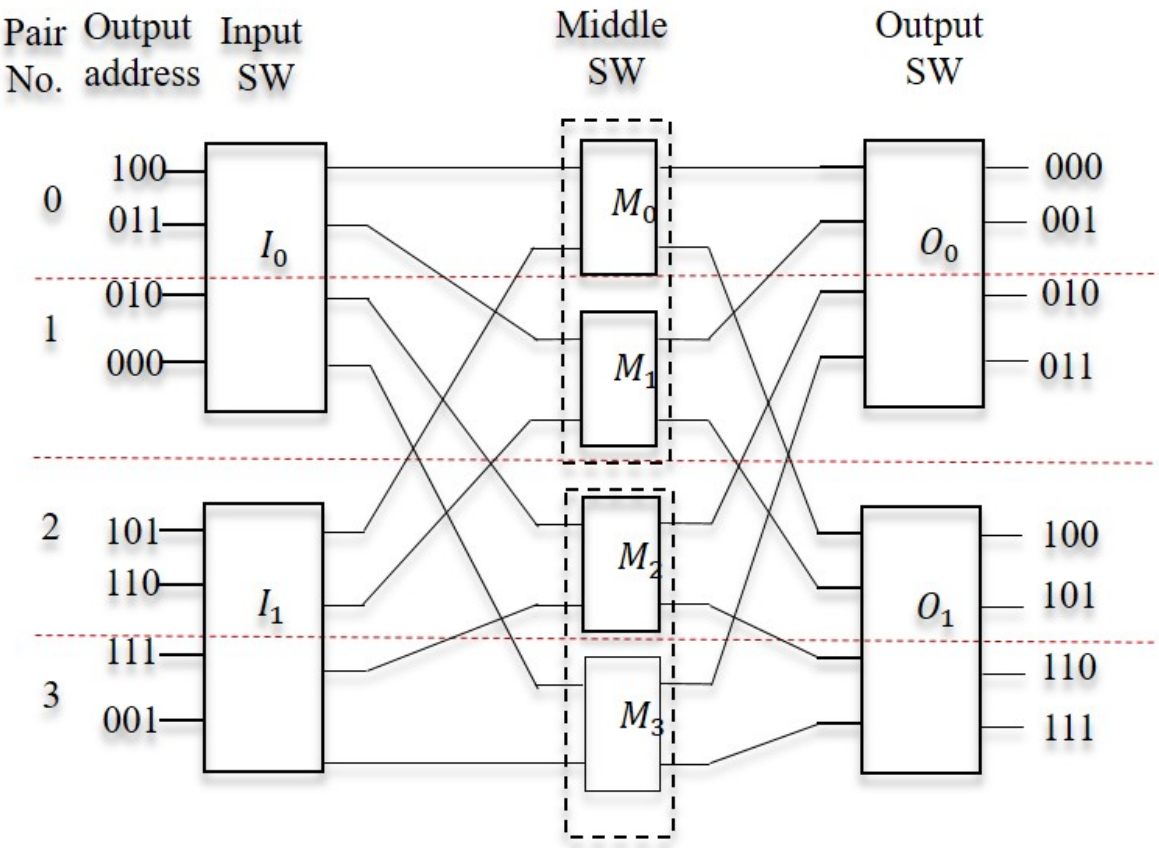


Figure 53: *Proposed routing principle with iterations*

in the $\log_2 N - 1$ cycles.

6.4.2 Hardware Implementation of the proposed algorithm

Here, a brief introduction of the design and operation of parallel processor elements (PE) for the proposed algorithm is introduced. The design is based on the use of parallel and

