ABSTRACT

Title of dissertation: RANDOM ROUTING AND CONCENTRATION IN QUANTUM SWITCHING NETWORKS

Rahul Ratan, Doctor of Philosophy, 2008

Dissertation directed by: Professor A. Yavuz Oruç
Department of Electrical and Computer Engineering

Flexible distribution of data in the form of quantum bits or qubits among spatially separated entities is an essential component of envisioned scalable quantum computing architectures. Accordingly, we consider the problem of dynamically permuting groups of quantum bits, i.e., qubit packets, using networks of reconfigurable quantum switches.

We demonstrate and then explore the equivalence between the quantum process of creation of packet superpositions and the process of randomly routing packets in the corresponding classical network. In particular, we consider an $n \times n$ Baseline network for which we explicitly relate the pairwise input-output routing probabilities in the classical random routing scenario to the probability amplitudes of the individual packet patterns superposed in the quantum output state.

We then analyze the effect of using quantum random routing on a classically non-blocking configuration like the Beneš network. We prove that for an $n \times n$ quantum Beneš network, any input packet assignment with no output contention

is probabilistically self-routable. In particular, we prove that with random routing on the first $(\log n - 1)$ stages and bit controlled self-routing on the last $\log n$ stages of a quantum Beneš network, the output packet pattern corresponding to routing with no blocking is always present in the output quantum state with a non-zero probability. We give a lower bound on the probability of observing such patterns on measurement at the output and identify a class of 2^{n-1} permutation patterns for which this bound is equal to 1, i.e., for all the permutation patterns in this class the following is true: in *every* pattern in the quantum output assignment all the valid input packets are present at their correct output addresses.

In the second part of this thesis we give the complete design of quantum sparse crossbar concentrators. Sparse crossbar concentrators are rectangular grids of simple 2×2 switches or crosspoints, with the switches arranged such that any k inputs can be connected to some k outputs. We give the design of the quantum crosspoints for such concentrators and devise a self-routing method to concentrate quantum packets. Our main result is a rigorous proof that certain crossbar structures, namely, the fat-slim and banded quantum crossbars allow, without blocking, the realization of all concentration patterns with self-routing.

In the last part we consider the scenario in which quantum packets are queued at the inputs to an $n \times n$ quantum non-blocking switch. We assume that each packet is a superposition of m classical packets. Under the assumption of uniform traffic, i.e., any output is equally likely to be accessed by a packet at an input we find the minimum value of m such that the output quantum state contains at least one packet pattern in which no two packets contend for the same output. Our cal-

culations show that for $m = 9$ the probability of a non-contending output pattern occurring in the quantum output is greater than 0.99 for all n up to 64.

RANDOM ROUTING AND CONCENTRATION IN QUANTUM SWITCHING NETWORKS

by

Rahul Ratan

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2008

Advisory Committee:
Professor A. Yavuz Oruç, Chair/Advisor
Professor Richard La
Professor Prakash Narayan
Professor Robert Newcomb
Professor G. W. Stewart, Dean's Representative

Acknowledgments

It is said that that all great labor is 99% perspiration and 1% inspiration. This page is an attempt to give credit where it is due for that intangible but all important 1% portion of my endeavour. First of all I am deeply indebted to my adviser Prof. Yavuz Oruç for his constant support and encouragement through the course of my graduate studies. He has not only been an academic adviser but a mentor to me in spheres of life unrelated to my intellectual journey. Thank you sir for all that you have done. I am also thankful to the other members of my Dissertation committee: Prof. Newcomb, Prof. Narayan, Prof. La and Prof. Stewart for devoting time towards my cause.

A person is measured by his friends and the company he/she keeps. I have been fortunate to have friends who meet and far exceed any standard of measurement. They have all made my journey as a graduate student an unforgettable and life-affirming experience. Special thanks are due to my fellow researcher Manish Shukla for all the discussions related to research and otherwise in which he was never selfish in sharing any knowledge. My other close friends and room mates at University of Maryland who will always be fondly remembered: Gunjan Sharma, Sudarshana Koushik, Archana Anibha, Ravi Tandon, Anuj Rawat, Shaju John, Rajagopal, Srikanth, Pavan Turaga, Abhinav Gupta, Swati Jarial, Vishal, Abhishek, Rakesh, all of you have enriched me greatly with your friendship and thoughtfulness. I also want to acknowledge my former wing-mates from IIT Kanpur who have maintained and nurtured the wonderful relationship we forged in our

younger days.

In my parents I have an unwavering foundation of support which I can always rely and count upon. They have provided the bedrock on which my life stands, I can never repay their debt. I will always treasure their patience and willingness to let me discover my own path in life. My elder sister, Swati, has also been a wellspring of constant love and encouragement. Both she and her husband Jyothis have welcomed me unreservedly whenever I have called on them. I cannot thank them enough for loving me unconditionally.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Quantum computation and its related field of quantum information science are recent areas of research which have rapidly gained in prominence from their origins around three decades ago. Originally conceived as a means to simulate quantum mechanical systems [1], which are hard to model on conventional or classical computers, the quantum computation and information processing models have led to discovery of a large body of fundamental results which have spawned new areas of application of quantum mechanical principles like quantum cryptography and quantum error correcting codes [2, 3]. The uniqueness or power of quantum systems surprisingly comes from the inherent randomness of quantum entities and the non-local interactions between them. These properties as manifested in the form of quantum parallelism and entanglement have been used in finding efficient solutions for classically intractable problems. The interest in this area received a major boost with the discovery of some seminal algorithms which have demonstrated that quantum systems can be used to solve interesting exponential complexity problems with speedups that are impossible in classical computing. The two most famous examples include Shor's polynomial time algorithm for finding prime factors of a composite number [4] and Grover's search algorithm that can search an unstructured database with n elements in $O(\sqrt{n})$ time steps.

Other more recent examples include adiabatic solution of optimization problems [5], Pell's equation [6] and Gauss sums [7].

Mirroring the advances on the theoretical front, practical implementation of quantum information processing systems has also garnered a lot of attention. Early stage quantum computers have utilized many different technologies for performing computation: trapped ions and molecules in solution [8], Josephson junctions [9], photons [10], bulk spin NMR [11] and phosphorous impurities in silicon [12]. Of these proposals, only those building on a solid-state platform are expected to provide the scalability required to achieve a useful computational substrate. Several recent promising proposals for large scale quantum computer architectures based on solid-state silicon are a step in this direction [9, 12]. As quantum systems are scaled up and the number of components grows transporting quantum data becomes a critical requirement. This issue of quantum data transport has been recognized as a particularly critical requirement in upcoming silicon-based quantum computing technologies [13, 14].

Spatially distributed components introduce the need for *quantum wires* over which quantum data can be transmitted but building quantum wires and transferring quantum bits (qubits) is a non-trivial operation as, in general, quantum bits cannot be copied, which is a consequence of the no-cloning theorem [15]. Proposals for building quantum wires in solid state technologies include the quantum swapping and teleportation based architectures in [13, 14, 16]. The high cost of such wires implies that the $O(n^2)$ wires needed to interconnect n quantum devices can be a major bottleneck in implementing quantum systems.

This cost can be greatly reduced by using efficient switching network designs. The basic premise of this idea is to use arrangements of reconfigurable switches with input quantum bits on their own quantum wires and then route them to the required destination. These switches are represented using quantum circuits composed of quantum gates. In addition to reducing wire count, reconfigurable quantum switches can form integral parts of the quantum data distribution system in envisioned architectures for scalable quantum computing. For example, in the Quantum Logic Array (QLA) microarchitecture proposed in [17], the high-level quantum computer structure consists of logical qubits connected with a programmable communication network in which integrated switch islands are used to redirect quantum data from nearby logical qubits.

The first design of a network for permuting individual quantum bits was given by Tsai and Kuo [18]. They gave a method to map a decomposition of any given permutation in terms of disjoint cycles and transpositions onto a quantum switch circuit which realizes that particular permutation. Since the circuit realizes only one permutation, it needs to be made anew for any new permutation to be realized. The first quantum switch network using reconfigurable switches to route groups of quantum bits (which comprise quantum bit packets) was given by Shukla et. al. in [19, 20]. This quantum network can permute quantum information packets between its n inputs and n outputs. It was shown that this network realizes $n^{n/2}$ permutations and can be used to resolve blocking when transmitting classical packets by creating a superposition of packets whenever they contend for the same wire in the network. This phenomenon reflects the give-and-take between quantum

and classical switch designs. While on the one hand classical structures can reduce the wire count for quantum systems, on the other hand, quantum properties like superposition or quantum parallelism can be harnessed to address classical problems like blocking when switching is done in the quantum domain. Recently, Cheng and Wang [21] have proposed a quantum merge sorting based network that can route all $n!$ permutations using $O(n \log^3 n)$ gates. Switching network configurations suitable for quantum networks have also been identified [22].

Quantum computation exploits the ability for a single quantum bit, a qubit, which can be implemented by the polarization states of a photon or the spin of a single atom, to exist in a superposition of the binary "0" and "1" states. With n qubits a quantum computer can be in 2^n unique states at any given time. These states can be inter-correlated such that a single logic gate can act on all possible 2^n states at once. This is the source of quantum parallelism.

In classical switching networks packet assignments from the n inputs to the n outputs can only be issued one assignment at a time even though they can be overlapped by pipelining them. For quantum networks on the other hand, packets are composed of quantum bits and they can represent a superposition of many possible assignment patterns of packets that may be routed through the network all at once as a consequence of quantum parallelism. It is this parallel aspect of routing in quantum networks that we analyze in this thesis.

In the first part of this thesis we analyze the output state of quantum switch networks which are used to realize ordered connections when superpositions of packets are created in a controlled fashion at the internal switches in the network.

We first define the concepts of quantum packets, quantum assignments and non-blocking networks in terms of superpositions. For any quantum switch configuration we show the equivalence between the quantum process of creation of packet superpositions and the process of randomly routing packets in the corresponding classical network. In particular, we give the example of the Baseline network, where we use the regular connection structure and the self-routing property of this network to explicitly relate the pairwise input-output routing probabilities in the classical random routing scenario to the probability amplitudes of the individual packet patterns superposed in the quantum output state.

We then analyze the effect of using quantum random routing on a classically non-blocking configuration like the Beneš network. We prove that for a $n \times n$ quantum Beneš network, $n = 2^p$, any input packet assignment with no output contention is probabilistically self-routable. In particular, we prove that given any non-contending input assignment pattern with random routing on the first $\log_2 n - 1$ stages and bit controlled self-routing on the last $\log_2 n$ stages of a quantum Beneš network, there exist in the quantum output assignment patterns with non-zero probability amplitude, in which, all the valid input packets are present at their correct output addresses. Additionally, we give a lower bound on the probability of observing such patterns at the outputs upon measurement.

We also identify a class of permutation patterns for which this bound is equal to 1, i.e., for all the permutation patterns in this class the following is true: in *every* pattern in the quantum output assignment all the valid input packets are present at their correct output addresses.

In the second part of this thesis we give the complete design of quantum sparse crossbar concentrators. Concentrators are connectors used to realize unordered connections, i.e., the concentration function is defined with respect to connecting a subset of inputs to some subset of outputs with the matchings within the subsets being irrelevant. Sparse crossbar concentrators are $m \times n$ rectangular grids of simple 2×2 switches or crosspoints, with the switches arranged such that any $k \leq n$ inputs can be connected to some $k \leq m \leq n$ outputs. The ordering of the connections within the sets is not important. The minimum crosspoint count for such structures is $(n - m + 1)m$ [23]. Optimal arrangements of crosspoints which meet this lower bound and allow concentration are also well-known [24, 25].

A concentrator, in general, is a rectangular structure in which the number of inputs, n, is not equal to the number of outputs, m. This implies that the input-output mappings are not reversible. This is a problem for quantum implementation as all quantum operations are reversible. Our first contribution is a method to make a sparse crossbar concentrator square so that it can be implemented in the quantum domain using quantum switches or crosspoints. We then present a method to self-route packets on such concentrators which greatly simplifies routing and makes a quantum implementation more feasible as no classical external control is required. Our main result is a rigorous proof that certain crossbar structures, namely, the fat-slim and banded quantum crossbars allow, without blocking, the realization of all valid concentration patterns when self-routing is done.

In the last part we consider the scenario in which quantum packets are queued at the inputs to an $n \times n$ quantum non-blocking switch. We assume that each

packet is a superposition of m classical packets. m is a parameter of the quantum system which we want to optimize. Under the assumption of uniform traffic, i.e., any output is equally likely to be accessed by a packet at an input we find the minimum value of m such that the output quantum state contains at least one packet pattern in which no two packets contend for the same output. Our calculations show that this value is insensitive to variations in n, indeed, for $m = 9$ the probability of a non-contending output pattern occurring in the quantum output is greater than 0.99 for all n upto 64.

The rest of the dissertation is organized as follows: in Section 1.1 and Section 1.2 we give brief introductions to quantum circuits and switching networks respectively. Chapter 2 contains results related to random routing on quantum Baseline and quantum Beneš networks. In Chapter 3 we give the design of quantum sparse crossbar concentrators. Chapter 4 gives the bounds on size of superpositions on input quantum packets for non-blocking networks. Chapter 5 relates the conclusions and future work.

1.1 Quantum Circuits

In this section we give a brief description of basic concepts related to quantum information, quantum circuits and the quantum gates which are used to manipulate such information.

1.1.1 Qubits

The indivisible unit of classical information is the bit: an object that can take either one of two values: 0 or 1. The corresponding unit of quantum information is the quantum bit or *qubit*. Unlike a classical bit a qubit can take values which are in some sense a combination of the values 0 and 1, i.e., it can be simultaneously be both 0 and 1. Formally, a qubit's state is represented as a unit vector in a two-dimensional complex Hilbert space and can be expressed as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \quad |\alpha|^2 + |\beta|^2 = 1;\ \alpha, \beta \in \mathbb{C} \tag{1.1}$$

The vectors $|0\rangle$ and $|1\rangle$ constitute an orthonormal basis for this space. These two vectors are referred to as the computational basis vectors. We can perform a measurement that projects the state of this qubit onto the computational basis, i.e., the measurement projects $|\psi\rangle$ onto the basis $\{|0\rangle\,, |1\rangle\}$. The outcome of the measurement is not deterministic— the probability that we observe the result to be $|0\rangle$ is $|\alpha|^2$ and the probability that we observe the result to be $|1\rangle$ is $|\beta|^2$. α and β are referred to as the probability amplitudes of the states $|0\rangle$ and $|1\rangle$ respectively.

The state of an n-qubit system can be expressed as a vector in a complex Hilbert space of dimension 2^n. This 2^n-dimensional space is a tensor product of the n two-dimensional spaces representing individual qubits. The orthonormal basis for this space can be chosen as the states in which each qubit has a definite value, either $|0\rangle$ or $|1\rangle$. A general normalized vector representing an n-qubit state can be expanded in this basis as

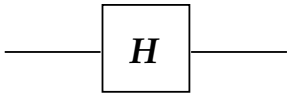
$$
\begin{aligned}
|\psi\rangle &= \alpha_0 |00\cdots00\rangle + \alpha_1 |00\cdots01\rangle + \cdots + \alpha_{2^n-2} |11\cdots10\rangle + \alpha_{2^n-1} |11\cdots11\rangle \\
&= \sum_{i=0}^{2^n-1} \alpha_i |i\rangle \qquad (1.2)
\end{aligned}
$$

where we have associated with each string the number that it represents in binary notation, ranging in value from 0 to $2^n - 1$. Here the α_i's are complex numbers satisfying $\sum_i |\alpha_i|^2 = 1$. A measurement of all n qubits by projection of each onto the $\{|0\rangle, |1\rangle\}$ basis, yields the outcome $|i\rangle$ with probability $|\alpha_i|^2$ [26]. The state of a multiple qubit system is given by the tensor product of the individual state vectors. For example, the combined state of the system of two qubits ψ_1 and ψ_2 where the states of the two qubits are $|\psi_1\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $|\psi_2\rangle = \frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle$ is given by the tensor product:

$$
\begin{aligned}
|\psi_1\rangle \otimes |\psi_2\rangle &= \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle\right) \\
\Rightarrow |\psi_1\rangle |\psi_2\rangle &= \frac{1}{\sqrt{8}}(|0\rangle \otimes |0\rangle) + \frac{\sqrt{3}}{\sqrt{8}}(|0\rangle \otimes |1\rangle) + \frac{1}{\sqrt{8}}(|1\rangle \otimes |0\rangle) + \frac{\sqrt{3}}{\sqrt{8}}(|1\rangle \otimes |1\rangle) \\
&= \frac{1}{\sqrt{8}}(|0\rangle |0\rangle) + \frac{\sqrt{3}}{\sqrt{8}}(|0\rangle |1\rangle) + \frac{1}{\sqrt{8}}(|1\rangle |0\rangle) + \frac{\sqrt{3}}{\sqrt{8}}(|1\rangle |1\rangle) \\
&= \frac{1}{\sqrt{8}}|00\rangle + \frac{\sqrt{3}}{\sqrt{8}}|01\rangle + \frac{1}{\sqrt{8}}|10\rangle + \frac{\sqrt{3}}{\sqrt{8}}|11\rangle \qquad (1.3)
\end{aligned}
$$

where $\otimes$ denotes the tensor product, and we use the convention $|x\rangle \otimes |y\rangle = |x\rangle |y\rangle$. Further, if $|x\rangle$ and $|y\rangle$ are basis state vectors then $|x\rangle |y\rangle = |xy\rangle$.

This concept of the probabilistic projection of a combined state onto one or the other basis vectors on measurement captures the basic principle of quantum theory which posits that the act of acquiring information about the state of a physical system inevitably disturbs the state of the system. The trade-off between acquiring



$$|0\rangle \rightarrow \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$|1\rangle \rightarrow \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Figure 1.1: Hadamard gate.

information and creating a disturbance is related to quantum randomness. It is because the outcome of a measurement has a random element that we are unable to infer the initial state of the system from the measurement alone.

1.1.2 Quantum Gates

The state of qubits is transformed using quantum gates and circuits composed of such gates. The quantum gate formalism was first proposed by Deutsch [27]. A quantum gate is a linear, unitary transformation on the space of qubit state vectors. The unitary nature of these transformations implies that quantum gates are *reversible*, i.e., given the state of qubits at the output of a gate, the input state can be uniquely determined. The unitarity also implies that the gates preserve the norm of the input state which amounts to preserving probability. These requirements of reversibility and norm preservation are basic axioms of quantum theory. An example of a single qubit gate is the Hadamard gate, H, (Figure 1.1) which transforms the basis vectors $|0\rangle$ and $|1\rangle$ as

$$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),\ |1\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{1.4}$$

In the above mapping we say that the basis vectors $|0\rangle$ and $|1\rangle$ are put in an equal superposition by the Hadamard gate as after transformation the probability of observing either of the basis vectors at the output is equal to $1/2$. Thus, the Hadamard gate can be considered a quantum randomizer which takes a 0 or 1 bit and transforms it so that the output is either 0 or 1 with probability $1/2$ [28].

A gate is completely specified by the mapping it performs on the basis vectors as all the rest of the input states can be represented as vectors which are a linear combination of these basis vectors. In the case of the Hadamard gate this means that an input qubit in the general state $\alpha |0\rangle + \beta |1\rangle$ would be transformed to $\frac{\alpha}{\sqrt{2}}(|0\rangle + |1\rangle) + \frac{\beta}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{\alpha+\beta}{\sqrt{2}} |0\rangle + \frac{\alpha-\beta}{\sqrt{2}} |1\rangle$.

An n-bit quantum circuit can simultaneously act on all the 2^n components of the input state and transform them according to some specified mapping at once. This is the source of massive *quantum parallelism*. To make this more clear, suppose we are interested in finding the properties of a function $f(i)$ which takes the n-bit binary string i as input. The table of values for $f(i)$ is of size 2^n and is clearly infeasible to calculate for large n. But, with a quantum computer, U_f that acts according to

$$|i\rangle |0\rangle \xrightarrow{U_f} |i\rangle |f(i)\rangle \tag{1.5}$$

When we write any two qubit states side-by-side, it means we are taking a tensor product, thus $|i\rangle |0\rangle = |i\rangle \otimes |0\rangle$. We can put the input register consisting of the

qubits corresponding to i in a superposed state similar to the one in Eqn. (1.2):

$$\underbrace{\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) \otimes \cdots \otimes \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)}_{n \text{ qubits}} = \frac{1}{2^{n/2}} \sum_{i=0}^{2^n-1} |i\rangle \tag{1.6}$$

where we have taken the tensor product of the n qubits to get the complete state. By computing $f(i)$ only once, we can generate a state

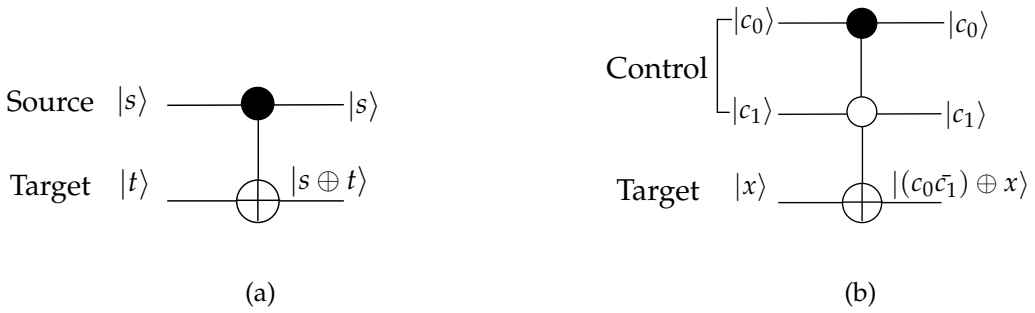
$$\frac{1}{2^{n/2}} \sum_{i=0}^{2^n-1} |i\rangle |0\rangle \xrightarrow{U_f} \frac{1}{2^{n/2}} \sum_{i=0}^{2^n-1} |i\rangle |f(i)\rangle \tag{1.7}$$

All the global properties of f are encoded in this state and can be extracted if an efficient method is devised. This is the kind of *massive parallelism* Shor uses in his famous factorization algorithm [4].

This same parallelism enables us to probabilistically combine packets in quantum switching networks. The input to the quantum switching network can be a superposition of multiple packet assignments, all of which are routed in parallel to the outputs. The switching network operates in different switch configurations for different packet assignments. We give a more detailed explanation of this concept when we describe random routing on quantum switching networks.

Among gates which operate on multiple qubit inputs, the most common type of gates used in quantum circuits are the controlled quantum gates. A controlled gate's function is determined by the state of some control qubits. In general, if a gate performs the unitary transformation $U : |x\rangle \rightarrow U|x\rangle$, then the controlled version of this gate, which becomes active when a control qubit, c, is in state $|1\rangle$, does the following transformation

$$|x\rangle \xrightarrow{U} U|x\rangle \implies |0\rangle |x\rangle \xrightarrow{Controlled-U} |0\rangle |x\rangle, |1\rangle |x\rangle \xrightarrow{Controlled-U} |1\rangle U|x\rangle \tag{1.8}$$



(a) (b)

Figure 1.2: Controlled-not gates: (a) Controlled not gate (b) Controlled-controlled not gate.

The simplest controlled gate is the controlled-not (CNOT) gate shown in Figure 1.2(a). The function of this gate is given by the mapping:

$$|s\rangle\,|t\rangle \xrightarrow{CNOT} |s\rangle\,|s \oplus t\rangle \tag{1.9}$$

Here $\oplus$ denotes the XOR operation or modulo 2 addition, i.e., if y is a binary variable then $y \oplus 0 = y$ and $y \oplus 1 = \bar{y}$, where $\bar{y}$ is the complement of y. Hence bit t is inverted when $s = 1$ and remains unchanged when $s = 0$. This gate functions as a NOT gate for the lower or target qubit when the control qubit is in state $|1\rangle$. The controlled-controlled-not (CC-NOT) gate shown in Figure 1.2(b), with two control qubits c_0 and c_1 performs the following transformation

$$|c_0, c_1, x\rangle \xrightarrow{\boldsymbol{CC-NOT}} |c_0, c_1, (c_0.\bar{c_1} \oplus x)\rangle \tag{1.10}$$

thus, it inverts x when $c_0 = 1$ and $c_1 = 0$. If a gate becomes active when a control qubit is 1, then that is indicated by a solid circle, (for c_0) and if a gate becomes active when a control qubit is 0 then that is indicated by an open circle, (for c_1). The CC-NOT gate transforms the basis vectors $|100\rangle$ and $|101\rangle$ to $|101\rangle$ and $|100\rangle$ respectively and leaves all the remaining six basis vectors unchanged. The qubit

affected by the operation of a controlled gate is called the target qubit. If we initialize x to 0, then this gate can be viewed as a *quantum comparator* which sets the output qubit to $|1\rangle$ when $c_0 < c_1$ and leaves it unchanged otherwise.

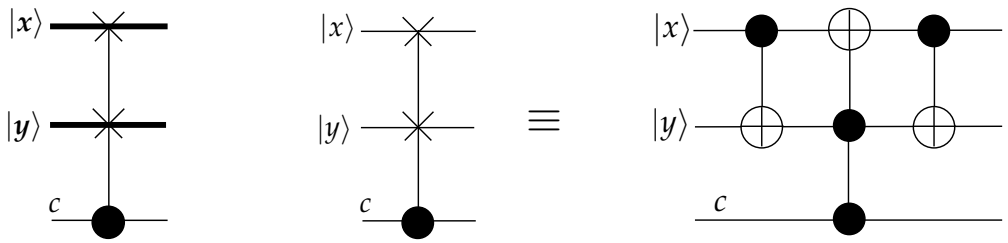
1.1.3 Quantum Copy and Switch Gates

Consider the CNOT gate shown in Figure 1.2(a). When $t = 0$ we see that the mapping is of the form $|s\rangle\,|0\rangle \xrightarrow{CNOT} |s\rangle\,|s\rangle$, thus a CNOT gate can also be viewed as a copier which copies the upper or source qubit on to the lower or target qubit when the target qubit is initialized to state $|0\rangle$. Note that this copy operation can only be done when the upper qubit is in one of the two basis states: $|0\rangle$ or $|1\rangle$. For a source qubit in the general state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ the mapping is given by:

$$\begin{aligned}(\alpha\,|0\rangle + \beta\,|1\rangle)_s\,|0\rangle_t \xrightarrow{CNOT} \alpha\,|0\rangle_s\,|0 \oplus 0\rangle_t + \beta\,|1\rangle_s\,|1 \oplus 0\rangle_t \\ = \alpha\,|0\rangle_s\,|0\rangle_t + \beta\,|1\rangle_s\,|1\rangle_t \\ = \alpha\,|00\rangle_{st} + \beta\,|11\rangle_{st} \qquad (1.11)\end{aligned}$$

Where we have used the linearity of quantum transformations. The output state is either $|00\rangle$ with probability $|\alpha|^2$ or $|11\rangle$ with probability $|\beta|^2$ and we have copied 0 and 1 bits to the target. Note that this is not equivalent to copying the complete source qubit to the target qubit because such an operation would require the following transformation:

$$\begin{aligned}(\alpha\,|0\rangle + \beta\,|1\rangle)_s\,|0\rangle_t \rightarrow (\alpha\,|0\rangle + \beta\,|1\rangle)_s \otimes (\alpha\,|0\rangle + \beta\,|1\rangle)_t \\ = \alpha^2\,|00\rangle + \alpha\beta\,|01\rangle + \alpha\beta\,|10\rangle + \beta^2\,|11\rangle\end{aligned}$$



(a) (b)

Figure 1.3: Switch gate: (a) Controlled-swap or switch gate (b) Switch gate made using CNOT gates.

It can be shown that this mapping is not unitary and hence cannot be done using quantum gates [26]. Thus a general quantum state can not be copied or cloned, this is known as the no-cloning theorem [15]. As a consequence of this theorem, wires in quantum circuits cannot be fanned out or split.

The basic gate used to build quantum switching networks is the controlled swap gate or switch gate, which is shown in Figure 1.3(a). A switch gate is a multi-qubit gate which swaps two sets of qubits or quantum packets when a control qubit c is $|1\rangle$ and passes them through unchanged otherwise [19, 20]. These two states of the switch gate are referred to as the cross and through states respectively. Thus, this gate can be used as a reconfigurable 2×2 switch to route quantum bit packets. The function of this gate can be represented as

$$|\boldsymbol{x}\rangle\,|\boldsymbol{y}\rangle\,|0\rangle_c \xrightarrow[Through]{Switch} |\boldsymbol{x}\rangle\,|\boldsymbol{y}\rangle\,|0\rangle_c \tag{1.12}$$

$$|\boldsymbol{x}\rangle\,|\boldsymbol{y}\rangle\,|1\rangle_c \xrightarrow[Cross]{Switch} |\boldsymbol{y}\rangle\,|\boldsymbol{x}\rangle\,|1\rangle_c \tag{1.13}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are equal length binary strings. The thick lines in Figure 1.3(a) for $\boldsymbol{x}$ and $\boldsymbol{y}$ indicate that there are multiple qubits on them. An implementation of the

switch gate with strings $\boldsymbol{x}$ and $\boldsymbol{y}$ of length 1, using two controlled-not (CNOT) and one CC-NOT gate is shown in Figure 1.3(b). If $\boldsymbol{x} = x_1x_0$ and $\boldsymbol{y} = y_1y_0$ are strings of length two, then the transformation done by the switch gate is given by:

$$|x_1x_0\rangle\,|y_1y_0\rangle\,|0\rangle_c \xrightarrow[Through]{Switch} |x_1x_0\rangle\,|y_1y_0\rangle\,|0\rangle_c \tag{1.14}$$

$$|x_1x_0\rangle\,|y_1y_0\rangle\,|1\rangle_c \xrightarrow[Cross]{Switch} |y_1y_0\rangle\,|x_1x_0\rangle\,|1\rangle_c \tag{1.15}$$

1.2 Switching Networks

Switching networks are arrangements of small reconfigurable switches which can be set dynamically to facilitate connections in parallel between a set of input entities and a set of output entities. A multistage switching network consists of multiple stages of smaller switches with each stage connected to the stages before and after it with some fixed pattern of wires. The switches comprising a stage are not ordinarily connected to each other. The most common types of networks are of size $2^p \times 2^p$ and are made using 2×2 switches. Multi-stage networks have many nice properties including constant path length and delay between any input and output and modular scalable structure.

A switching network is said to be *nonblocking* if any permutation assignment $\pi : i \rightarrow \pi(i)$ can be realized between the inputs and outputs without any conflicts. Thus, a nonblocking switching network can route all $n!$ permutation maps without any conflicts. Shannon [29] showed that any $n \times n$ nonblocking switch network built using 2×2 comparators has a minimum hardware cost of $\Omega(n \log n)$. Clos [30] gave the first fully nonblocking network which used unequal sized switches

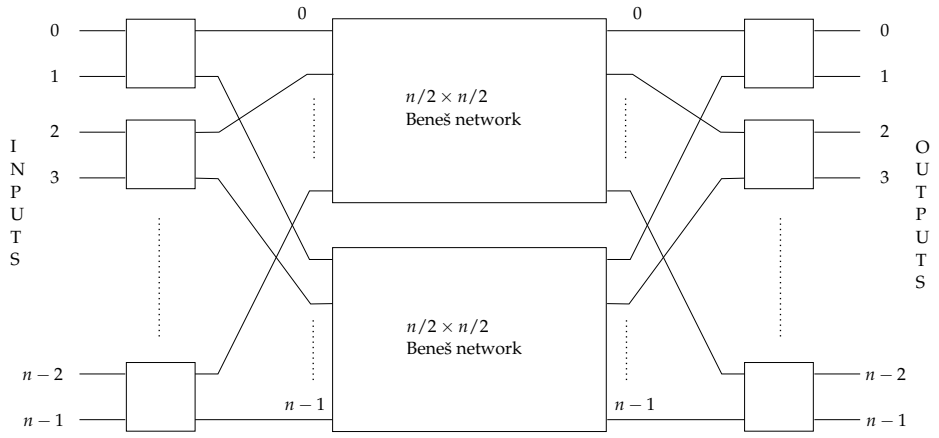


Figure 1.4: Beneš network.

in a three stage structure. He was able to show that the Clos network has a crosspoint count of $O(n^{3/2})$ which is lower than the $O(n^2)$ crosspoint count for a simple crossbar. Other nonblocking networks include the Beneš network [31], the Cantor switch [32] and generalized connectors [33]. The $n \times n$ Beneš is shown in Figure 1.4. As can be seen in the figure, this network is defined recursively, and thus has $2 \log n - 1$ stages of switches with each stage consisting of $n/2$ switches from top to bottom for a total of $(n \log n - n/2)$ 2×2 switches.

In addition to the hardware cost, a major consideration for switching networks is the path setup cost or the routing complexity. The routing complexity is determined by the time required to calculate the switch settings which realize any set of connection requests. Nonblocking networks provide multiple paths between input-output pairs so that if an internal link along one path between an input-output pair gets blocked by another connection then another clear path can be established. Usually, this multiplicity of internal paths and high hardware cost implies that most routing algorithms for such networks are centralized, i.e., given

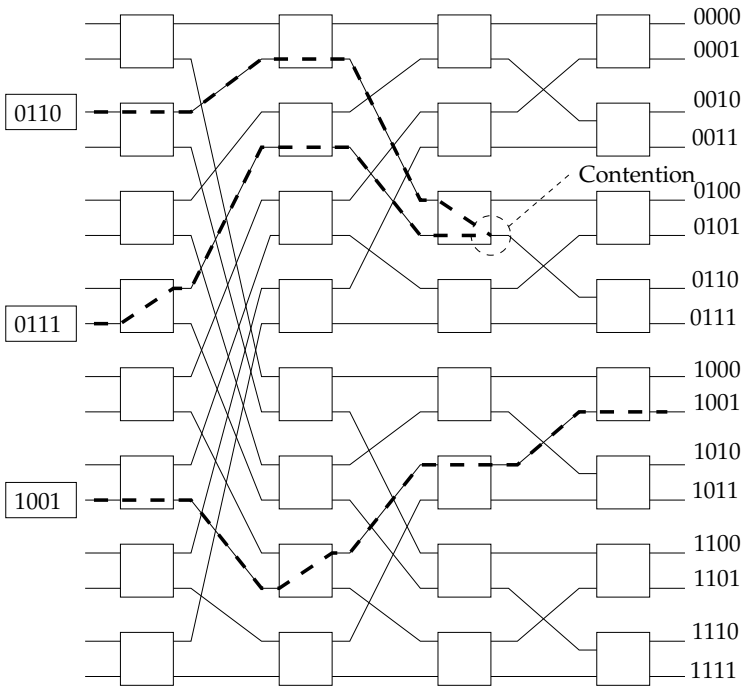


Figure 1.5: Self-routing and contention in Baseline network [20].

a connection assignment an external entity with knowledge of the entire switch state determines the switch settings. As a result many interconnection fabrics like the Delta [34] and Banyan networks [35] have been designed which allow simple decentralized routing.