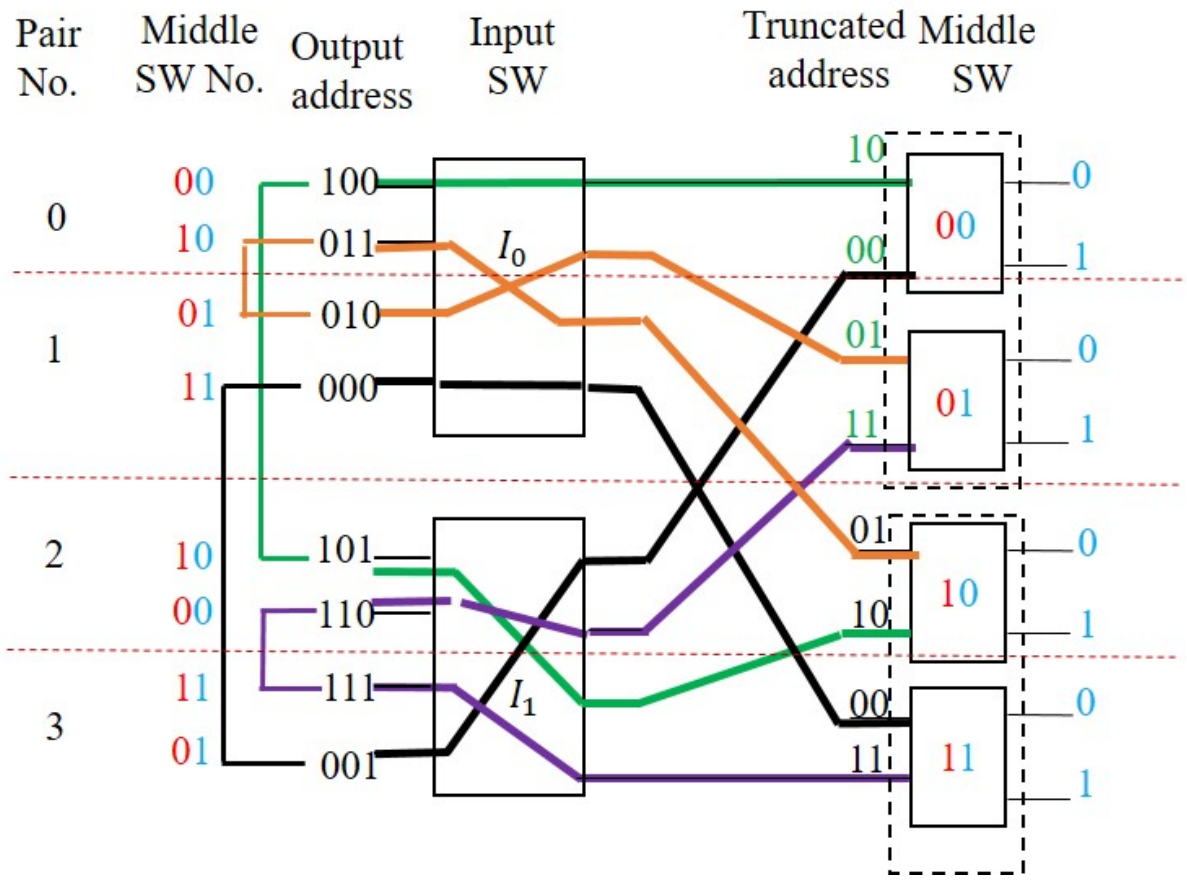


Figure 54: *Routing principle with iterations of the proposed algorithm*

distributed PE designed for the Benes network [37]. The proposed algorithm is recursive in nature and therefore focus is restricted to the first cycle. The hardware implementation of this design principle is as follows;
$N/2$ PEs are interconnected to multiple buses e.g. b_{00} to b_{11} as shown in Fig. 54(a). The first two PEs control the upper virtually divided switch of the first stage and the other two PEs control the lower virtually divided switch. Each of the PE is assigned with a unique binary suffix number and is represented as $PE_{b(p)}$ where $0 \leq p \leq N/2-1$, and $b(p)$ is the binary expression of p. The proposed parallel algorithm consists of three phases.
(i).First phase
The first phase involves searching for a neighbor PE and is performed in parallel. The process is performed to find PEs with the same truncated address. This process begins by a $PE_{b(p)}$ which broadcasts its addresses onto $B_{b(p)}$ in series. The PEs compare their own truncated addresses with the incoming addresses using comparators that are inbuilt in them. As shown in the figure 54 PE_{00} broadcasts its truncated address $\underline{10}$ from its upper inlet to the bus B_{00}. PE_{10} discovers it as a neighbor because it contains $\underline{10}1$ at its upper inlet. The underlined part indicates the range of truncated addresses. The two addresses 100 and 101 both sit on the upper inlets of the two PEs. These two addresses must be routed to separate middle stage switches. As a result, the virtual 2×2 switches must be set in different modes (i.e., bar(0) and the other cross (1)). This gives the link relationship of *'not equal to'*, that is the PE_{00} is *'not equal'* to PE_{10}. However, when PE_{00} broadcasts its $\underline{01}$ from its lower inlet, the PE_{01} discovers it as a neighbor because