In these networks the arrangement of the internal switches is such that the state of a switch can be determined by using only the locally available information about the destination addresses of the two input packets. The state of a switch is determined by a control bit or a routing bit. This routing bit is determined from the bits in the destination addresses of the input packets. For example, in the Baseline network shown in Figure 1.5 the control bit for a switch in the ith stage, c_i is the given by $c_i = a_{p-1-i}$, where $a_{p-1}a_{p-2}\cdots a_0$ is the binary output address of the input packet to this switch. If the control bit is 1 then the packet is routed to

the lower output and if the control bit is 0 then the packet is routed to the upper output on the switch. This can be seen in Figure 1.5 for the packet with the output address 1001. The control bit in the 0th stage is 1 so the packet gets routed to the upper output of the switch in this stage, the control bit for stage 1 is 0 and so the packet gets routed to the upper output of the switch and so on. This method of bit controlled routing is called self-routing. Self-routing networks like the Baseline network allow only a single unique path between any input-output pair and hence may block when there is output contention among packets incident on the same switch. Such a scenario is also shown in Figure 1.5. Hence, these are blocking networks. The permutation assignments for which blocking does not occur are called admissible permutations. Any admissible permutation can be self-routed over a network which allows bit controlled routing.

Chapter 2

Quantum Switching Networks and Random Routing

We define the concepts of quantum packets, quantum assignments, quantum non-blocking networks and random routing on quantum networks by using superpositions. We then characterize the output state for random routing on a blocking network, namely, the Baseline network. The output quantum state is also characterized for the non-blocking Beneš network when some stages in it are randomly routed.

2.1 Definitions

We define quantum packets, quantum assignments and the mapping performed by certain quantum switching networks in this section. These definitions are based on those given in [36, 37].

2.1.1 Quantum Packet

We assume that quantum packets composed of qubits are routed over a quantum network. A quantum packet consists of a set of *data qubits*, a set of *address qubits* and one additional qubit which we refer to as the *routing qubit*. Reversibility considerations in quantum systems mean that unlike classical systems no connecting wire or input/output line can remain empty. We use the routing qubit to overcome

this constraint, the routing qubit is set to $|1\rangle$ to indicate the presence of a quantum packet and to $|0\rangle$ to indicate an empty wire or absence of a packet. A quantum packet is defined below:

Definition 2.1. (Quantum packet) Let an input have m n_d-bit packets, $d_0, \ldots, d_{m-1}$. If packet d_i is to be transmitted with probability p_i to an output with n_a bit binary address a_i, then the source at this input feeds into the switching network a *quantum packet* of the following form:

$$\sum_{i=0}^{m-1} \alpha_i \, |r_i, a_i, d_i\rangle \tag{2.1}$$

where $|\alpha_i|^2 = p_i$ and $r_i = 1$. We refer to the individual components of the quantum packet, the bit strings $|r_i, a_i, d_i\rangle$ as *classical packets*. The routing bits in the classical packets are r_i. The length of the quantum packet is $n_d + n_a + 1$ qubits.

If the input source has no packets to transmit then the empty line is indicated by a single $n_d + n_a + 1$-bit string in which the routing bit is set to 0, the other $n_d + n_a$ bits can take arbitrary values. Note that we are only considering unicast assignments, i.e., assignments in which each classical packet is addressed to at most one output. This means that for an $n \times n$ network the length of the address field, n_a, is equal to $\log n$[1] bits.

2.1.2 Quantum Assignment

An input *assignment pattern* for an n-input quantum network is a sequence of classical packets, each of which belongs to a quantum packet on the n inputs from top

[1] All logarithms are in base 2 unless stated otherwise.

to bottom. A *quantum input assignment* is a superposition of a set of assignment patterns. An assignment pattern is called *non-contending* when no two classical packets in the pattern are addressed to the same output. A quantum assignment is *non-contending* when all of its assignment patterns are non-contending and *contending* otherwise. A non-contending assignment pattern is called a *permutation* assignment pattern when every input is matched to some output and a *sub-permutation* assignment pattern otherwise. These terms are formally defined as follows:

Definition 2.2. (Quantum assignment, assignment pattern) A *quantum assignment* for an $n \times n$ quantum network is a superposition of a set of T assignment patterns of the form:

$$\sum_{j=0}^{T-1} \gamma_j \left|(r_{j,0}, a_{j,0}, d_{j,0}), \cdots, (r_{j,n-1}, a_{j,n-1}, d_{j,n-1})\right\rangle \tag{2.2}$$

where the *assignment pattern* $\left|P_j\right\rangle = \left|(r_{j,0}, a_{j,0}, d_{j,0}), \cdots, (r_{j,n-1}, a_{j,n-1}, d_{j,n-1})\right\rangle$ consists of n classical packets in which $\left|(r_{j,i}, a_{j,i}, d_{j,i}\right\rangle$ is the packet at input i, $i = 0, \ldots, n-1$. $|\gamma_j|^2$ is the probability of the jth assignment pattern being realized and $\sum_{j=0}^{T-1} |\gamma_j|^2 = 1$. Assignment pattern $\left|P_j\right\rangle$ is *non-contending* if no two classical packets in it are addressed to the same output, i.e., $a_{j,i} \neq a_{j,k}$ for every $0 \leq i, k \leq n-1$ for which $r_{j,i} = r_{j,k} = 1$. Assignment pattern $\left|P_j\right\rangle$ is a *permutation* pattern or a permutation if it is non-contending and $r_{j,i} = 1$ for all $0 \leq i \leq n-1$ and a *sub-permutation* pattern or a sub-permutation if it is non-contending and $r_{j,i} = 0$ for some i, $0 \leq i \leq n-1$.

Note that the output addresses in a permutation pattern form a permutation on the numbers $0, \ldots, n-1$, i.e., there is a permutation map $\pi : i \rightarrow \pi(i)$ such

that $a_i = \pi(i), 0 \leq i \leq n-1$. Also a sub-permutation pattern is a non-contending pattern which is not a permutation.

A quantum assignment is a permutation (sub-permutation) assignment if all its patterns are permutations (sub-permutations).

If the quantum packets at all the inputs are of the kind given in Eqn. (2.1) then the quantum assignment can be expressed as a tensor product of the quantum packets as follows:

$$\bigotimes_{i=0}^{n-1} |Q_i\rangle = \bigotimes_{i=0}^{n-1} \left(\sum_{j=0}^{t_i-1} \alpha_{ij} \left| r_{ij}, a_{ij}, d_{ij} \right\rangle \right) \tag{2.3}$$

where $|Q_i\rangle = \sum_{j=0}^{t_i-1} \alpha_{ij} \left| r_{ij}, a_{ij}, d_{ij} \right\rangle$ is the quantum packet on input i. The tensor product in Eqn. (2.3) when expanded to a quantum assignment of the form in Eqn. (2.2) contains $\prod_{i=0}^{n-1} t_i$ assignment patterns.

2.1.3 Quantum Non-Blocking Networks

An $n \times n$ network is called a non-blocking network if it can realize all non- contending assignment patterns between its inputs and outputs. We extend this definition to quantum networks as follows:

Definition 2.3. (Quantum non-blocking network) An n-input, n-output quantum switch network is called an *n-quantum non-blocking network* or n-QNN, if for any non-contending assignment pattern $|P\rangle = |(r_0, a_0, d_0), \ldots, (r_{n-1}, a_{n-1}, d_{n-1})\rangle$ and a finite number of auxiliary qubits, each of which is initialized to state $|0\rangle$, it does

the following transformation:

$$|P\rangle\,|00\cdots0\rangle_{aux} \xrightarrow{n-\text{QNN}} \left|(r'_0,a'_0,d'_0),\ldots,(r'_{n-1},a'_{n-1},d'_{n-1})\right\rangle\,|\Psi_P\rangle_{aux} \tag{2.4}$$

where for all $j = 0,\ldots,n-1$, $d'_j = d_i$ and $r'_j = 1$ if there is a packet (r_i, a_i, d_i) in $|P\rangle$ such that $r_i = 1$ and the output address of the packet at input i is equal to j, i.e., $a_i = j$. If there is no packet addressed to output j then $r'_j = 0$. A finite number of auxiliary qubits are needed to ensure reversibility as many input patterns can get mapped to the same output pattern. These qubits are transformed to some state $|\Psi_P\rangle$.

If the quantum switching network is made using m smaller switches, then usually some fixed number of auxiliary qubits are needed per switch for a total of $O(m)$ auxiliary qubits. Due to linearity of quantum systems, an n-QNN can simultaneously realize all the patterns in a non-contending quantum assignment of the kind given in Eqn. (2.2). For a non-contending pattern, the n-QNN does not change the probability coefficients associated with an assignment pattern, it just changes the ordering of the valid classical packets within it.

A quantum network that doesn't perform the transformation given in Eqn. (2.4) for every non-contending input pattern is said to be blocking. A blocking network may still be able to realize such a transformation for a subset of non-contending input patterns. The non-contending assignment patterns for which the transformation in Eqn. (2.4) is realized are said to be *admissible*. All $n!$ permutation patterns are admissible for an n-QNN.

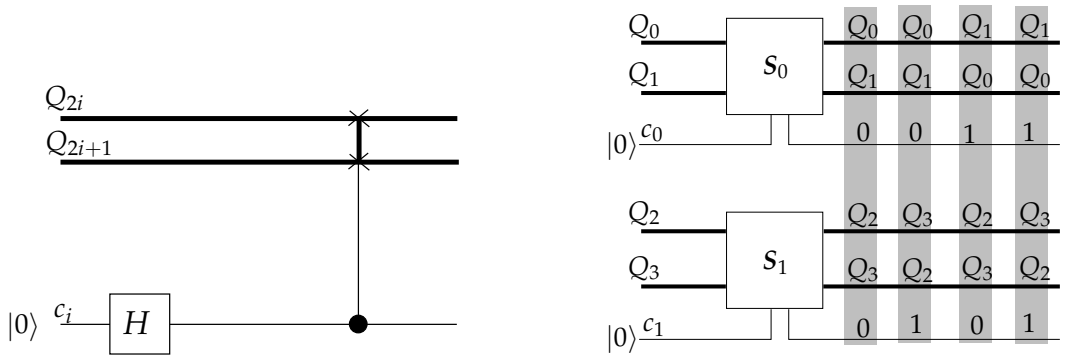
In networks consisting of reconfigurable switches, the state of each individ-

ual switch can viewed as being determined by setting a bit, called the control bit. When the value of the control bit is determined only by the local information available from the headers of the two input packets then the switches as well as the network is said to be self-routing. So, in a self-routing network the control bit or equivalently the switch state for any internal switch is a function of the destination addresses and routing bits of the input packets only at that switch.

If a 2×2 switch has the classical packets $|r_0, a_0, d_0\rangle$ and $|r_1, a_1, d_1\rangle$ incident on its upper and lower inputs respectively then the switch is self-routing if $c = f(r_0, r_1, a_0, a_1)$, where c is the control bit and f is a binary function of the bit strings r_0, r_1, a_0, and a_1. The alternative to self-routing is centralized routing in which switch settings are determined according to a centralized algorithm where, in general, the control bit of one switch may be determined by the address and routing bits at other switches. For quantum switching networks it is desirable to have self-routing switches as centralized control would have to be implemented classically resulting in greater overhead and slower speed.

2.1.4 Quantum Random Routing

We now describe the implementation of random routing of packets on quantum networks via the generation of quantum superpositions. Consider a simple 4×4 quantum network shown in Figure 2.1(b) which uses 2×2 switches as shown in Figure 2.1(a). The classical packet at input i is denoted by $Q_i = (r_i, a_i, d_i)$, $0 \leq i \leq 3$. The Hadamard gate in each switch is used to put the control qubit c in an



(a) Randomizing switch, S_i.

(b) Quantum output assignment.

Figure 2.1: Random routing on a 4×4 quantum network.

equal superposition in order to randomize the output. The operation of switch S_i, $i = 0, 1$, is defined as follows:

$$|Q_{2i}\rangle |Q_{2i+1}\rangle |0\rangle_{c_i} \xrightarrow[Through]{S_i} |Q_{2i}\rangle |Q_{2i+1}\rangle |0\rangle_{c_i} \tag{2.5}$$

$$|Q_{2i}\rangle |Q_{2i+1}\rangle |1\rangle_{c_i} \xrightarrow[Cross]{S_i} |Q_{2i+1}\rangle |Q_{2i}\rangle |1\rangle_{c_i} \tag{2.6}$$

where $|Q_{2i}\rangle$ and $|Q_{2i+1}\rangle$ are packets incident on the top and bottom input ports respectively of switch S_i. Therefore, S_i is set to through state when c_i is in state $|0\rangle$ and to cross state when c_i is in state $|1\rangle$. In S_i, the Hadamard gate, H, performs the following transformation on the control qubit, c_i which is initialized to state $|0\rangle$:

$$|0\rangle_{c_i} \xrightarrow{H} \frac{1}{\sqrt{2}} |0\rangle_{c_i} + \frac{1}{\sqrt{2}} |1\rangle_{c_i} \tag{2.7}$$

Using the linearity property of quantum systems, from Eqn. (2.5), Eqn. (2.6) and Eqn. (2.7) the mapping performed by switch S_0 is given by:

$$|Q_0, Q_1\rangle |0\rangle_{c_0} \xrightarrow[H]{S_0} \frac{1}{\sqrt{2}} \left(|Q_0, Q_1\rangle |0\rangle_{c_0} + |Q_1, Q_0\rangle |1\rangle_{c_0} \right) \tag{2.8}$$

and the mapping performed by switch S_1 is given by:

$$|Q_2, Q_3\rangle |0\rangle_{c_1} \xrightarrow[\boldsymbol{H}]{S_1} \frac{1}{\sqrt{2}} \left(|Q_2, Q_3\rangle |0\rangle_{c_1} + |Q_3, Q_2\rangle |1\rangle_{c_1} \right) \tag{2.9}$$

The tensor product of both the sides in Eqn. (2.8) and Eqn. (2.9) with the control qubits rearranged so that they are grouped together at the end in the order c_0, c_1 gives the following overall mapping for this network:

$$|Q_0, Q_1, Q_2, Q_3\rangle |00\rangle_c \xrightarrow[\boldsymbol{H}]{S_0 S_1} \frac{1}{2} |Q_0, Q_1, Q_2, Q_3\rangle |00\rangle_c + \frac{1}{2} |Q_0, Q_1, Q_3, Q_2\rangle |01\rangle_c + \frac{1}{2} |Q_1, Q_0, Q_2, Q_3\rangle |10\rangle_c + \frac{1}{2} |Q_1, Q_0, Q_3, Q_2\rangle |11\rangle_c \tag{2.10}$$

The four patterns and the associated control qubits at the output are shown in Figure 2.1(b) as the vertical grey columns. Recall that we defined each packet string Q_i to be of the form r_i, a_i, d_i where a_i are address qubits. Random routing in the switches does not make use of the address qubits a_i, $i = 0, \ldots, 3$, but we retain this representation to maintain continuity. The output quantum assignment contains four patterns each of which corresponds to a particular state of the network. We define a network state as:

Definition 2.4 (Network State)**.** The state of a switching network or *network state* is a configuration of the network in which the constituent switches are set in specific states (through or cross). For a quantum network in which the switches are configured using auxiliary control qubits, the state of the auxiliary qubits at the output corresponds to the network state. A network with m 2×2 switches can have 2^m network states.

For example, the network state in which both the upper and lower switches are set to cross state is indicated by the two control qubits for this pattern being in the combined state $|11\rangle$. If the input is (Q_0, Q_1, Q_2, Q_3) then the output for the $|11\rangle$ network state is (Q_1, Q_0, Q_3, Q_2). This network has a total of $2^2 = 4$ states and the four patterns in the quantum output assignment correspond to outputs obtained from the switch configurations for these four states. Every configuration of the switches in a network corresponds to the realization of some permutation map between the inputs and the outputs and as a consequence every network state corresponds to some input-output permutation mapping. The specific permutation map depends on the interconnection structure and control settings of the individual switches. For the above example, each network state corresponds to one particular value of the bit string formed by the two control qubits, and we can index network states by the number corresponding to this bit string, i.e., the four network states are 00, 01, 10 and 11. On measurement, one of the four patterns is observed with probability equal to $|1/2|^2 = 1/4$. Hence, the output quantum assignment in Eqn. (2.10) can be seen as a probabilistic multiplexing of all the possible ways an input assignment pattern can be routed on the network shown in Figure 2.1. The pattern probabilities can be changed by creating unequal superpositions of the control qubits c_0 and c_1 in the switches. This is explained in detail below.

Consider a randomizing quantum switch as shown in Figure 2.1(a) in which instead of the Hadamard gate, H, we use a generalized Hadamard gate with pa-

rameter α, denoted by H_α, whose operation is defined by the transformations:

$$|0\rangle \xrightarrow{H_\alpha} \sqrt{1-\alpha}\,|0\rangle + \sqrt{\alpha}\,|1\rangle \tag{2.11}$$

$$|1\rangle \xrightarrow{H_\alpha} \sqrt{\alpha}\,|0\rangle - \sqrt{1-\alpha}\,|1\rangle \tag{2.12}$$

Hence from Eqn. (2.11) we see that by acting on a control qubit for a 2×2 switch initialized to state $|0\rangle$, H_α can be used to generate a quantum output assignment in which, on measurement, the output pattern corresponding to the cross state and through state is observed with probability α and $1-\alpha$ respectively. We call such a switch a α–generalized randomizing switch. $\alpha = 1/2$ corresponds to a switch with a regular Hadamard gate, H, for randomization.

The transformation due to random routing on a general quantum switching network is given by:

Definition 2.5. (Random routing) Routing on an $n \times n$ quantum network done using generalized randomizing switches corresponds to the following mapping for an input assignment pattern $|P\rangle = |(r_0, a_0, d_0), \ldots, (r_{n-1}, a_{n-1}, d_{n-1})\rangle$

$$|P\rangle\,|00\cdots0\rangle_{aux} \xrightarrow[Routing]{Random} \sum_{j=0}^{T-1} \gamma_j \;\left|(r_{j,0}, a_{j,0}, d_{j,0}), \ldots, (r_{j,n-1}, a_{j,n-1}, d_{j,n-1})\right\rangle |\Psi_{P_j}\rangle_{aux} \tag{2.13}$$

where pattern $\left|P_j\right\rangle = |(r_{j,0}, a_{j,0}, d_{j,0}), \ldots, (r_{j,n-1}, a_{j,n-1}, d_{j,n-1})\rangle$ is the output corresponding to the jth network state. The classical packets $(r_{j,0}, a_{j,0}, d_{j,0}), \ldots, (r_{j,n-1}, a_{j,n-1}, d_{j,n-1})$ are a reordered version of the input assignment pattern $|P\rangle$. $|\gamma_j|^2$ is the probability of the jth network state being realized. A fixed number of auxiliary qubits initialized to state $|0\rangle$ are transformed to state $|\Psi_{P_j}\rangle_{aux}$ for the jth network

state. There are a total of T assignment patterns in the output quantum assignment. If the $n \times n$ network consists of m switches then we need m auxiliary qubits.

In random routing, the packets in a pattern are not routed according to the address qubits a_i in the packet header but by putting the internal switches in a superposition of through and cross states. If the $n \times n$ network is composed of m 2×2 switches then the maximum value of T above is 2^m. For a permutation input pattern, there can be a maximum of $n!$ distinct assignment patterns in the quantum output assignment.

If, in a network, there are multiple paths between input-output pairs then it is possible that given an input assignment pattern $|P\rangle$, for two distinct network states k and l, the corresponding output patterns $|P_k\rangle$ and $|P_l\rangle$ respectively, are the same. Although the output assignment patterns may be the same, the network states k and l are distinct, and hence, the associated auxiliary qubit states $\left|\Psi_{P_k}\right\rangle$ and $\left|\Psi_{P_l}\right\rangle$ are distinct and reversibility is maintained.

In the following sections we characterize the quantum output assignments for random routing on a blocking network: the $n \times n$ quantum Baseline network and also for a non-blocking network: the $n \times n$ quantum Beneš network.

2.2 Random Routing on Quantum Baseline Network

We characterize the output quantum assignment for random routing on a particular kind of self-routing multi-stage switching network called the Baseline network. We use the connection structure and self-routing property of this network to char-

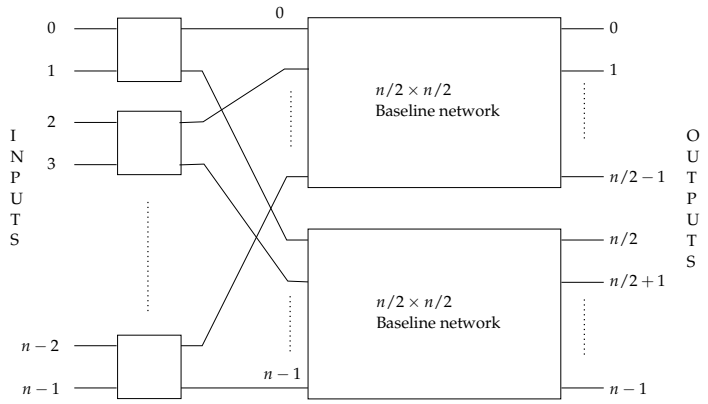


Figure 2.2: $n \times n$ Baseline network (recursive structure).

acterize the quantum output when quantum packets are randomly routed in the way described in Section 2.1.4. In particular, for an output assignment of the form given in Eqn. (2.13) we give a method to derive values of the probability amplitudes, γ_j, and characterize the set of permutation patterns superposed in the output. We first introduce some notation and describe some useful properties of the Baseline network.

2.2.1 The $n \times n$ Baseline Network

An $n \times n$ Baseline network, $n = 2^p$, is constructed recursively using 2×2 switches as shown in Figure 2.2. This network is composed of a stage of $n/2$ switches connected to two $n/2 \times n/2$ Baseline networks in an inverse shuffle pattern. The expanded network is shown for $n = 8$ in Figure 2.3. The input ports, also the output ports, of a network are numbered $0, \ldots, n-1$, top to bottom. The $n \times n$ network consists of $\log n = p$ stages of switches numbered $0, 1, \ldots, p-1$ from the input side to the output side. The switches in a stage are numbered $0, \ldots, n/2-1$ from

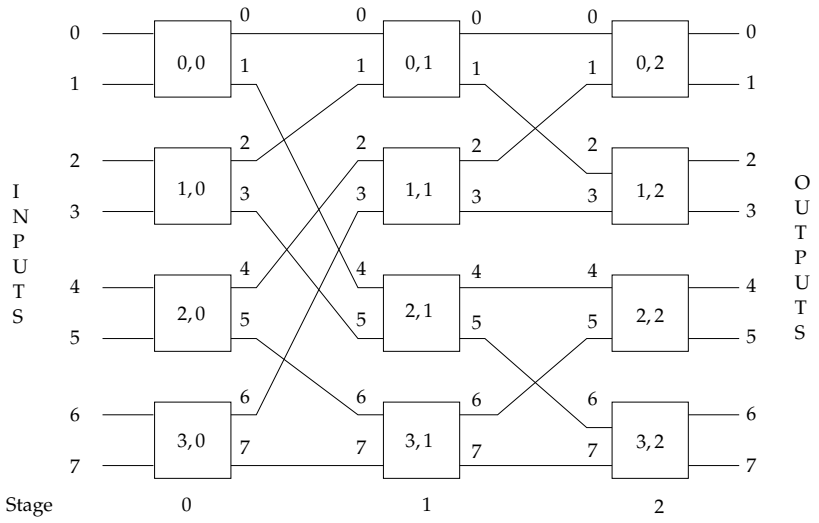


Figure 2.3: 8×8 Baseline network.

top to bottom. Switch (i, k) is the ith switch in kth stage as seen in Figure 2.3. The input ports and output ports to each stage are also numbered $0, \ldots, n-1$ from top to bottom. For each stage, every input (respectively, output) port has a unique and distinct index x, $0 \leq x \leq n-1$, which can be represented in binary form as $x_{p-1} \cdots x_0$ with x_{p-1} being the most significant bit (MSB). If output port b of stage m is connected to input port a of stage $m+1$ then the index a is obtained by doing a circular right shift of 1 bit on the lower $p-m$ bits of $b_{p-1} \cdots b_0$, i.e., $a_{p-1} \cdots a_0 = b_{p-1} \cdots b_{p-m} b_0 b_{p-m-1} \cdots b_1$. The Baseline network is blocking in nature as it cannot realize all permutation patterns, but it can be shown that all admissible permutation patterns can be self-routed on it. Furthermore, in every Baseline network, the path between any input and any output is unique, therefore, the number of admissible permutations is equal to the total number of network states which is equal to $2^{n/2 \log n} = n^{n/2}$. Packets can be *self-routed* in a Baseline network as follows: Suppose the output addresses of the packets on the

upper and lower inputs of a 2×2 switch in the mth stage are binary numbers $a_{p-1}a_{p-2}\cdots a_0$ and $b_{p-1}b_{p-2}\cdots b_0$ respectively. This switch is set in through state if $a_{p-m-1} = 0$ and $b_{p-m-1} = 1$ and in cross state if $a_{p-m-1} = 1$ and $b_{p-m-1} = 0$. When $a_{p-m-1} = b_{p-m-1}$ there is a contention and one of the packets has to be either dropped or buffered.

2.2.2 Random Routing: Problem Definition

Consider an $n \times n$ quantum Baseline network composed of generalized randomizing switches, where switch (i,k) is a α_{ik}–generalized randomizing switch. We give a method to calculate the probability of observing the output pattern corresponding to any network state j, i.e., $|\gamma_j|^2$ in Eqn. (2.13), for this network. We state the random routing problem formally as follows [38]:

Random Routing Problem: For an $n \times n$ Baseline network ($n = 2^p$), given that switch (i,k) is set in cross state with probability α_{ik}, $i = 0,\cdots,n/2-1$, $k = 0,\cdots,p-1$, find the probability routing matrix $\boldsymbol{P} = \{p_{ab}\}$ of routing input a to output b for all $a,b = 0,\cdots,n-1$.

This problem has a direct relation to finding the coefficients γ_j in Eqn. (2.13). This relation is given by the following theorem:

Theorem 2.1. *For an $n \times n$ quantum Baseline network, if network state j corresponds to the admissible input-output permutation map $\pi : i \rightarrow \pi(i)$, $0 \leq i \leq n-1$, then*

$$|\gamma_j|^2 = \prod_{i=0}^{n-1} p_{i,\pi(i)} \tag{2.14}$$

Proof. In a Baseline network there is a unique path that connects every input-output pair, thus every pattern in the output quantum assignment corresponds to a unique admissible input-output mapping. Therefore, the probability of a pattern being realized is equal to the product of individual path probabilities in the corresponding input-output mapping. The unique path nature also implies that there is one and only one network state for an output pattern and hence one network state for the associated permutation map. The result follows. □

Remark: We can calculate the right hand side of Eqn. (2.14) for any permutation π but the γ_j coefficients exist only when π corresponds to an admissible permutation map. Thus, to completely determine the output quantum state of a randomly routed Baseline network we need a method to find all the admissible permutations $\pi : i \rightarrow \pi(i)$. We do this later in Section 2.2.3.

2.2.2.1 Symmetric Case

We first attempt to solve the random routing problem for the symmetric case when $\alpha_{ik} = \alpha \ \forall \ i, \ k$. Consider an $n \times n$ (p-stage) Baseline network where $n = 2^p$ for some positive integer p.

We know that there is a unique path connecting any input-output pair in this network. Additionally, this path is realized by setting exactly p switches in either through or cross states. Thus, there is a unique setting of p switches corresponding to this path which realizes this connection. Assume that input a is routed randomly to output b. It is clear from the preceding argument that the probability of

this event, denoted by p_{ab}, is equal to the probability of setting the p intermediate switches in the appropriate combination of through and cross states to create the unique $a \rightarrow b$ path.

The Baseline network is self-routing and the switch settings are determined from the bits in the output address in the following manner. Let a packet have destination address b with binary representation $b_{p-1}b_{p-2}\cdots b_0$. In the path followed by this packet through the network, the switch in stage k routes this packet to its lower output if bit $b_{p-1-k} = 1$ and to its upper output if $b_{p-1-k} = 0$. For any switch in the Baseline network, the address of the output to which an input can be connected immediately via a switch can differ from the input address in only the least significant bit, e.g., in Figure 2.3 input 4 at stage 1 can connect only to outputs 4 and 5, input 3 at stage 0 can connect only to inputs 2 and 3 etc. Thus, a switch is set to a cross state in stage k if and only if b_{p-1-k} is not equal to the least significant bit of the input address at which this packet is present at stage k. It can be shown [39] that this address is equal to $b_{p-1}b_{p-2}\cdots b_{p-k}\,a_{p-1}\cdots a_{k+1}a_k$ for the packet routed along the $a \rightarrow b$ path. Thus, for the path from input a to output b, the switch at stage k is set in a cross state if and only if $b_{p-1-k} \neq a_k$, which is true if and only if $b_{p-1-k} \oplus a_k = 1$. This implies the following:

Proposition 2.1. *In an $2^p \times 2^p$ Baseline network, for the path from input a to output b, the switch in stage k is in a cross state if and only if $a_k \oplus b_{p-k-1} = 1$, where $\oplus$ denotes modulo 2 addition and $k = 0, \ldots, p-1$.*

Hence, the number of switches, d, set in cross state along the $a \rightarrow b$ path is

$$d = \sum_{k=0}^{p-1} a_k \oplus b_{p-k-1} \tag{2.15}$$

Notice that d is equal to the number of positions in which the corresponding bits differ in the binary representations of b and $\rho(a)$, where ρ is the bit reversal permutation, i.e., $\rho(a_{p-1}\, a_{p-2} \cdots a_1\, a_0) = a_0\, a_1 \cdots a_{p-2}\, a_{p-1}$. This is nothing but the Hamming distance between b and $\rho(a)$. Denoting the Hamming distance between a and b by $D(a\|b)$ we get

$$p_{ab} = \alpha^d (1-\alpha)^{p-d} = \alpha^{D(\rho(a)\|b)} (1-\alpha)^{p-D(\rho(a)\|b)} \tag{2.16}$$

Where α is the probability of setting a switch in cross state. Thus, the matrix $\boldsymbol{P} = \{p_{ab}\}$, where $p_{ab} = \alpha^{D(\rho(a)\|b)} (1-\alpha)^{n-D(\rho(a)\|b)}$, $0 \leq a, b \leq n-1$.

It is easy to show that the matrix thus formed is a probability or stochastic matrix, i.e., the rows sum up to 1. In fact, we can show that $\boldsymbol{P}$ is symmetric and hence it is doubly stochastic, i.e., both the rows and the columns sum up to 1.

Theorem 2.2. *$\boldsymbol{P}$ is symmetric and doubly-stochastic.*

Proof. For any p-bit number, x, there are exactly $\binom{p}{k}$ p-bit binary strings that differ from the p-bit binary representation of x in k places and hence their Hamming distance from x is k. Therefore, the sum of the probabilities in the row corresponding to input with address x in $\boldsymbol{P}$ is

$$\sum_{k=0}^{p-1} p_{xk} = \sum_{k=0}^{p} \binom{p}{k} \alpha^k (1-\alpha)^{p-k} = (1-\alpha+\alpha)^p = 1 \tag{2.17}$$

The same argument holds for the sum of entries for the column corresponding to the output with address x. Hence, $\boldsymbol{P}$ is doubly stochastic.

Also, $p_{ab} = \alpha^{D(\rho(a)\|b)}(1-\alpha)^{p-D(\rho(a)\|b)}$ and since $D(\rho(a)\|b) = D(\rho(b)\|a)$ this implies $p_{ab} = p_{ba}$, i.e., $\boldsymbol{P}$ is symmetric. □

By adjusting the parameter α we can get different distributions for $\boldsymbol{P}$. For example, when $\alpha = 1$, i.e., all switches are always put in cross state, we see that $p_{ab} = 1$ if $D(\rho(a)\|b) = p$ and 0 otherwise. Thus, inputs are connected to the outputs with addresses which have the maximum Hamming distance from their own bit reversed addresses. For $\alpha = 0$, i.e., all switches are always put in through state, we see that inputs are connected to the outputs which have 0 Hamming distance from their bit reversed addresses, i.e., $\boldsymbol{P}$ corresponds to the bit reversal permutation.

2.2.2.2 General Case

We now generalize this routing scheme by setting each 2×2 switch in cross state with an arbitrary probability. Specifically, the ith switch in the kth stage is set in cross state with probability α_{ik}, $i = 0, \cdots, n/2-1$ and $k = 0, \cdots, p-1$. We define a bit sequence $m_{p-1}, \cdots, m_0$ called the routing mask, where

$$m_l = a_{p-1-l} \oplus b_l \quad \forall\, l = 0, 1, \cdots, p-1 \tag{2.18}$$

Thus, from Proposition 2.1, $m_{p-l-1} = 1$ if and only if the path from input a to output b passes through a switch in cross state in stage l. Therefore, in this case,

$$p_{ab} = \prod_{r=0}^{p-1} \alpha_{i_r r}^{m_{p-r}} \, (1-\alpha_{i_r r})^{1-m_{p-r}}, \quad 0 \le a, b \le n-1 \tag{2.19}$$

where the path from input a to output b passes through switch i_r in stage r. At any stage r, input port x is incident on switch $(\lfloor x/2 \rfloor, r)$, thus i_r can be obtained from

the following equation:

$$
\begin{aligned}
i_r &= \begin{cases} \lfloor (a_{p-1} \cdots a_1 a_0)/2 \rfloor & r = 0, \\ \lfloor (b_{p-1} \cdots b_{p-r}\, a_{p-1} \cdots a_{r+1} a_r)/2 \rfloor & r = 1, \ldots, p-1. \end{cases} \\
\Rightarrow i_r &= \begin{cases} a_{p-1} \cdots a_1 & r = 0, \\ b_{p-1} \cdots b_{p-r}\, a_{p-1} \cdots a_{r+1} & r = 1, \ldots, p-1. \end{cases}
\end{aligned} \tag{2.20}
$$

The resulting routing matrix $\boldsymbol{P}$ obtained using equations Eqn. (2.18), Eqn. (2.19) and Eqn. (2.20) is still doubly stochastic as all inputs have packets and the network simply permutes them to the outputs, i.e., all outputs also have a packet with probability one. However, $\boldsymbol{P}$ is, in general, no longer symmetric.

As we proved earlier in Theorem 2.1, the coefficient, γ_j, of the output assignment pattern for network state j is given by $|\gamma_j|^2 = \prod_{i=0}^{n-1} p_{i,\pi(i)}$ when network state j corresponds to the admissible input-output mapping $\pi : i \rightarrow \pi(i)$. We now give a characterization of all the admissible input-output permutation maps for the Baseline network.

2.2.3 Admissible Permutation Patterns

The set of admissible permutation patterns for a quantum Baseline network were characterized in [20] based on the concept of frames given by Çam [40] and balanced matrices. In a nutshell, it was shown that, for any admissible permutation pattern, certain sub-matrices of the $n \times p$ matrix in which row i is the binary address of the output to which input i is routed are balanced. A $2^k \times k$ binary matrix is called balanced when the 2^k numbers whose binary representation is given by

the k-bit binary strings in the rows are all distinct, i.e., they form a permutation of the numbers $0, 1, \ldots, 2^k - 1$.