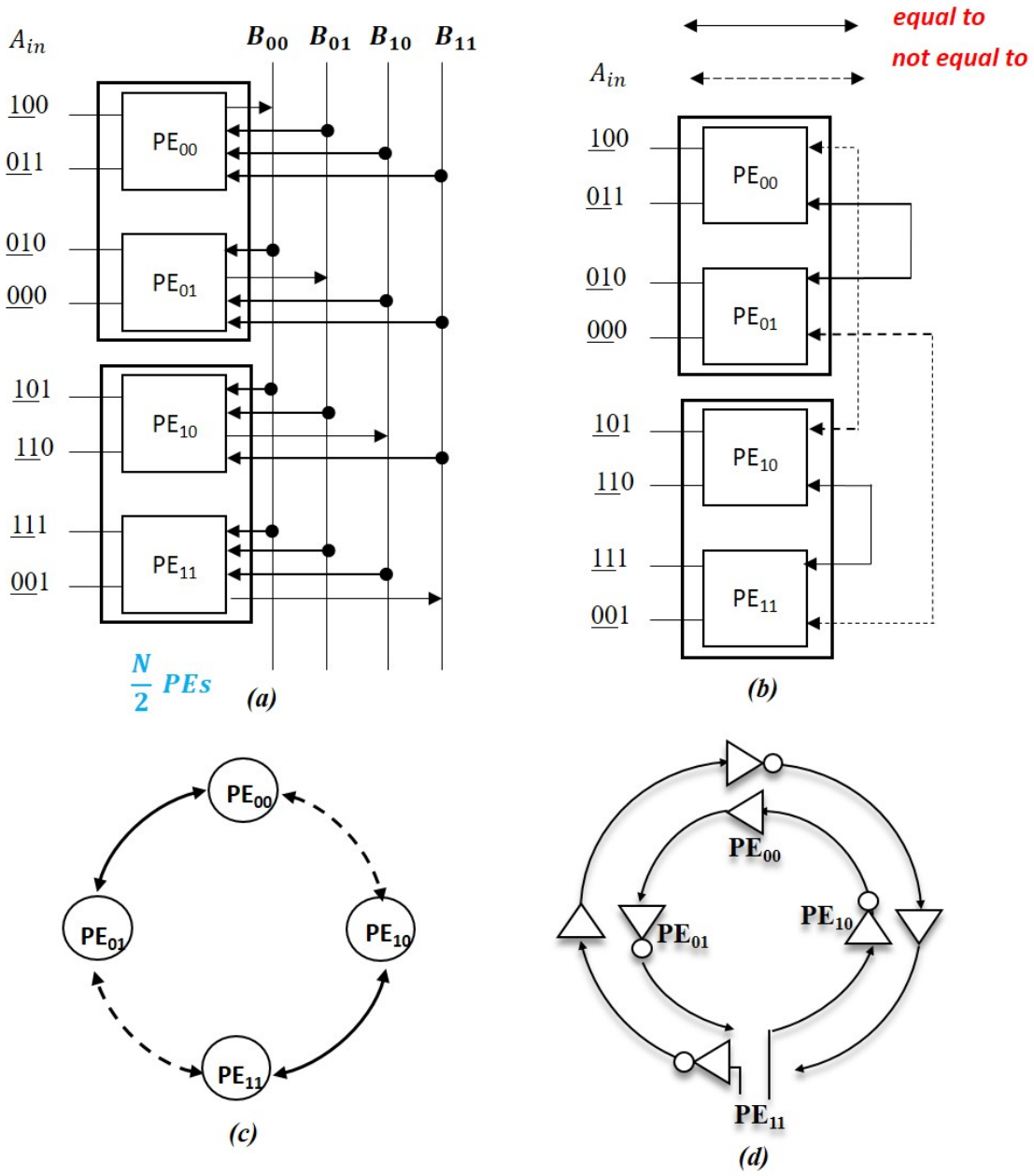


Figure 55: *Arrangement and operation of parallel processing elements (a)PEs connected to bus bars (b) Neighbour search establishment (c) Link relationship status (d)Implementation of link status using Inverting and non inverting gates*

it has 010 on its upper inlet. Here, the state of PE_{00} can be set 'equal to' the state of PE_{01}. The link relation status for all the other PEs is obtained and represented in figure 54 (b) and (c) with solid lines representing the *'equal to'* relation while the dotted line represents *'not equal to'* relation. Practically, the link status relationship between PEs can be implemented in FPGA using inventing and non inverting gates. The process of neighbor search is realized in parallel and the time complexity is that of $O(1)$.

(ii). Second phase

The second phase involves choosing a representative PE. The process to choose a representative PE is an iterative process. It is important to choose a representative element because it acts as a starting point for setting the status of the other PEs from the link

relation status obtained in phase one. At the beginning of the process, each PE is a potential candidate and its status is shown using a double circle in Fig. 55. Each PE is related to the others and a system of relations that exists between them is referred to as *initialization equations*. These equations are solved in [34] in a relatively complex manner. The proposed solution in this study is based on the principle that if we appoint a PE as a representative in the link diagram obtained, and set its status, the status of the other PEs will be generated automatically based on the link status relation in the link diagram. For example, if we choose PE_{11} as the representative element, and set it to a default state such as bar, the status of the other PEs will be generated automatically in the link diagram. The selection of a representative element is done through a tournament process i.e., a PE with a larger binary suffix information y_i will win the tournament to those with a lower suffix. The iterative process begins by each PE broadcasting its binary suffix information to the two neighboring PEs. Each PE then compares its own suffix y_0 with incoming suffixes y_1 and y_2. This process happens consecutively if the number of PEs are more than two. Hence there are three possible cases. (a) If $(y_0 < y_1) \cup (y_0 < y_2)$, then a PE will lose the completion and it will become transparent transferring the information y_1 and y_2 to its opposite neighbors. As can be seen in Fig. 55(b) PE_{00} and PE_{01} loses the tournament and are shown with a dashed line in the figure.

(b) If $(y_0 > y_1) \cap (y_0 > y_2)$, then a PE will remain as a potential representative. Both y_1 and y_2 will be discarded.

(c) Finally, if $y_0 = y_1 = y_2$, then PE becomes a final representative and the process halts as shown in Fig. 55(d).

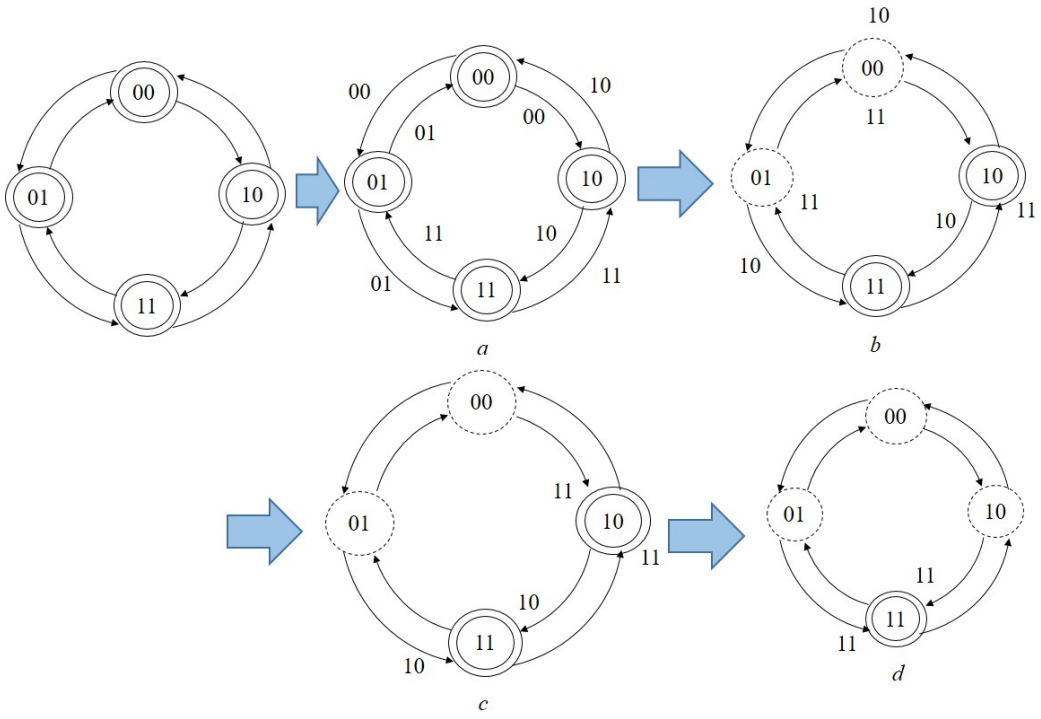


Figure 56: *Iterative procedure for determining the representative PE*

PEs that lose, bypass incoming data through simple switches embedded in the PEs. This means that the bypass delay is negligible. This tournament comparison result is updated and stored using the representative flag (f_r). In order to specify the bus through which the suffix information can be received through, the suffix of the neighbor PEs are referred to. After each iteration, the number of candidates for a representative is reduced to less than half of each iteration. It is observed that the second phase is completed in time of order of at most $O(\log_2 N)$. In general, several loops can exist, and each of those loops will have to specify a representative. Here, for simplicity sake, an example with only a single loop is given.

(iii). Third phase

The third phase is performed to determine the status of each PE and pass address information to the next stage based on the directive of the representative element. As can be seen in Fig. 56(a), PE_{11} sets an initial state of bar(0). This state becomes the trigger for the link diagram to set the other status of the PEs as shown in fig. 56(b). Switch control bits (SCB) are sent to the switch elements in the switch body. These links were already established through buses at the start of the third phase. The inverting and non-inverting gates are configured based on the flag bit status (f_s). Once the status of the PEs is set, the pair of input addresses are transferred to the next group of PEs in the next stage. The third phase completes the process in time of $O(1)$. It is clear that the critical time for the first stage is that of the phase for choosing the representative element i.e. second phase and it is given by $O(\log_2 N)$.

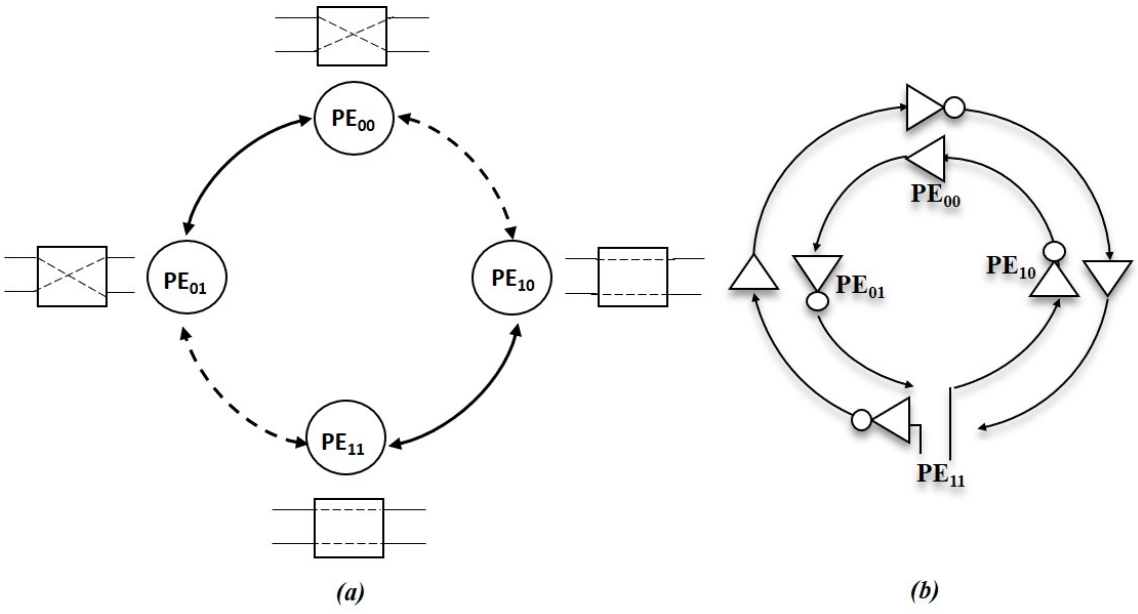


Figure 57: *Determination of link status of each PE (a).Representative PE initially setting its status to bar triggering the other PE status (b). The link status implementation using inverting and non-inverting gates*

6.4.3 FPGA design and experimental results

The TSCN was implemented in FPGA in a similar manner as that of the Benes switch described in section 6.3. Fig. 57 shows the experimental results of the processing time for the first part of the processing with respect to the TSCN size of N.