We now give a more detailed description of this concept. For an $n \times n$ Baseline network and a permutation input assignment pattern let b_i be the output address to which packet at input i is routed and let the binary representation of b_i be $b_{i,p-1}, \cdots, b_{i,0}$. Let $\boldsymbol{S}$ be an $n \times p$ binary matrix such that

$$\boldsymbol{S} = \{s_{ij}\}, s_{ij} = b_{i,p-1-j} \quad , 0 \leq i \leq n-1, 0 \leq j \leq p-1 \tag{2.21}$$

Define the following sub-matrices of $\boldsymbol{S}$ of size $2^{k+1} \times (k+1)$

$$\boldsymbol{R}_{ik}(l, m) = \{s_{xm} : x = i \cdot 2^{k+1} + l\} \tag{2.22}$$

where $\boldsymbol{R}_{ik}(l, m)$ refers to the entry in row l and column m in $\boldsymbol{R}_{ik}$, $0 \leq l \leq 2^{k+1} - 1$, $0 \leq m \leq k$, $0 \leq i \leq 2^{p-k-1} - 1$, $0 \leq k \leq p-1$. Let $r_{ik}(l)$ be the number whose binary representation is the $k+1$-bit binary string in row l of $\boldsymbol{R}_{ik}$. Then the following result holds [20]:

Theorem 2.3. *A permutation input assignment pattern is admissible on an $n \times n$, ($n = 2^p$) Baseline network if and only if the 2^{k+1} numbers $r_{ik}(l)$, $l = 0, \ldots, 2^{k+1} - 1$, are all distinct and this is true for all i, $0 \leq i \leq 2^{p-k-1} - 1$ and k, $0 \leq k \leq p-1$.*

The proof for this theorem was derived for characterizing the output state of a quantum Baseline network with self-routing of packets to their desired destinations, where superpositions are made when there is blocking at an internal switch. We relate our formulation of Theorem 2.3 to the one given in [20]. A frame is an $n \times p$ grid of the kind shown in Figure 2.4. In a frame, each rectangle of size

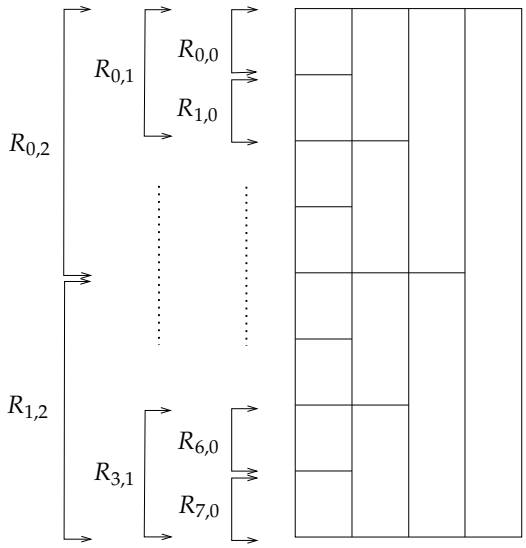


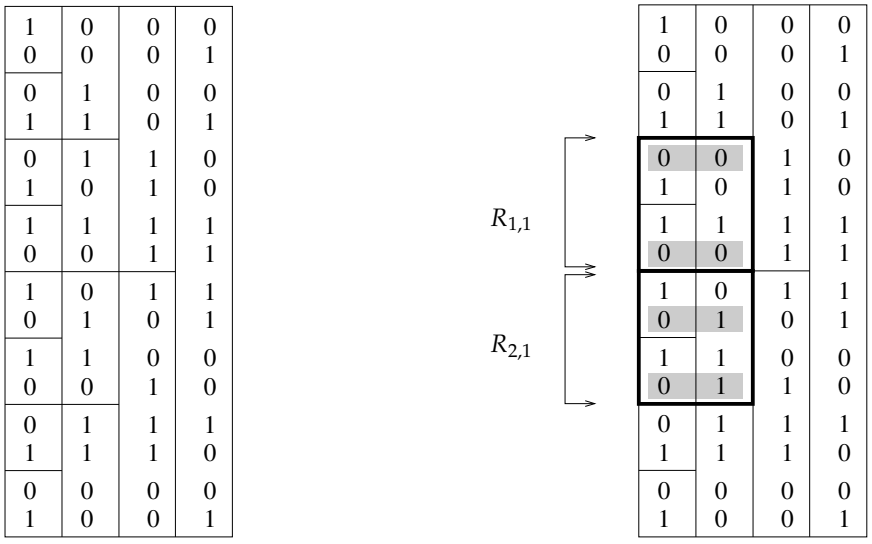
Figure 2.4: The 16×4 frame.

$2^m \times m$ corresponds to a unique subset R_{ik}. For instance, all the 2×1 rectangles in column 0 of the frame in Figure 2.4 correspond to the sets $R_{0,0}, R_{1,0}, \ldots, R_{7,0}$, the 4×2 rectangles to the sets $R_{0,1}, R_{1,1}, R_{2,1}, R_{3,1}$ etc. Figure 2.5(a) shows a binary permutation matrix in a frame. We see in this figure that all the rows in any rectangle are distinct, hence the permutation assignment corresponding to this binary matrix is admissible on the 16×16 Baseline network. Figure 2.5(b) shows a different binary permutation over the same frame. In this case we see that rows 0 and 3 in $R_{1,1}$ and rows 2 and 3 in $R_{2,1}$ are the same. Hence, the permutation assignment corresponding to this binary matrix is not admissible on the 16×16 Baseline network.

Thus, from the two preceding sections the quantum assignment at the output of the quantum Baseline network is of the form:

$$\sum_{j=0}^{T-1} \gamma_j \left| (r_{\pi_j^{-1}(0)}, a_{\pi_j^{-1}(0)}, d_{\pi_j^{-1}(0)}), \cdots, (r_{\pi_j^{-1}(n-1)}, a_{\pi_j^{-1}(n-1)}, d_{\pi_j^{-1}(n-1)}) \right\rangle \qquad (2.23)$$

where the jth network state corresponds to the permutation map π_j in which input



(a) An admissible permutation.

(b) A non-admissible permutation.

Figure 2.5: Frames and permutation matrices.

i is connected to output $\pi_j(i)$, $0 \leq i \leq n-1$. π_j^{-1} is the inverse of such a permutation map. All such permutations, π_j, are admissible and they satisfy the conditions of Theorem 2.3 and the coefficients γ_j satisfy equation Eqn. (2.14).

2.3 Quantum Beneš Network

An $n \times n$ Beneš network ($n = 2^p$), denoted by $\mathcal{B}_p$, is defined recursively as shown in Figure 2.6. It consists of two $n/2 \times n/2$ Beneš networks stacked one on top of the other and surrounded by columns of $n/2$ 2×2 switches on both sides [31]. The column on the input side is connected to next stage by an inverse shuffle connection while the output side column is reached by a shuffle connection. The expanded form of $\mathcal{B}_3$ is shown in Figure 2.7. $\mathcal{B}_p$ consists of $2p-1$ stages of 2×2

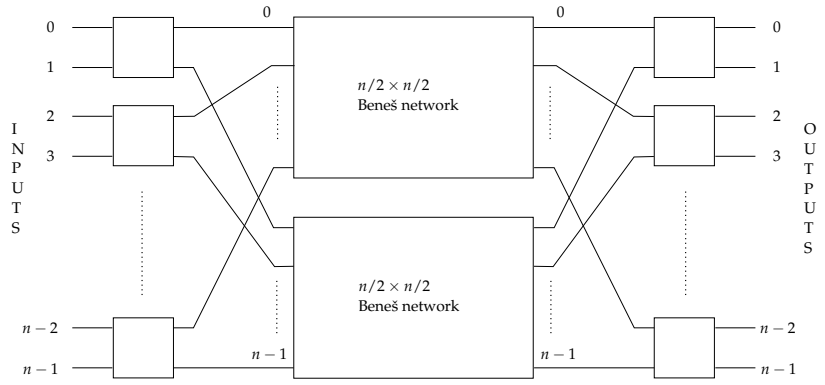


Figure 2.6: $n \times n$ Beneš network (recursive structure).

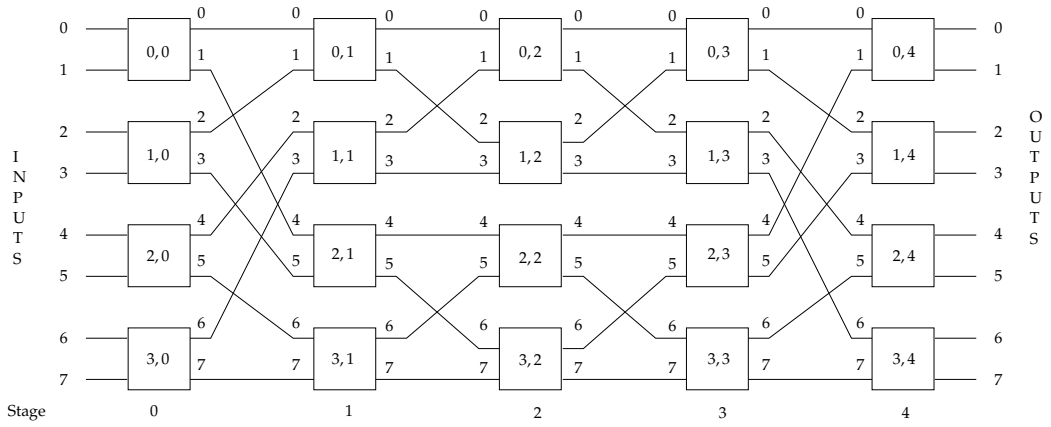


Figure 2.7: 8×8 Beneš network.

switches where stages 0 to $p-1$ form a $n \times n$ Baseline network and stages $p-1$ to $2p-2$ form an $n \times n$ inverse Baseline network. Similar to the Baseline network the stages are numbered $0, \ldots, 2p-2$ from left to right. The Beneš network is non-blocking but the routing algorithms to determine switch settings for arbitrary permutations are centralized. The well-known looping algorithm by Waksman can route any permutation assignment in $O(n \log n)$ time [41]. Parallel algorithms to determine switch settings require $O(\log^2 n)$ or $O(\log^4 n)$ time using n fully connected or shuffle-exchange interconnected processors respectively [42, 43, 44]. Self-

routing algorithms—which route permutations by setting switches on-the-fly using only local information (source and destination addresses of the incident packets) at each switch in the Beneš network, were given by Nassimi and Sahni [45] and by Raghavendra and Boppana [46]. In case of contention for an output link at a switch Nassimi and Sahni's algorithm gives priority to the packet at the upper input port of the switch whereas Raghavendra and Boppana's algorithm gives priority, based on some type of comparison operation, to the smaller of the two packets. These algorithms can pass only particular classes of permutations, namely, the bit-permute-complement class and the linear complement class respectively.

For a quantum Beneš network we too consider a self-routing scheme, but it differs from the classical algorithms in two significant ways. First, we use two different kinds of switches to compose the Beneš network. All the switches in the first $\log n - 1$ stages are quantum randomizing switches of the kind shown in Figure 2.1(a) while all the switches in the last $\log n$ stages are quantum self-routing switches of the kind first introduced in [20]. Second, contentions are resolved in the self-routing switches by using quantum superpositions. The quantum self-routing switch performs the same transformation as a classical bit-controlled self-routing switch when there is no contention for one of its output links. However, in the case that a contention occurs, the quantum self-routing switch creates an output consisting of a superposition of the two patterns corresponding to the through and cross states. In each of the two patterns, the routing qubit of the classical packet at the output for which there was contention is marked as zero.

Lee [47] proved that for any $(2 \log n - 1)$ stage network formed by the reduced

concatenation of two $\log n$-stage $n \times n$ Banyan networks (the redundant last stage of the first network is removed), any input permutation assignment can be routed over the first $\log n - 1$ stages (Network 1) such that the resulting permutation is self-routable over the last $\log n$ stages (Network 2). The proof was constructive in nature and gave a routing algorithm for Network 1. Since a Beneš network is a reduced concatenation of Baseline and inverse Baseline networks (which are both Banyan networks), this algorithm can be adapted for use over the Beneš network as well. Lee gave such an algorithm in [48].

We use Lee's result to prove that for an $n \times n$ quantum Beneš network, $n = 2^p$, any non-contending input assignment pattern is probabilistically self-routable. In particular, we consider non-contending input assignment patterns with random routing on the first $\log n - 1$ stages and self-routing on the last $\log n$ stages of a quantum Beneš network. For any such input assignment pattern, we prove that in the quantum output assignment, there exist assignment patterns with non-zero probability amplitude, in which all the valid input packets are present at their correct output addresses. Additionally, we give a lower bound on the probability of observing such patterns on measurement at the output.

We also identify a class of permutation patterns, for which this bound is equal to 1, i.e., for all the permutation patterns in this class, the following is true: In every pattern in the quantum output assignment, all the valid input packets are present at their correct output addresses.

2.3.1 Randomizing and Self-Routing Quantum Switches

Consider an assignment pattern in which the packets $|r_u, a_u, d_u\rangle$ and $|r_l, a_l, d_l\rangle$ are incident on the upper and lower input lines of a 2×2 quantum switch in the Beneš network. If this switch is in stage j, $0 \leq j \leq p-2$, (the first $\log n - 1$ stages) then it is a randomizing switch, denoted by R, and does the following transformation:

$$|(r_u, a_u, d_u), (r_l, a_l, d_l)\rangle \, |0\rangle_c \xrightarrow{R} \frac{1}{\sqrt{2}} \, |(r_u, a_u, d_u)(r_l, a_l, d_l)\rangle \, |0\rangle_c + \frac{1}{\sqrt{2}} \, |(r_l, a_l, d_l), (r_u, a_u, d_u)\rangle \, |1\rangle_c \qquad (2.24)$$

Note that the address qubits are not used to determine the output.

If the switch is in stage j, $p-1 \leq j \leq 2p-2$, then the switch, denoted by S, is self-routing. The routing decision at S is made based on bits $a_{u,2p-2-j}$ and $a_{l,2p-2-j}$ where $a_{u,p-1} \cdots a_{u,0}$ and $a_{l,p-1} \cdots a_{l,0}$ are the binary representations of a_u and a_l respectively.

The routing on S is done as follows: the input packet in which bit $2p-2-j$ is 0 is routed to the upper output port and the input packet in which bit $2p-2-j$ is 1 is routed to the upper output port. Thus, S is set in through state if $a_{u,2p-2-j} = 0, a_{l,2p-2-j} = 1$ and in cross state if $a_{u,2p-2-j} = 1, a_{l,2p-2-j} = 0$. If both these bits are 0 then there is contention for the upper output port, and if both these bits are 1 then there is contention for the lower output port. We resolve the contention by using quantum superposition. The transformation for S when both the input packets are valid, i.e., $r_u = r_l = 1$, is given by:

$$|(1,a_u,d_u),(1,a_l,d_l)\rangle\,|0\rangle_c\,|00\rangle_{aux}$$

$$\xrightarrow[Through]{S} |(1,a_u,d_u),(1,a_l,d_l)\rangle\,|0\rangle_c\,|00\rangle_{aux}\,,\ a_{u,2p-2-j}=0, a_{l,2p-2-j}=1 \tag{2.25}$$

$$\xrightarrow[Cross]{S} |(1,a_l,d_l),(1,a_u,d_u)\rangle\,|1\rangle_c\,|00\rangle_{aux}\,,\ a_{u,2p-2-j}=1, a_{l,2p-2-j}=0 \tag{2.26}$$

$$\xrightarrow[Superpose]{S} \frac{1}{\sqrt{2}}(|(1,a_u,d_u),(0,a_l,d_l)\rangle\,|0\rangle_c + |(1,a_l,d_l),(0,a_u,d_u)\rangle\,|1\rangle_c)\otimes|01\rangle_{aux}\,,$$

$$a_{u,2p-2-j}=0, a_{l,2p-2-j}=0 \tag{2.27}$$

$$\xrightarrow[Superpose]{S} \frac{1}{\sqrt{2}}(|(0,a_u,d_u),(1,a_l,d_l)\rangle\,|0\rangle_c + |(0,a_l,d_l),(1,a_u,d_u)\rangle\,|1\rangle_c)\otimes|10\rangle_{aux}\,,$$

$$a_{u,2p-2-j}=1, a_{l,2p-2-j}=1 \tag{2.28}$$

An explicit quantum circuit for S was given in [20]. The two auxiliary qubits are required to set the routing bit on the packet at the non-contending output to zero in case of output contention. Thus, we see from Eqns. (2.27) and (2.28) that, in the case of contention for an output, a quantum assignment with two superposed patterns is created at the outputs of the switch. One pattern corresponds to the output map for the through state and the other for the cross state. In both the patterns, the routing qubit for the packet at the non-contending output is set to zero.

2.3.2 Output State

The first p stages form a Baseline network which is a unique path network. Thus, any configuration of switch settings, i.e., network state, corresponds to a unique permutation of the packets in the input pattern. Classically, each switch can be set in 2 states: either through or cross. In $p-1$ stages, there are $n(p-1)/2$ switches and hence $2^{n(p-1)/2}$ possible network states with each network state corresponding to a unique permutation map. This implies that, for an input assignment pattern $|P\rangle = |(r_0, a_0, d_0), \cdots, (r_{n-1}, a_{n-1}, d_{n-1})\rangle$ and control qubits in state $|0\rangle$, the quantum assignment at the output of the first $p-1$ stages is of the following form:

$$\begin{aligned}
&\sum_{j=0}^{T-1} \gamma_j |P_j\rangle |\Psi_{P_j}\rangle_c \\
&= \sum_{j=0}^{T-1} 2^{-n(p-1)/4} |(r_{\pi_j^{-1}(0)}, a_{\pi_j^{-1}(0)}, d_{\pi_j^{-1}(0)}), \cdots, (r_{\pi_j^{-1}(n-1)}, a_{\pi_j^{-1}(n-1)}, d_{\pi_j^{-1}(n-1)})\rangle |\Psi_{P_j}\rangle_c
\end{aligned} \tag{2.29}$$

where $\pi_j : i \rightarrow \pi_j(i)$ is the permutation map corresponding to network state j and π_j^{-1} is the inverse permutation, i.e., $\pi_j^{-1}(i)$ is the input connected to output i in π_j. $T = 2^{n(p-1)/2}$, is the total number of network states. Ψ_{P_j} is an $n(p-1)/2$-bit binary string representing the control qubits of all the switches for network state j. Every network state corresponds to a unique permutation map, hence no two π_js are the same.