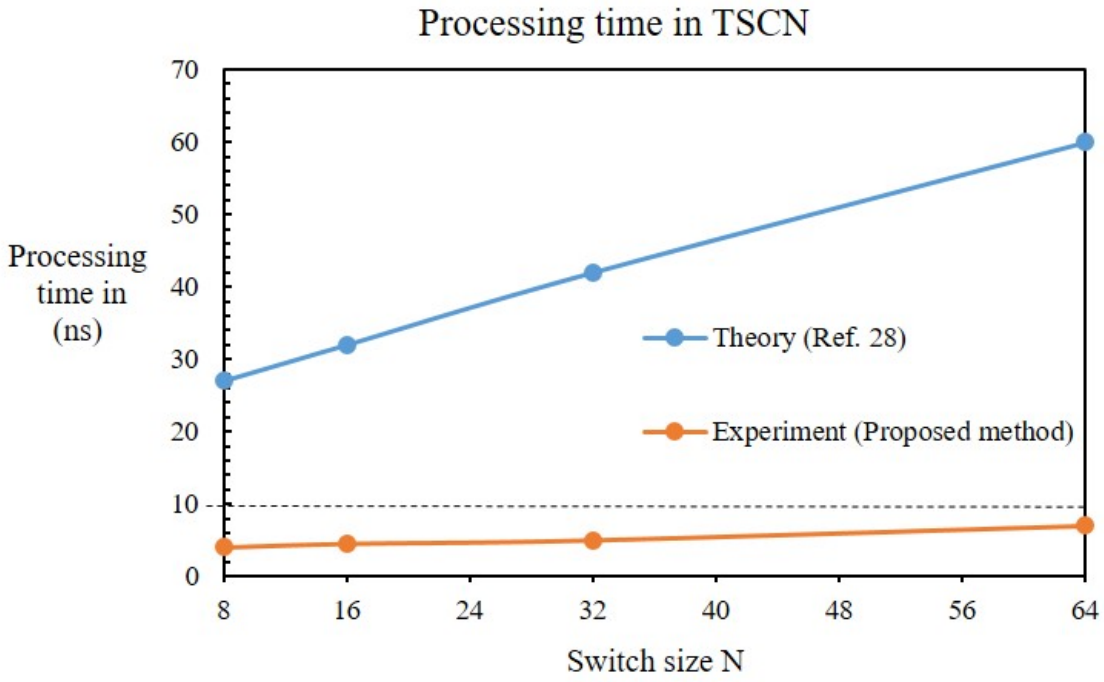


Figure 58: *Experimental processing time for the first part of the algorithm*

Figure 57 above also shows the processing time for the conventional parallel algorithms estimated from reference [29] and the processing time obtained from the proposed method. Note that the $O(1)$ processing time for phases 1 and 3 are neglected. From the results, it can be seen clearly that the proposed algorithm has significantly reduced processing time than the conventional method. Experimentally estimated processing time for the first part of the algorithm is approximated as $O(\log_2 N^{0.16})$. This means that only a single clock cycle of 10 ns is required for switch sizes of upto $N = 128$. The processing time in terms of clock cycles reduces by several times than conventional algorithms. Generally, the processing time of the proposed algorithm will be reduced to $\log_2 N^{\alpha}/2^{t-1}$ at the t-th iteration and t range from 1 to $\log_2 N$ and $\alpha < 1$. α depends on the FPGA device used. Various types of FPGAs have different values. The total time complexity of the proposed algorithm becomes $O(\log_2 N)$ upto a certain switch size N.

6.4.4 Summary

The proposed fast parallel algorithms for setting up TSCN has been implemented in FPGA. The performance of the algorithm was investigated using the prototype built in FPGA consisting of parallel PEs for a switch size ranging from $N = 8$ to $N = 64$. The results obtained demonstrate that the proposed design outperforms the conventional methods because the time complexity of the proposed algorithm is reduced to $O(\log_2 N)$ from $O((\log_2 N)^2)$ up to a certain switch size N due to the asynchronous operation that was introduced.

Chapter 7

Conclusion and Future works

In this dissertation, a new three stage Clos network with modified crossbar switches which has extra inlets and outlets is proposed and investigated.
In chapter 1,

- The need for switching in communication systems is briefly introduced.
- A background on the switching fabrics , their architecture and control algorithms is introduced.

In chapter 2, related works and literature review is given with a focus on

- Conventional Clos structure and its properties.
- Recent routing algorithms in Benes and Clos switches.
- And a summary of the contributions for the Benes and Clos routing proposed in this thesis.

In chapter 3, the proposed design and operation of the new three stage Clos architecture is discussed.

- Firstly, Conventional crossbar switches which has idle ports on the upper and right side are modified and used in the proposed architecture.
- The idle ports are modified and used as extra inlets and outlets, then these are applied to the TSCN to form a new architecture.
- The extra inlets and outlets provides extra routes within the switch thereby increasing the number of routes while maintaining the same cross point count.
- The modified XBS allow signals to flow through them in two directions, i.e., a signal can enter from left traveling to the right and turning bottom or a signal can enter from the right side traveling to the left and turning top.
- This allows a single XBS in the first stage to connect to each of the middle stage XBS using two links as opposed to a single link that exists in the conventional TSCN.
- The extra routes provided by idle ports brings about a reduction in the number of middle stage switches required to maintain both SNB and RNB properties of the TSCN.

In chapter 4, the performance of the new architecture was investigated through computer simulations.

- The connection status of the switch was simulated, i.e. connection or disconnection of a connection request using C programming.
- The XBSs inlets and outlets were represented as arrays.
- By varying the number of middle stage switches, connection routes were searched by prioritization of row/column pairing. If routes were found, a setup was initiated and a new disconnection was performed to test other connection patterns. If no routes were found, it was concluded that blocking happened and the search terminated.
- The number of middle stage switches were increased and the procedure repeated. It was revealed that a new lower bound on m for the RNB property was obtained as $\left\lceil \frac{3n}{4} \right\rceil$ which is smaller by 25% that of the conventional Clos.
- This reduction is brought about by the row/column sharing technique.
- It was further demonstrated that when $m = n$, the number of rearrangements is reduced to one at most regardless of the value of n and r compared to the $r - 1$ rearrangements required in the worst case of the conventional TSCN.
- Further more, a special WSNB was developed from the conventional SNB TSCN. It was demonstrated that the bidirectional WSNB condition can be expressed as $m = \left\lceil \frac{3n}{2} \right\rceil + 1$ which is smaller than $m = 2n$–1 for conventional Clos by 25%.
- The proposed switch is referred to as WSNB because of the row/column sharing that is being followed when setting up calls or connection requests in the switch. The validity of the preposition was also tested through simulations.
- The proposed architecture can be used to reduce the number of middle switches especially in those switches which have larger middle stage switches than their input and output switches. Though the viability of the bidirectional TSCN was shown through simulations, to realize these switches fully, a detailed experimental analysis is necessary. The actual experimental analysis is left for future study.

In Chapter 5, a new design principle of unfolded two stage switches using bidirectional switches was introduced.

- A new architecture composed of Input switch modules (ISM) connected to Output switch modules (OSM) is proposed. Half of the outputs to the switch are taken from the upper side of the OSMs and the other half on bottom part.
- The non-blocking properties were investigated by firstly determining the number of rearrangements and then proposing the number of OSMs required to obtain a strictly non-blocking switch.
- For the first time it has been proposed that UTSNs requires only a total of $m \geq n + 1$ number of middle stage switches to have a strictly non blocking.
- The switch complexity becomes minimum when $n = \sqrt{N/2}$ and saturates at $N^2/2$ as $N \to \infty$.

In Chapter 6, a new parallel routing algorithm was designed based on parallel and distributed PEs to set up permutations in Benes networks.

- The algorithm can handle both full and partial permutations in a unified manner with little overhead time and additional hardware costs.
- The proposed design has a reduced hardware complexity of $O(N(N\log_2 N)^2)$ from that of a sequential algorithm of $O(N^2)$.The reduction is as a result of the distributed architectural design of the proposed method.
- The proposed pipe-lined architecture further reduces the time complexity from that of $O((\log_2 N)^2)$ to $O(\log_2 N)$.
- To increase the processing speed,the iterative search for the representative element in the first part of the algorithm and the DTR in the second part were allowed to proceed asynchronously.This reduced the time complexity to $O(\log_2 N^\alpha)$, where $\alpha < 1$ and α is dependent on the type of FPGA used.
- Only a single clock cycle of 10 ns is required for processing up to a switch of size $N = 128$ in the second phase of the first part.
- The result of this study revealed that the proposed design requires as few as five clocks for the first part only, compared to those of 17 clocks in a recent method.
- The parallel and distributed PE prototype was built in FPGA. Experimental results revealed that the proposed design occupied less FPGA resources about six times less than the conventional method.This method outperforms conventional methods by several times in terms of hardware resource usage and processing time complexities.
- A fast parallel algorithm to set up a three stage Clos network which has a switch size of a power of two was proposed and implemented in FPGA. A similar result obtained confirmed that the proposed method outperforms the conventional methods.

The implementation of the algorithm for larger switches of size $N \geq 128$ is needed for further verification, since in this study the increment was only estimated to increase by a single clock. Another unresolved issue is that switches with a switch size that is not of a power of two have no optimized parallel control algorithm. The proposed algorithm and design will expand the applicability of both the Benes and Clos networks. The fast parallel algorithm can be used to develop a re-configurable switching fabric with nanosecond speed processing.

References

[1] Cisco Visual Networking, "The zettabyte era-trends and analysis, cisco whitepaper," (2017).

[2] J. Y. Hui, *Switching and Traffic Theory for Integrated Broadband Networks*, vol. 91. Springer Science & Business Media, 2012. doi:10.1007/978-1-4615-3264-4.

[3] Q. Cheng, M. Bahadori, M. Glick, S. Rumley, and K. Bergman, "Recent advances in optical technologies for data centers:a review," *Optica*, vol. 5, no. 11, pp. 1354–1370, 2018. doi:10.1364/OPTICA.5.001354.

[4] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center network," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, (New York, NY, USA), pp. 51–62, Association for Computing Machinery, 2009. doi:10.1145/1592568.1592576.

[5] T. Anttalainen, *Introduction to telecommunications Network Engineering*. Artech House, 2003.

[6] B. Parhami, *Introduction to parallel processing: algorithms and architectures*. Springer Science & Business Media, 2006. doi:10.1007/b116777.

[7] W. Kabacinski, C.-T. Lea, and G. Xue, "Guest editorial-50th anniversary of clos networks," *IEEE Communications Magazine*, vol. 41, no. 10, pp. 26–27, 2003. doi:10.1109/MCOM.2003.1235590.

[8] Q. Cheng, M. Bahadori, Y.-H. Hung, Y. Huang, N. Abrams, and K. Bergman, "Scalable microring-based silicon Clos switch fabric with switch-and-select stages," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 25, no. 5, pp. 1–11, 2019. doi:10.1109/JSTQE.2019.2911421.

[9] L. Wang, T. Ye, and T. T. Lee, "A parallel route assignment algorithm for fault-tolerant clos networks in OTN switches," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 5, pp. 977–989, 2018. doi:1109/TPDS.2018.2880782.

[10] O. T. Sule and R. Rojas-Cessa, "TRIDENT: A Load-Balancing Clos-Network Packet Switch With Queues Between Input and Central Stages and In-Order Forwarding," *IEEE Transactions on Communications*, vol. 67, no. 10, pp. 6885–6896, 2019. doi:10.1109/TCOM.2019.2926730.