All the patterns $\left|P_j\right\rangle$ are self-routed using bit control over stages $p-1, \ldots, 2p-2$. Bit a_{2p-2-j} is used as the control bit at stage j and the switch operation under bit-controlled routing is given by Eqn. (2.25)–(2.28). Thus, if pattern $\left|P_j\right\rangle$ passes

through the last p stages without contention then at the output the pattern resulting from $\left|P_j\right\rangle$ has all the valid packets in $\left|P_j\right\rangle$ correctly routed to their destination output lines. If packets in a pattern encounter contention while being routed then some of them have their routing bits marked as zero and hence the number of valid packets at the output is lesser than the number of valid packets in $\left|P_j\right\rangle$. We now give a brief discussion of Lee's [47] routing algorithm for Beneš networks which will be useful in showing some properties of the output assignment of the quantum Beneš network.

2.3.3 Lee's Routing Algorithm

Consider the $n \times n$, Beneš network, $\mathcal{B}_p$, $n = 2^p$ as a composition of two networks, $SN1$ and $SN2$, where $SN1$ is the network formed by the first $p-1$ stages and $SN2$ is the network formed by the last p stages. Lee [47] proved that any input permutation can be routed over $SN1$ stage-by-stage using a recursive partitioning method for the packets such that the resulting permutation at the output of $SN1$ is admissible for $SN2$. Since all admissible permutations for the inverse Baseline network are self-routable using bit control, this means that the permutation can be self-routed without conflicts over $SN2$. The proof was constructive in nature and hence a control algorithm for routing over $SN1$ was also given [48]. We give a short description of this algorithm which is required to prove our subsequent results.

2.3.3.1 Routing Control for $SN1$

The routing control algorithm for $SN1$ to transform any arbitrary input permutation into an $SN2$ passable permutation is as follows:

1. Start at stage $k = 0$.

2. For stage k divide the 2^{p-1} switches into 2^k contiguous blocks from top to bottom, with each block containing 2^{p-1-k} switches.

3. Consider one such block at stage k. At every input to this block calculate the following quotient:

$$\lfloor b/2^{k+1} \rfloor = b_{p-1} \cdots b_{k+1} \tag{2.30}$$

   where $b = b_{p-1}b_{p-2} \cdots b_0$ is the binary string representing the output address of the packet at an input. Thus, a quotient at stage k is equal to integer division of the address by 2^{k+1}.

4. It can be shown that in each block there are exactly two inputs for which these quotients are equal. Connect any one of these inputs to the upper output of its 2×2 switch and connect the other input to the lower output of its own 2×2 switch.

5. Repeat this process for all such pairs of inputs in that block.

6. Repeat steps 3–5 for all the remaining blocks in stage k.

7. Increment k by one and if $k < p - 2$ go to step 2 else stop.

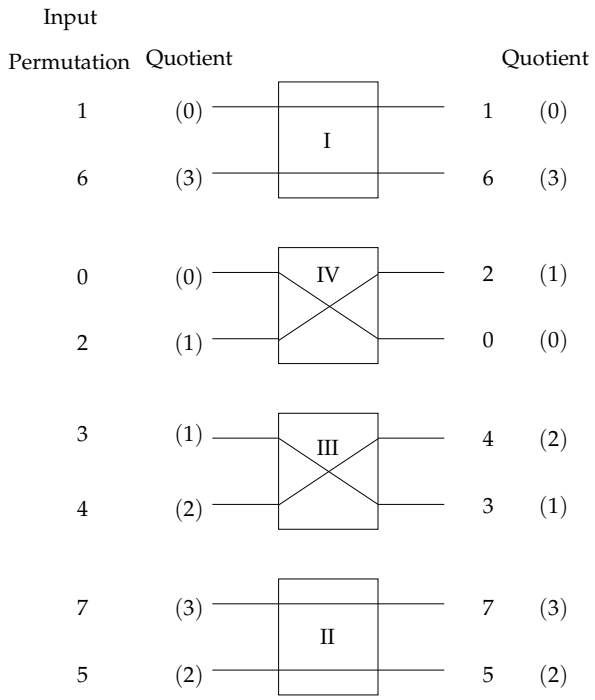


Figure 2.8: A quotient partition realized on a set of switches.

Using this algorithm the 2^{p-1-k} inputs in a block are partitioned into two halves, one assigned to the upper outputs, and one to the lower outputs of the internal switches based on the quotient calculations of (2.30). Since one partition is done per block, we have one partition in stage 0, two in stage 1 and so on till 2^{p-2} partitions in stage $p-2$. This algorithm requires intra-stage communication between switches and hence is not a self-routing algorithm. The time complexity of the algorithm is $O(n \log n)$. Pipelining can reduce this cost to $O(n)$. We explain the process of performing a partition on a single block of four switches with an example in Figure 2.8. The input permutation is the first column of numbers on the left, i.e., $\{1, 6, 0, 2, 3, 4, 7, 5\}$. The quotients on integer division by 2, i.e., the integral portion of division by 2, are written in parentheses. Switch I is set straight arbitrarily.

By doing this we missed quotient 3 for the upper output port. So we find another quotient 3 which is the number 7 in switch II and set switch II straight. By doing this we missed quotient 2 for the upper output port. So we connect switch III to cross, and so on. A possible order in which switches are set is I, II, III, IV. Note that quotients on the upper outputs (respectively, lower outputs) of the switches form all distinct numbers from 0 to 3.

In this example we note that the partitioning process formed a single cycle which included all the four switches. The setting of the first switch, in this case, switch I, is arbitrary and hence, there are a total of two different settings for the switches in this cycle which enable partitioning of the incoming connections. It is possible that the number of cycles, k, is greater than one. In that case, the total number of switch configurations to achieve the partitioning based on quotient calculations is equal to 2^k.

2.3.3.2 Routing Control for $SN2$

The quotient partitioning based routing algorithm for $SN1$ outputs a permutation which can be routed without conflicts over $SN2$. The routing on $SN2$ is bit controlled. The control bit C_i for the ith stage of $SN2$, $0 \leq i \leq p-1$, (i.e., $(p-1+i)$th stage of $\mathcal{B}_p$) to realize any admissible permutation for $SN2$ without blocking is given by: $C_i = b_{p-1-i}$. If the control bit of the packet at the upper input is "0" and the control bit of the packet at the lower input is "1", the switch is set in a through state. If it is "1" for the upper input and "0" for the lower input, the switch is set to

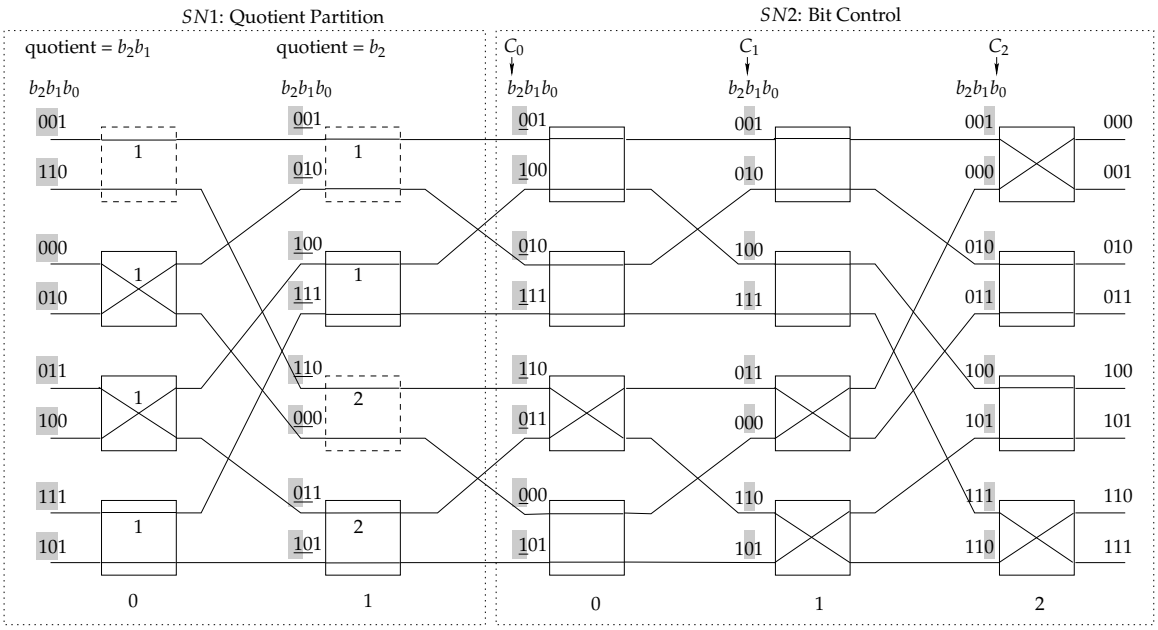


Figure 2.9: Routing on 8×8 Beneš network using Lee's algorithm.

cross. For all admissible permutations for an $n \times n$ inverse Baseline network (the same as $SN2$) these two control bits at any internal switch are always unequal, i.e., they are never both "0" or both "1", thus we can decide the control based on just the control bit at the upper input. It is a well-known result that this choice for the control bit realizes all admissible permutations, i.e., permutations which can pass without conflict, on a $n \times n$ inverse Baseline network (the same as $SN2$) [39, 34]. An example of this algorithm is shown in Figure 2.9 for the same permutation as was shown in Figure 2.8. For stage 0 of $SN1$, we set the switches so that the quotients given by bits b_2b_1 differ. For stage 1, we perform two partitions independently on the two blocks of switches marked 1 and 2 to form four total partitions. The highlighted bits for $SN1$ show the bits used to calculate quotients at the inputs. $SN2$ is bit controlled. The control bits are highlighted in Figure 2.9. For any partition on

a block of switches one switch can be set arbitrarily and thus are redundant, these switches are shown by dashed lines in Figure 2.9.

2.3.4 Routing with Arbitrary Switch Settings

We use Lee's algorithm to show that for any input permutation, there exist multiple network states for $SN1$ which lead to an output which does not block over $SN2$. In addition to this we will also prove that there is a class of input permutations for which any network state on $SN1$ leads to a permutation output which does not block over $SN2$. This means that for such input permutation assignments, we can set the switches in $SN1$ arbitrarily and still get a permutation which is admissible over $SN2$. We begin with the following lemma:

Lemma 2.1. *Given any input permutation on an $n \times n$ Beneš network, there are at least $2^{n/2-1}$ distinct switch configurations or network states for $SN1$ such that the resulting permutations are admissible over $SN2$.*

Proof. In the routing algorithm for $SN1$, one switch per block is redundant as one switch can be set arbitrarily. 2^k quotient partitions are performed in stage k, $0 \leq k \leq p-2$. Thus a total of $\sum_{k=0}^{p-2} 2^k = 2^{p-1} - 1 = n/2 - 1$ switches can be set arbitrarily in $SN1$ such that the resulting permutation is passable through $SN2$. Since each switch can be set in either of two states: through or cross, this implies that the corresponding total number of network states for $SN1$ are $2^{n/2-1}$. □

The redundancy of $n/2 - 1$ switches in a $\mathcal{B}_p$ matches the result of Waksman for an asymptotically minimum switch count for a rearrangeable network [41]. Note

that, for particular kinds of permutations, more than $n/2-1$ switches in $SN1$ may be set arbitrarily. Whenever, at a 2×2 switch, the quotients at both the inputs are equal then according to the routing algorithm in Section 2.3.3.1 the switch can be set in an arbitrary state, hence we can have more than $n/2-1$ switches set arbitrarily.

Until now we have considered network states which for any input permutation, lead to routing in a non-blocking fashion. We now address a different problem: Identify a class of permutations for which *all* the switches in $SN1$ can be set arbitrarily such that the resulting permutation can be passed without conflicts over $SN2$.

Let the p-bit binary representation of a number x, $0 \leq x \leq 2^p - 1$, be $x_{p-1} \cdots x_0$. The substrings of the binary string for x are represented as follows:

$$(x)_a^b = x_b x_{b-1} \cdots x_{a+1} x_a, \quad \text{where } 0 \leq a \leq b \leq p-1 \tag{2.31}$$

For any two numbers x and y

$$(x)_a^b = (y)_a^b \Rightarrow x_a = y_a;\ x_{a+1} = y_{a+1}; \ldots;\ x_b = y_b \tag{2.32}$$

Also let $\pi : i \rightarrow \pi(i)$ be the input permutation to the Beneš network, i.e., input i is to be routed to output $\pi(i)$.

Theorem 2.4. *Let π be an input permutation to the $n \times n$ Beneš network for which the following conditions are true:*

$$(i)_k^{p-1} = (j)_k^{p-1} \Rightarrow (\pi(i))_k^{p-1} = (\pi(j))_k^{p-1}, \textit{ for all } i \neq j \textit{ and all } 1 \leq k \leq p-1 \tag{2.33}$$

where $0 \leq i, j \leq n-1, n = 2^p$*, then for any arbitrary setting of the switches in* $SN1$*, the permutation can be routed without any conflicts over* $SN2$ *using bit controlled self-routing.*

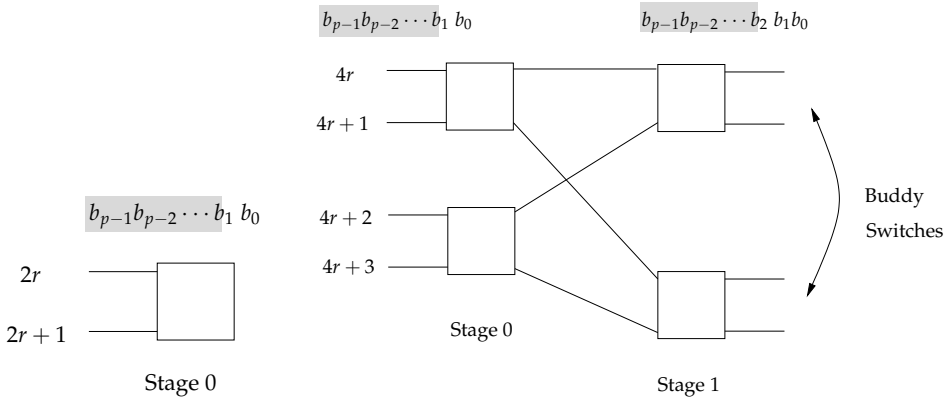
The total number of such permutations is equal to 2^{n-1}*.*

Proof. Recall, that in the routing algorithm for $SN1$, a switch can be set arbitrarily if the quotients of the output addresses at both the upper and lower inputs to a switch are the same. If the output address tag of a packet is $b = b_{p-1} \cdots b_1 b_0$ then in the partition calculation at stage k the quotient is $b_{p-1} \cdots b_{k+1}$. Thus a switch in stage k of $SN1$ is set arbitrarily if the $p-k-1$ most significant bits, $b_{p-1} \cdots b_{k+1}$, are the same for the packets at the upper and lower inputs.

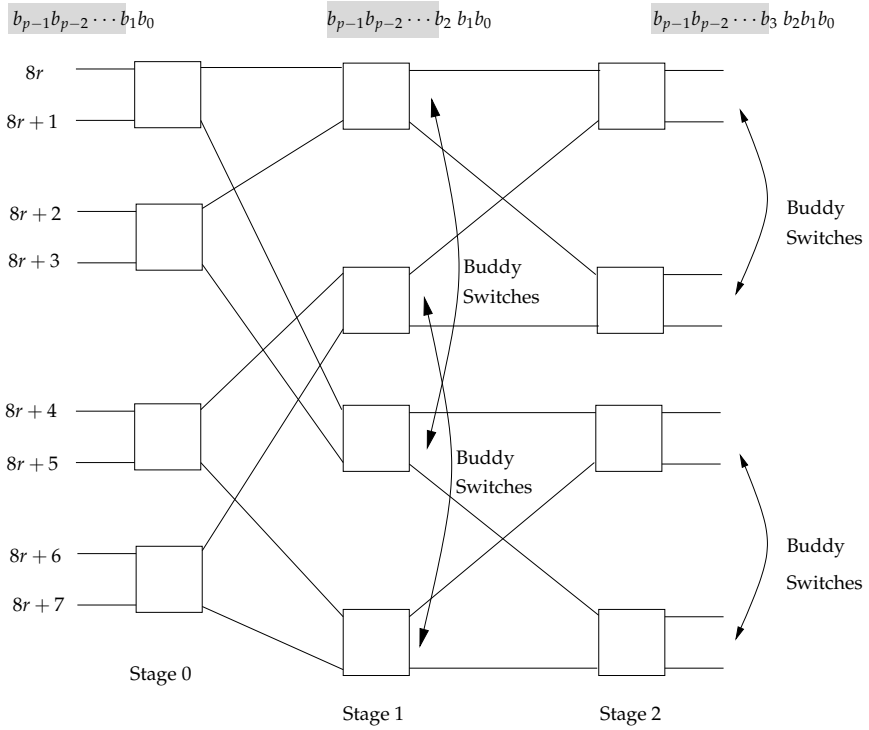
To prove this result we will proceed stage-by-stage, starting with stage $k = 0$.