[11] D. M. Marom, P. D. Colbourne, A. Derrico, N. K. Fontaine, Y. Ikuma, R. Proietti, L. Zong, J. M. Rivas-Moscoso, and I. Tomkos, "Survey of photonic switching architectures and technologies in support of spatially and spectrally flexible optical networking," *Journal of Optical Communications and Networking*, vol. 9, no. 1, pp. 1–26, 2017. doi:10.1364/JOCN.9.000001.

[12] S. Ohta, "The number of rearrangements for Clos networks–new results," *Theoretical Computer Science*, vol. 814, pp. 106–119, 2020. doi:10.1016/j.tcs.2020.01.2018.

[13] C. Clos, "A study of non-blocking switching networks," *Bell System Technical Journal*, vol. 32, no. 2, pp. 406–424, 1953. doi: 10.1002/j.1538-7305.1953.tb01433.x.

[14] M. Paull, "Reswitching of connection networks," *Bell System Technical Journal*, vol. 41, no. 3, pp. 833–855, 1962. doi:10.1002/j.1538-7305.1962.tb00478.x.

[15] D. Smith, "Lower bound on the size of a 3-stage wide-sense nonblocking network," *Electronics Letters*, vol. 13, no. 7, pp. 215–216, 1977. doi:10.1049/el:19770156.

[16] K.-H. Tsai, D.-W. Wang, and F. Hwang, "Lower bounds for wide-sense nonblocking Clos network," *Theoretical computer science*, vol. 261, no. 2, pp. 323–328, 2001. doi:10.1016/S0304-3975(00)00147-X.

[17] F. Chang, J. Guo, F. K. Hwang, and C. Lin, "Wide-sense nonblocking for symmetric or asymmetric 3-stage Clos networks under various routing strategies," *Theoretical computer science*, vol. 314, no. 3, pp. 375–386, 2004. doi:10.1016/j.tcs.2003.12.021.

[18] A. Jajszczyk and G. Jekel, "A new concept-repackable networks," *IEEE transactions on communications*, vol. 41, no. 8, pp. 1232–1237, 1993. doi:10.1109/26.231967.

[19] H. Obara, "Strictly non-blocking three-quarter crossbar switch with simple switch control," *Electronics Letters*, vol. 52, no. 25, pp. 2051–2053, 2016. doi:10.1049/el.2016.3561.

[20] K. Tanizawa, K. Suzuki, K. Ikeda, S. Namiki, and H. Kawashima, "Novel PILOSS Port Assignment for Compact Polarization-Diversity Si-Wire Optical Switch," in *Optical Fiber Communication Conference*, p. M3I.6, Optical Society of America, 2016. doi:10.1364/OFC.2016.M3I.6.

[21] H. Obara, "Cascaded versus parallel architectures of two-stage optical crossbar switches with an extra set of inputs and outputs," *IET Optoelectronics*, vol. 12, no. 4, pp. 196–201, 2018. doi:10.1049/iet-opt.2017.0096.

[22] K. Goossens, L. Mhamdi, and I. V. Senin, "Internet-router buffered crossbars based on networks on chip," in *2009 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools*, pp. 365–374, IEEE, 2009. doi:10.1109/DSD.2009.211.

[23] L. Mhamdi, K. Goossens, and I. V. Senin, "Buffered crossbar fabrics based on networks on chip," in *2010 8th Annual Communication Networks and Services Research Conference*, pp. 74–79, IEEE, 2010. doi:10.1109/CNSR.2010.18.

[24] F. Hassen and L. Mhamdi, "Congestion-aware multistage packet-switch architecture for data center networks," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2016. doi:10.1109/GLOCOM.2016.7841681.

[25] F. Hassen and L. Mhamdi, "High-capacity clos-network switch for data center networks," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–7, 2017. doi:10.1109/ICC.2017.7997147.

[26] W. Nicolas, "MEMS Jumpstart Series: Creating an Optical Switch," *Eureka magazine, Design Engineering Whitepapers*, vol. Whitepaper, pp. 1–8, 2016.

[27] T. S. El-Bawab, *Optical switching*. Springer Science & Business Media, 2008.

[28] A. Waksman, "Corrigendum: A Permutation Network," *Journal of the ACM (JACM)*, vol. 15, no. 2, p. 340, 1968. doi:10.1145/321439.321449.

[29] E. Lu and S. Zheng, "Parallel routing algorithms for nonblocking electronic and photonic switching networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 8, pp. 702–713, 2005. doi::10.1109/TPDS.2005.95.

[30] G. F. Lev, N. Pippenger, and L. G. Valiant, "A fast parallel algorithm for routing in permutation networks," *IEEE transactions on Computers*, vol. 100, no. 2, pp. 93–100, 1981. doi:10.1109/TC.1981.6312171.

[31] D. Nassimi and S. Sahni, "Parallel algorithms to set up the Benes permutation network," *IEEE Computer Architecture Letters*, vol. 31, no. 02, pp. 148–154, 1982. doi:10.1109/TC.1982.1675960.

[32] T. Jain and K. Schneider, "Routing partial permutations in interconnection networks based on radix sorting," in *2018 13th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pp. 1–10, IEEE, 2018. doi:10.1109/ReCoSoC.2018.8449372.

[33] C.-Y. Lee and A. Y. Oruç, "A fast parallel algorithm for routing unicast assignments in Benes networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 3, pp. 329–334, 1995. doi:10.1109/71.372780 .

[34] T. T. Lee and S.-Y. Liew, "Parallel routing algorithms in Benes-Clos networks," *IEEE transactions on communications*, vol. 50, no. 11, pp. 1841–1847, 2002. doi:10.1109/TCOMM.2002.805258.

[35] Y. Jiang and M. Yang, "Hardware Implementation of Parallel Algorithm for Setting Up Benes Networks," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, p. 10, The Steering Committee of The World Congress in Computer Science, Computer , 2016. doi:.

[36] Y. Jiang and M. Yang, "Hardware design of parallel switch setting algorithm for Benes networks," *International Journal of High Performance Systems Architecture*, vol. 7, no. 1, pp. 26–40, 2017. doi:https://doi.org/10.1504/IJHPSA.2017.083645.

[37] Y. Kai, K. Hamada, Y. Miao, and H. Obara, "Design of partially-asynchronous parallel processing elements for setting up Benes networks in $O(\log_2 N)$ time," in *2009 International Conference on Photonics in Switching*, pp. 1–2, IEEE, 2009. doi:10.1109/PS.2009.5307812.

[38] S. Andresen, "The looping algorithm extended to base 2^t rearrangeable switching networks," *IEEE Transactions on communications*, vol. 25, no. 10, pp. 1057–1063, 1977. doi:10.1109/TCOM.1977.1093753.

[39] S.-Q. Zheng, A. Gumaste, and E. Lu, "A practical fast parallel routing architecture for clos networks," in *Proceedings of the 2006 ACM/IEEE symposium on Architecture for networking and communications systems*, pp. 21–30, 2006. doi:10.1145/1185347.1185351 .

[40] H. Obara, "Reduced crossbar switch with minimum number of switching cells," *Electronics Letters*, vol. 44, no. 14, pp. 888–889, 2008. doi:10.1049/el:20083528.

[41] H. Obara, "Design of optical multi/demultiplexers composed of bidirectional 2 × 2 switch elements for reducing component count," *Electronics Letters*, vol. 51, no. 15, pp. 1182–1184, 2015. doi: 10.1049/el.2015.1526.

[42] V. E. Beneš, *Mathematical theory of connecting networks and telephone traffic*. Academic press, 1965.

[43] W. Kabacinski, *Nonblocking electronic and photonic switching fabrics*. Springer Science & Business Media, 2005.

[44] M. Jinno, "Spatial channel network (SCN): opportunities and challenges of introducing spatial bypass toward the massive SDM era," *Journal of Optical Communications and Networking*, vol. 11, no. 3, pp. 1–14, 2019. doi: 10.1364/JOCN.11.000001.

[45] M. Fiorani, M. Tornatore, J. Chen, L. Wosinska, and B. Mukherjee, "Spatial division multiplexing for high capacity optical interconnects in modular data centers," *Journal of Optical Communications and Networking*, vol. 9, no. 2, pp. A143–A153, 2017. doi:10.1364/JOCN.9.00A143.

[46] M. Jinno, "Architectures of spatial add/drop multiplexer and cross-connect for spatial channel networks," in *2020 International Conference on Optical Network Design and Modeling (ONDM)*, pp. 1–3, 2020. doi:10.23919/ONDM48393.2020.9132997.

[47] I. White, M. Ding, A. Wonfor, Q. Cheng, and R. Penty, "High port court switch architectures for data center applications," in *Photonic Networks and Devices*, pp. NeW1B–2, Optical Society of America, 2017. doi: 10.1364/NETWORKS.2017.NeW1B.2.

[48] C.-T. Lea, "Expanding the switching capabilities of optical crossconnects," *IEEE Transactions on Communications*, vol. 53, no. 11, pp. 1940–1944, 2005. doi:10.1109/TCOMM.2005.858662.

[49] D. C. Opferman and N. T. Tsao-Wu, "On a class of rearrangeable switching networks part i: Control algorithm," *The Bell System Technical Journal*, vol. 50, no. 5, pp. 1579–1600, 1971. doi:10.1002/j.1538-7305.1971.tb02569.x.

[50] K. Y. Lee, "A new Benes network control algorithm," *IEEE Transactions on Computers*, vol. 100, no. 6, pp. 768–772, 1987. doi: 10.1109/TC.1987.1676970.

[51] P. Hall, "On representatives of subsets," in *Classic Papers in Combinatorics*, pp. 58–62, Springer, 2009. doi: 10.1007/978-0-8176-4842-8_4.

[52] F. K.-M. Hwang, *The Mathematical Theory Of Nonblocking Switching Networks*, vol. 15 of *Series On Applied Mathematics*. World Scientific Publishing Company.

[53] C.-L. Wu and T.-Y. Feng, "On a class of multistage interconnection networks," *IEEE transactions on Computers*, vol. 100, no. 8, pp. 694–702, 1980. doi:10.1109/TC.1980.1675651.

[54] X. Tang, E. Giacomin, B. Chauviere, A. Alacchi, and P.-E. Gaillardon, "OpenFPGA: An Open-Source Framework for Agile Prototyping Customizable FPGAs," *IEEE Micro*, vol. 40, no. 4, pp. 41– 48, 2020. doi:10.1109/MM.2020.2995854.

[55] "Xilinx: Spartan-6 family overview. https://www.xilinx.com/support/ documentation/data_sheets/ds160.pdf, (2020). accessed december 2020," pp. 1–20, 2020. doi:https://www.xilinx.com/support/documentation/datasheetds160.pdf.