$k = 0$: A switch in stage 0 is set arbitrarily if the bits $b_{p-1} \cdots b_1$ are equal for both input packets. Thus the output addresses can be different only in b_0. In Figure 2.10(a) we can see that the index of the inputs to a switch are of the form $2r$ and $2r+1$, i.e., they differ in only the least significant bit. Therefore if the switch input indexes differ in only the last bit then the corresponding address tags also differ in only the last bit, i.e., if $(i)_1^{p-1} = (j)_1^{p-1} \Rightarrow (\pi(i))_1^{p-1} = (\pi(j))_1^{p-1}$ for all inputs $i \neq j$ then all the switches in stage 0 are set arbitrarily. There are $2^{n/2} \cdot (n/2)!$ permutations which satisfy this condition as at each switch we can choose the output addresses in 2 ways and then rearrange these $n/2$ blocks of 2 numbers in $(n/2)!$ ways.

$k = 1$: Switches in a Beneš network form a "buddy" structure as seen in Fig-



(a) $k = 0$. (b) $k = 1$.



(c) $k = 2$.

Figure 2.10: Structure of Beneš network (proof for Theorem 2.4).

ure 2.10(b). Any two switches in stage s which share the same two parent switches in stage $s-1$ are called buddy switches. A switch in stage 1 is set arbitrarily if the bits $b_{p-1}\cdots b_2$ are equal at both inputs. Also buddy switches in stage 1 share the same parents in stage 0, thus at all four inputs to the buddy parents the bits $b_{p-1}\cdots b_2$ are the same. The indexes of the four inputs are of the form $4r, 4r+1, 4r+2, 4r+3$, i.e., the input indexes themselves share the $n-2$ most significant bits (MSBs). Combined with the condition for stage 0 this means that if $(i)_l^{p-1} = (j)_l^{p-1} \Rightarrow (\pi(i))_l^{p-1} = (\pi(j))_l^{p-1}$ for all inputs $i \neq j$ and $l = 1, 2$ then all the switches in stages 0 and 1 are set arbitrarily. There are $2^{n/2} \cdot 2^{n/4} \cdot (n/4)!$ permutations which satisfy this condition. This is derived as follows: the input permutation is divided into $n/4$ blocks of four numbers each, in which $p-2$ MSBs are the same. These blocks can be permuted in (n/4)! ways. In a block of four numbers, there are 2 blocks of two numbers which share $p-1$ MSBs, these two blocks can be rearranged in 2 ways for a total of $2^{n/4}$ ways to do so. There are two ways to rearrange numbers in each block of 2 numbers and a total of $n/2$ such blocks for a total of $2^{n/2}$ ways. Thus, the total number of permutations is $2^{n/2} \cdot 2^{n/4} \cdot (n/4)!$.

$k = 2$: A switch at stage 2 is set arbitrarily if the bits $b_{p-1}\cdots b_3$ are equal at both inputs. Using the buddy switch argument we can see that the four buddy parent inputs to stage 1 share the same bits $b_{p-1}\cdots b_3$ and hence the eight inputs to buddy parents in stage 0 also share the same bits $b_{p-1}\cdots b_3$. The indexes of these 8 inputs are of the form $8r, 8r+1, \ldots, 8r+7$ (see Figure 2.10(c)) and hence they share their $p-3$ most significant bits. Thus, combining this result with those for stages 0 and 1 we get: if $(i)_l^{p-1} = (j)_l^{p-1} \Rightarrow (\pi(i))_l^{p-1} = (\pi(j))_l^{p-1}$ for all inputs

$i \neq j$ and $l = 1,2,3$ then all the switches in stages $0,1$ and 2 are set arbitrarily. Using reasoning similar to the the $k = 2$ case, the number of such permutations is $2^{n/2} \cdot 2^{n/4} \cdot 2^{n/8} \cdot (n/8)!$.

Continuing in this fashion until $k = \log n - 2$ we get the conditions in Eqn. (2.33) and the total number of permutations as

$$\begin{aligned} & 2^{n/2+n/4+\cdots+n/(2^{\log n-1})} \cdot (n/2^{\log n-1})! \\ =& 2^{n(1-2^{-(\log n-1)})} \cdot (n \cdot 2^{-(\log n-1)})! \\ =& 2^{n(1-2/n)} \cdot (n \cdot 2/n)! \\ =& 2^{n-2} \cdot 2! = 2^{n-1} \end{aligned}$$ □

An input permutation which satisfies the conditions in 2.4 is the identity permutation, $\pi(i) = i$. We see that for this case the conditions in Eqn. (2.33) become: $(i)_k^{p-1} = (j)_k^{p-1} \Rightarrow (i)_k^{p-1} = (j)_k^{p-1}$, which is trivially true. Another input permutation for which Eqn. (2.33) holds is $\pi(i) = i + n/2 \bmod n$. Here $\pi(i)$ and i differ in only the most significant bit. This can be generalized: if $\pi(i)$ corresponds to the number obtained by complementing bits in i at some fixed positions, then π satisfies the conditions in Eqn. (2.33).

2.3.5 Output Quantum Assignment for the Beneš Network

We now derive results about some properties of the output quantum assignment for a quantum Beneš network in which packet routing is done using randomizing switches and bit controlled self-routing switches as described in Section 2.3.1.

Let the input to the $n \times n$ quantum Beneš network be a permutation pattern

$|P\rangle = |(r_0, a_0, d_0), \cdots, (r_{n-1}, a_{n-1}, d_{n-1})\rangle$, where $r_0 = r_1 = \cdots = r_{n-1} = 1$.

Then the output quantum assignment is of the form

$$\begin{aligned} &\sum_{j=0}^{T-1} \gamma_j |P_j\rangle |\Psi_{P_j}\rangle_{aux} \\ &= \sum_{j=0}^{T-1} \gamma_j |(r_0^j, a_0^j, d_0^j), \cdots, (r_{n-1}^j, a_{n-1}^j, d_{n-1}^j)\rangle |\Psi_{P_j}\rangle_{aux} \end{aligned} \tag{2.34}$$

where $|P_j\rangle = |(r_0^j, a_0^j, d_0^j), \cdots, (r_{n-1}^j, a_{n-1}^j, d_{n-1}^j)\rangle$ is the output assignment pattern corresponding to network state j. If $r_i^j = 1$, then $a_i^j = i$, for all i and j, i.e., all valid packets present in any pattern occur at their intended output destination. $|\Psi_{P_j}\rangle_{aux}$ is the state of the finite number of auxiliary and control qubits.

We are interested in the patterns at the output in which all the n valid classical packets in the input assignment pattern are routed to their destination output lines.

Let

$$\mathcal{S}_P = \left\{ |P_j\rangle |\Psi_{P_j}\rangle : r_i^j = 1, 0 \leq i \leq n-1, 0 \leq j \leq T-1 \right\}$$

be the set of patterns in the output which correspond to routing without contention for input permutation assignment pattern $|P\rangle$. In bit controlled self-routing on the last $\log n$ stages, the routing bits are altered to zero only when contention occurs (as shown in Eqn. (2.27) and Eqn. (2.28)), thus the patterns $\left|P_j\right\rangle$ in $\mathcal{S}_P$ encounter no contention. Hence, the following is true:

$$|A\rangle |\Psi_A\rangle \in \mathcal{S}_P \Leftrightarrow |A\rangle = |(1, 0, d_0), (1, 1, d_1), \cdots, (1, n-1, d_{n-1})\rangle \tag{2.35}$$

All packet patterns in $\mathcal{S}_P$ are the same, and correspond to the permutation map realized for input pattern $|P\rangle$. The elements of $\mathcal{S}_P$ can only be distinguished from the difference in the states of the auxiliary qubits. These auxiliary qubit states cor-

respond to switch configurations of the Beneš network for which the permutation map is realized for all the n valid packets in $|P\rangle$.

Denote the cardinality of $\mathcal{S}_P$ by $|\mathcal{S}_P|$. The following theorem follows as a direct consequence of lemma 2.1:

Theorem 2.5. *For any arbitrary input permutation pattern* $|P\rangle$,

$$|\mathcal{S}_P| \geq 2^{n/2-1} \tag{2.36}$$

$$\sum_{j:|P_j\rangle \in \mathcal{S}_P} |\gamma_j|^2 \geq 2^{n-1}/n^{n/2} \tag{2.37}$$

Proof. The quantum assignment at the output of the randomly routed $\log n - 1$ stages contains all permutation packet patterns corresponding to all the possible states of these stages. Thus, from lemma 2.1 there are at least $2^{n/2-1}$ patterns which can be passed without any contention in the last $\log n$ stages. Hence $|\mathcal{S}_P| \geq 2^{n/2-1}$.

The expression on the left in Eqn. (2.37) is the probability of observing, on measurement at the output, a pattern with n valid packets. As shown in Eqn. (2.29) there are a total of $n^{n/2(\log n-1)}$ unique permutation assignment patterns in the quantum output assignment after $\log n - 1$ randomizing stages of the $n \times n$ Beneš network, each of which has an equal probability of being observed. From the previous result we know that at least $2^{n/2-1}$ patterns at the output of the $\log n - 1$ randomizing stages are passed without conflict over the last $\log n$ stages. In these patterns all n packets are valid. Thus, the probability of these assignment patterns being observed on measurement of the output quantum assignment is at least $2^{n/2-1}/2^{n/2(\log n-1)} = 2^{n-1}/n^{n/2}$. □

Theorem 2.5 holds for any arbitrary input permutation assignment. We identify

a class of input permutation assignment patterns for which the probability lower bound in Theorem 2.5 is equal to 1.

We use the notation for binary strings defined in Eqns. (2.31) and (2.32) in the theorem below.

Theorem 2.6. *Let* $|P\rangle = |(r_0, a_0, d_0), \cdots, (r_{n-1}, a_{n-1}, d_{n-1})\rangle$ *be an input permutation assignment pattern for which the following conditions are true:*

$$(i)_k^{p-1} = (j)_k^{p-1} \Rightarrow (a_i)_k^{p-1} = (a_j)_k^{p-1}, \quad \forall\, i \neq j, 1 \leq k \leq p-1 \tag{2.38}$$

where $0 \leq i, j \leq n-1, n = 2^p$, *then*

$$|\mathcal{S}_P| = 2^{n/2(\log n - 1)} = (n/2)^{n/2} \tag{2.39}$$

$$\sum_{l:|P_l\rangle \in \mathcal{S}_P} |\gamma_l|^2 = 1 \tag{2.40}$$

There are 2^{n-1} *unique permutation maps* $i \rightarrow a_i$ *for which all the conditions in Eqn.* (2.33) *are satisfied.*

Proof. From Theorem 2.4 we get that for any permutation input pattern in which the output addresses satisfy the conditions in Eqn. (2.38) any arbitrary configuration of switches in the first $\log n - 1$ stages will result in an output pattern which can be routed without conflicts over the last $\log n$ stages. This implies that all the permutation patterns corresponding to the $2^{n/2(\log n-1)} = (n/2)^{n/2}$ network states are routed without conflicts in the last $\log n$ stages. Hence, Eqn. (2.39) and Eqn. (2.40) follow. □

Although we proved the above results assuming a permutation input assignment pattern, they can be easily generalized to non-contending assignment pat-

terns. The only difference is that the conditions in Eqn. (2.38) have to hold only for the inputs having valid packets.

Chapter 3

Self-Routing Quantum Sparse Crossbar Concentrators

3.1 Overview

An (n, m)-concentrator is a switching network with n inputs and m outputs, $1 \leq m \leq n$ in which any set of k inputs can be routed in parallel to some k outputs, $1 \leq k \leq m$. Concentrators as connectors are used to realize unordered connections, i.e., the specific input-output mapping within the k-sets mentioned above is not relevant as long as the entire input set can be passed through to an equal number of outputs. This is unlike routers or switches which enable ordered connections in which the output to which an input is to be routed is specified beforehand. This notion of concentration ties with matching sets in graphs. In graph theoretic terms a concentrator can be defined as follows: a graph $\mathcal{G}$ with n input vertices and m output vertices is called an (n, m)-concentrator, if every k of the n inputs, $1 \leq k \leq m$, has a matching set among the m outputs of $\mathcal{G}$. A matching can be viewed as a set of edge-disjoint paths between inputs and outputs.

A family of concentrators, called *sparse crossbar concentrators*, arises when the concentrator graph $\mathcal{G}$ can be represented as a bipartite graph. Any such bipartite graph $\mathcal{G}$ can be realized using a grid or matrix of crosspoints with m rows and n columns where a crosspoint exists between column i and row j if there exists an

edge between input i and output j in the bipartite graph representation $\mathcal{G}$.

Explicit sparse crossbar concentrator structures with theoretically minimum cost or crosspoint count for any arbitrary values of n and m are well-known [24, 25]. We will design quantum concentrators based on such optimal designs to take advantage of their low cost and simple single stage structure.

We first give an interpretation of concentration in a quantum network. In a classical concentrator network, packets for concentration are assigned at inputs and these input packet assignments can be issued only one assignment at a time. However, in a quantum concentrator, packets consist of quantum bits and thus represent a superposition of assignment patterns of packets which can be concentrated all at once by such a network due to the principle of quantum parallelism. This aspect is what distinguishes quantum concentrators from their classical counterparts. For example, consider a concentrator in which three inputs, say X, Y and Z, have packets which have to be concentrated. Suppose input X has 2 packets represented as x_1 and x_2, input Y has one packet y_1 and input Z has two packets z_1 and z_2. Y generates a "pure" packet while X and Z generate quantum packets by creating a superposition of both their respective packets and all three input sources then push their packets into a quantum concentrator. The outcome is that all the four possible input packet patterns: (x_1, y_1, z_1), (x_1, y_1, z_2), (x_2, y_1, z_1) and (x_2, y_1, z_2) are routed in parallel and the output is a superposition of four packet patterns each of which corresponds to the output obtained by concentrating one of the input packet patterns.

We use the optimal (n, m)-sparse crossbar concentrators described in [24] and

[25], known as the fat-slim and banded concentrators, as a basis for the design of $n \times m$ quantum sparse crossbar concentrators which we refer to as QSC(n, m). In the process of designing a QSC(n, m) we address some issues particular to quantum systems. One such issue is the reversibility constraint of quantum information processing. All quantum operations are inherently reversible in nature. This notion of reversibility is exactly the same as that commonly understood for any input-output mapping, i.e., given the output state of a quantum system, the corresponding input state can be uniquely determined. A "rectangular" structure like an (n, m)-sparse crossbar concentrator where the number of inputs, n, is not equal to the number of outputs, m, is inherently non-reversible. We devise a way to make (n, m)-crossbar concentrators "square" by using additional lines on the input and output sides of such concentrators and ensuring that valid packets for concentration are routed only among the original n inputs and m outputs.

Additionally, in a crossbar concentrator a subset of inputs, say $\mathcal{I}_s$ can, in general, be matched to multiple subsets of outputs and a subset of outputs, say $\mathcal{O}_s$ can be the matching for multiple input subsets. Even when $\mathcal{O}_s$ is the only matching for $\mathcal{I}_s$, there may be multiple settings for the internal crosspoints which realize this matching. As a simple example, consider an (n, m)-concentrator in which a k-input subset and a k output subset are interconnected by a $k \times k$ full crossbar, $k \leq m$. Also assume that the k inputs in the k-input subset are not connected to any other outputs. Then obviously this k-input subset can be matched to only one output set of size k but all possible $k!$ one-to-one maps are possible.

Thus, to ensure reversibility a routing algorithm is needed to determine the

crosspoint settings which fix the output matching subset for a given input subset. It is critical to have a self-routing algorithm for quantum concentration in which the state of a crosspoint is determined by using only the local information from the incoming packet headers. A centralized routing algorithm requires external control, which may not be feasible in quantum systems. Therefore even though efficient centralized routing algorithms for optimal crossbar concentrators with $O(\log n)$ delay for n tree-connected processors and $O(n \log n)$ delay for a single processor are known [25], [49], they cannot be adopted for quantum concentrators. Another advantage of self-routing packets is that the control quantum bits used to configure the crosspoint settings can be restored back to their original states easily thus preventing loss of information due to decoherence. We first give the design of quantum crosspoints which can be dynamically switched and then we describe a self-routing scheme which can be implemented on sparse crossbars built using our crosspoints. We prove the correctness of this scheme for fat-slim and banded sparse crossbar structures by showing that any k-input subset, $k \leq m$, has a matching k-output subset and giving the specific input-output mapping for that matching. We also show how to restore the control quantum bits for the quantum crosspoints back to their original state.

The rest of this chapter is organized as follows. In Section 3.2 we define some basic concepts related to quantum concentration. In Section 3.3 we give a brief overview of classical sparse-crossbar concentrators and define the functionality of quantum sparse-crossbar concentrators in Section 3.4. In Section 3.5 we present the design of quantum sparse crossbar concentrators and describe a self-routing

algorithm for such concentrators. In Section 3.6 we prove the correctness of this algorithm for the optimal banded and fat-slim quantum crossbar concentrators. In Section 3.7 we give an example to elucidate the concentration process on a quantum sparse crossbar concentrator. We address the issues of routing more than m packets on an n, m-quantum crossbar concentrator in Section 3.8 and restoring the state of auxiliary qubits to prevent decoherence in Section 3.9 respectively. The cost analysis of the gate count and routing delay are covered in Section 3.10. Section 3.11 concludes the chapter.

3.2 Definitions

We define quantum packets, quantum concentration assignments and the mapping performed by a quantum sparse crossbar concentrator in this section. Note that in the rest of this chapter whenever we write $|a\rangle$, where a is a binary variable, then this represents the state of a quantum bit. The binary variable itself is indicated as a.

A concentrator can connect any k-subset of inputs to some k outputs. Since the specific input-output map is not specified, no address bits are required in the packet header. Thus, using the earlier notation, a quantum packet input to a concentrator can be represented in the following form:

$$\sum_{i=1}^{m} \alpha_i \, |r_i, d_i\rangle \tag{3.1}$$

where $|\alpha_i|^2 = p_i$ is the probability with which the n_d-bit data packet d_i is to be concentrated and the routing bit, r_i, is set as $r_i = 1$. The length of the quantum

packet is $n_d + 1$ qubits. The individual components of the quantum packet, the bit strings (r_i, d_i) are the *classical packets*.

We refer to these strings as classical packets as they represent a basis state (with no superposition) of the constituent qubits and any group of quantum bits in a basis state are conceptually equivalent to a group of classical bits having the same value.

As before, the absence of a quantum packet or an empty line is indicated by setting the routing quantum bit to $|0\rangle$. There are no address bits in the packet, so we can consider the data portion of the classical packet for the concentrator as a combination of the address and data strings of a packet for a quantum switching network as described in Eqn. (2.1).

If the input source has no packets to concentrate then the empty line is indicated by a single $n_d + 1$-bit string in which the routing bit is set to 0, the data bits can be set to any arbitrary values.

An input *concentration pattern* for an n-input concentrator is a sequence of classical packets, each of which belongs to a quantum packet on the n inputs from top to bottom. A *quantum concentration assignment* is a superposition of a set of concentration patterns. We can define these terms formally as follows:

Definition 3.1 (Quantum concentration assignment)**.** A *quantum concentration assignment* for an n-input concentrator is a superposition of a set of t concentration patterns of the form:

$$\sum_{j=1}^{t} \gamma_j \left| (r_{j,1}, d_{j,1}), \cdots, (r_{j,n}, d_{j,n}) \right\rangle \tag{3.2}$$

where the *concentration pattern* $\left|(r_{j,1}, d_{j,1}), \cdots, (r_{j,n}, d_{j,n})\right\rangle$ is comprised of n classical packets in which $(r_{j,i}, d_{j,i})$ is the jth classical packet at input i. $|\gamma_j|^2$ is the probability of the jth concentration pattern being realized with $\sum_{j=1}^{t} |\gamma_j|^2 = 1$.

If the quantum packets at all the inputs are independent and of the kind given in Eqn. (3.1) then the quantum assignment can be expressed as a tensor product of the quantum packets as follows:

$$\bigotimes_{i=1}^{n} |Q_i\rangle = \bigotimes_{i=1}^{n} \left(\sum_{j=1}^{t_i} \alpha_{ij} \left| r_{ij}, d_{ij} \right\rangle \right) \tag{3.3}$$

where $|Q_i\rangle = \sum_{j=1}^{t_i} \alpha_{ij} \left| r_{ij}, d_{ij} \right\rangle$ is the quantum packet on input i. The tensor product in Eqn. (3.3) expanded to a quantum concentration assignment of the form in Eqn. (3.2) contains $\prod_{i=1}^{n} t_i$ concentration patterns.

As an example, consider three inputs indexed by $1, 2$ and 3 having the quantum packets: $\frac{1}{\sqrt{2}}(|1, d_{11}\rangle + |1, d_{12}\rangle)$, $|1, d_{21}\rangle$ and $\frac{1}{\sqrt{3}} |1, d_{31}\rangle + \frac{\sqrt{2}}{\sqrt{3}} |1, d_{32}\rangle$ respectively. Then the quantum concentration assignment is given by:

$$\begin{aligned} & \frac{1}{\sqrt{2}} (|1, d_{11}\rangle + |1, d_{12}\rangle) \otimes |1, d_{21}\rangle \otimes \left(\frac{1}{\sqrt{3}} |1, d_{31}\rangle + \frac{\sqrt{2}}{\sqrt{3}} |1, d_{32}\rangle \right) \\ = \; & \frac{1}{\sqrt{6}} |(1, d_{11}), (1, d_{21}), (1, d_{31})\rangle + \frac{1}{\sqrt{3}} |(1, d_{11}), (1, d_{21}), (1, d_{32})\rangle \\ & + \frac{1}{\sqrt{6}} |(1, d_{12}), (1, d_{21}), (1, d_{31})\rangle + \frac{1}{\sqrt{3}} |(1, d_{12}), (1, d_{21}), (1, d_{32})\rangle \end{aligned} \tag{3.4}$$

Thus, the quantum concentration assignment consists of a superposition of four concentration patterns, two of which have a probability $1/3$ and the other two a probability $1/6$ of being observed on measurement. A quantum concentrator can route such patterns contained in the input quantum concentration assignment in parallel.

3.3 Classical Sparse Crossbar Concentrators

An (n, m)-sparse crossbar network is a matrix of crosspoints or switches with m rows and n columns. Each crosspoint acts as a simple 2×2 switch which can either swap the data on its two inputs onto its outputs or pass them through unchanged. We refer to these two states of the crosspoint as the "cross" state and the "through" state respectively.

An (n, m)-sparse crossbar concentrator is an (n, m)-sparse crossbar in which any k inputs, $k \leq m$, can be routed over non-intersecting paths to some k outputs. Any sparse crossbar network is a concentrator if its crosspoint distribution is such that the constraints of Hall's theorem are satisfied. This theorem is stated below:

Hall's Theorem [50]. *Let O be a finite set and let $Y_1, Y_2, \ldots, Y_r$ be arbitrary subsets of O. There exist distinct elements $y_i \in Y_i$, $1 \leq i \leq r$ if and only if the union of any k of $Y_1, Y_2, \ldots, Y_r$ contains at least k elements.*

Let the set O in the theorem denote the set of outputs of a sparse crossbar and $Y_1, Y_2, \ldots, Y_r$ represent the neighbor sets of some r inputs $s_1, s_2, \ldots, s_r$ respectively, i.e., Y_i is the subset of outputs in O to which input s_i can be connected, $1 \leq i \leq r$. Then if the union of $Y_1, Y_2, \ldots, Y_r$ contains at least r outputs for any choices of $s_1, s_2, \ldots, s_r$ in the input set, and any r, $1 \leq r \leq m$, then Hall's theorem implies that the sparse crossbar is a concentrator.

Nakamura and Masson in [23] gave a lower bound of $m(n - m + 1)$ crosspoints on the crosspoint complexity, i.e., number of crosspoints, for (n, m)-sparse crossbar concentrators by showing that each output needs to share crosspoints with at least

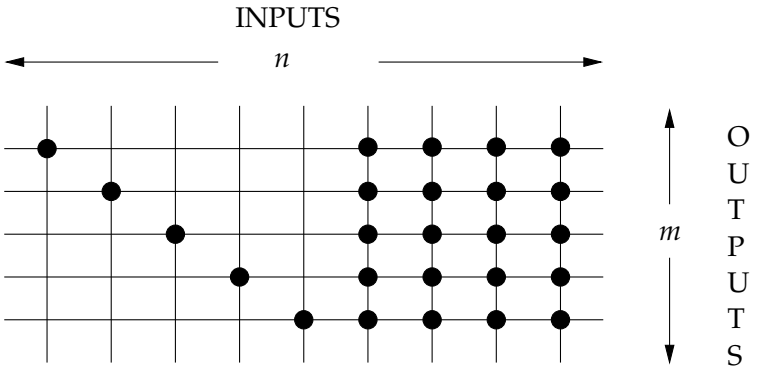
$n - m + 1$ inputs. Oruç et al. in [24] and [25] gave explicit crossbar structures of optimal concentrators which achieved this complexity bound for any n and m. They used Hall's theorem for distinct representatives to show that certain $n \times m$ sparse crossbar structures with exactly $m(n - m + 1)$ crosspoints can act as concentrators.

Two such optimal concentrators, the fat-slim and banded sparse crossbar concentrators, which we shall be using extensively, are shown in Figure 3.1. As seen in Figure 3.1(a), in a fat-slim crossbar concentrator, the input columns can be divided into a fat portion in which an input is connected to all outputs and a slim portion in which an input is connected to only one output with all slim inputs being connected to different outputs. In a banded crossbar concentrator (Figure 3.1(b)) all the crosspoints form a transverse band in the middle. Note that in these sparse crossbars each of the m outputs is connected to $n - m + 1$ inputs.

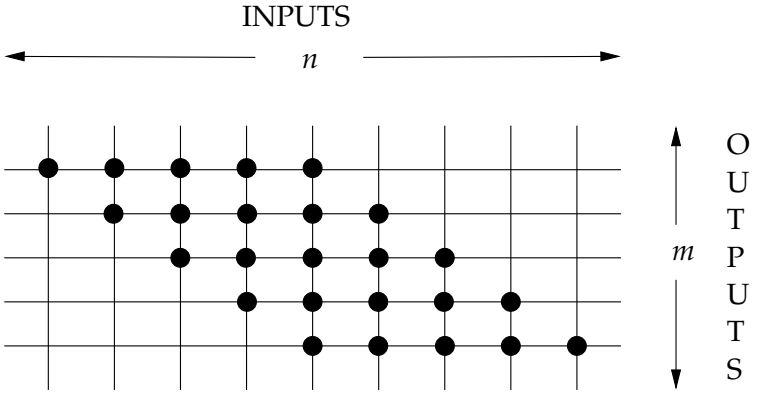
3.4 Quantum Sparse Crossbar Concentrators

An input concentration pattern for a concentrator is said to be *capacity achieving* if no greater than m packets in the pattern have their routing bits equal to 1, where m is the number of outputs of the concentrator. We call a quantum concentration assignment *capacity achieving* if all of its concentration patterns are capacity achieving. Also, we refer to packets with routing bit set to 1 as valid packets.

A quantum sparse crossbar network is obtained from a classical crossbar network by replacing the classical crosspoints by quantum crosspoints which can switch quantum packets. A quantum crosspoint can be viewed as a configurable



(a) Classical fat-slim crossbar concentrator, $n = 9, m = 5$.



(b) Classical banded crossbar concentrator, $n = 9, m = 5$.

Figure 3.1: Classical sparse crossbar concentrators.

2×2 switch which either swaps or passes through unchanged to its outputs the two quantum packets incident at its inputs. Reversibility in quantum systems implies, that for a quantum sparse crossbar, unlike a classical sparse crossbar, each crosspoint needs to have qubits coming in on each of its two inputs and qubits leaving on each of its outputs. As mentioned earlier in Section 2.1.1 an absence of a packet or an empty wire is indicated by a quantum bit string with the routing bit set to 0. Thus, in the quantum domain, for an $n \times m$ sparse crossbar network, the m empty wires coming in from the left can be represented by blocks of quantum bits in which the routing bit is set to 0. We can imagine m additional packet sources at these wires which generate quantum bit blocks in which the routing bit is always set to 0. The same reasoning can be applied to the n empty wires leaving the sparse crossbar at the bottom. This means that they can be viewed as n additional outputs and if the sparse crossbar is a concentrator then the exiting quantum bit strings on these outputs have their routing bits equal to 0 as long as the input pattern is capacity achieving. This in effect creates a "square" $(n+m) \times (n+m)$ crossbar network from a "rectangular" $n \times m$ network in which only the n inputs (on the top) can get valid packets. If the sparse crossbar is a concentrator and the input pattern is capacity achieving then all the valid packets are concentrated to the m outputs on the right hand side. If we number the inputs on the top $1, \ldots, n$ from left to right, the inputs on the left $n+1, \ldots, n+m$ from top to bottom, the outputs on the right $1, \ldots, m$ and outputs on the bottom $m+1, \ldots, m+n$ as shown in Figure 3.2 then:

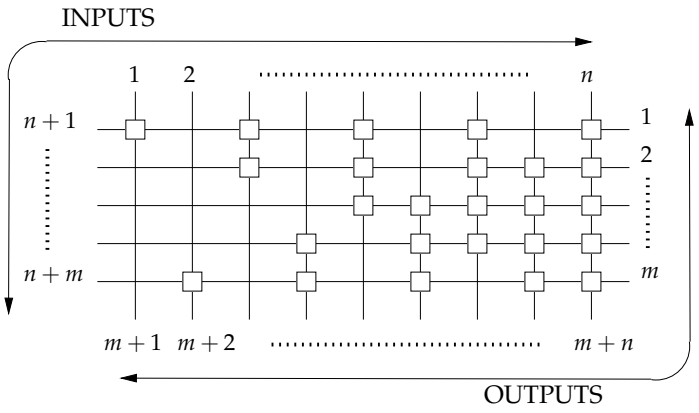


Figure 3.2: Numbering of inputs and outputs in an $n \times m$ quantum sparse crossbar network.

Definition 3.2 (QSC(n, m))**.** An $n \times m$ quantum sparse crossbar network ($n \geq m$) is called an (n, m)-*quantum sparse crossbar concentrator* or QSC(n, m) if, any capacity achieving input pattern with k valid packets, where $k \leq m$, is routed such that these packets appear on some k of the first m outputs. That is, for a capacity achieving input concentration pattern $|P\rangle = |(r_1, d_1) \cdots (r_n, d_n)\rangle$, a pattern $|(0, d_{n+1}) \cdots (0, d_{n+m})\rangle$ of m packets each with routing qubit set to $|0\rangle$ and a set of auxiliary qubits each of which is initialized to state $|0\rangle$, the following transformation occurs:

$$|\underbrace{(r_1, d_1) \cdots (r_n, d_n)}_{|P\rangle\text{: from top}}\rangle|\underbrace{(0, d_{n+1}) \cdots (0, d_{n+m})}_{\text{from left}}\rangle\,|00\cdots0\rangle_{aux} \xrightarrow{QSC(n,m)}$$

$$|\underbrace{(r'_1, d'_1) \cdots (r'_m, d'_m)}_{\text{on right}}\rangle|\underbrace{(0, d'_{m+1}) \cdots (0, d'_{n+m})}_{\text{bottom}}\rangle\,|\Psi_P\rangle_{aux} \quad (3.5)$$

where if $\mathcal{R}_1 = \{|r_i, d_i\rangle \; \forall\, r_i = 1\}$, $1 \leq i \leq n$, is the set of valid input packets and $\mathcal{R}_2 = \{|r'_j, d'_j\rangle \; \forall\, r'_j = 1\}$, $1 \leq j \leq m$, is the set of valid packets at the outputs then $\mathcal{R}_2 = \mathcal{R}_1$. Here $|00\cdots0\rangle_{aux}$ represents the state of a set of auxiliary qubits all of

which are in state $|0\rangle$.

In a crossbar network, a subset of inputs can potentially be matched to more than one set of outputs. Moreover, even if an input set can be matched to only one output set, it is possible that several one-to-one input-output maps within these sets realize the matching. Therefore, some form of a routing algorithm is needed to fix the matched output set and the input-output mapping for a capacity achieving input pattern. If a quantum crosspoint is configured by using only the information contained in the quantum packets incident on its inputs, then we call such a quantum crosspoint as a self-routing crosspoint. A quantum sparse crossbar concentrator built using such crosspoints is called a *self-routing quantum sparse crossbar concentrator*. Hence, a self-routing QSC(n, m) is a sparse crossbar network built using self-routing quantum crosspoints which realizes all maps of the kind given in Eqn. (3.5).

We describe such an algorithm in Section 3.5. The auxiliary qubits ensure reversible operation as their final state, $|\Psi_P\rangle$, encodes the state of the crossbar network, i.e., the states of the internal crosspoints or equivalently the input-output mapping.

3.5 Self-Routing Quantum Crosspoints

In this section we give the design of self-routing quantum crosspoints and a routing scheme for sparse crossbars composed of such crosspoints.

A self-routing $n \times m$ quantum sparse crossbar is derived from a classical sparse

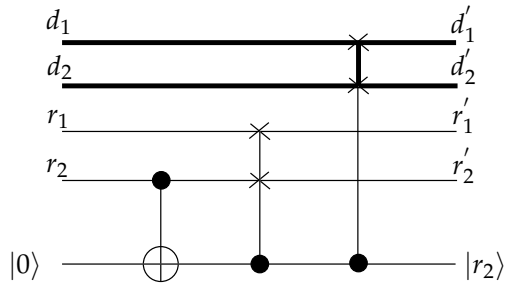


Figure 3.3: Circuit for the quantum crosspoint.

crossbar structure as follows: The crosspoints in the classical sparse crossbar are replaced by quantum crosspoints, the circuit for which is given in Figure 3.3. Here the upper input with packet $|r_1,\ d_1\rangle$ corresponds to the input line incident on a crosspoint from the top and the lower input with packet $|r_2,\ d_2\rangle$ corresponds to the input line coming in from the left. The data bit strings d_1, d_2 and the routing bits r_1, r_2 are shown separately for clarity. Each crosspoint uses an auxiliary control qubit initialized to state $|0\rangle$, which is then set according to the map in Table 3.1 and used to control the setting ("through" or "cross") of the switch gate. When the control qubit is set to state $|1\rangle$, the input packets get swapped and when the control qubit is in state $|0\rangle$, the input packets go through unchanged. In the crosspoint circuit the CNOT gate functions as a copier which sets the state of the control qubit to $|r_2\rangle$. This qubit is then used to control the two swap gates which switch r_1, r_2 and d_1, d_2 respectively.

Thus, the input-output mapping performed by the quantum crosspoint can be represented as:

$$|r_1, d_1, 0, d_2\rangle\,|0\rangle \xrightarrow[Through]{crosspoint} |r_1, d_1, 0, d_2\rangle\,|0\rangle$$

Inputs		Outputs				
r_2	r_1	r_2'	r_1'	d_2'	d_1'	State
0	0	0	0	d_2	d_1	Through
0	1	0	1	d_2	d_1	Through
1	0	0	1	d_1	d_2	Cross
1	1	1	1	d_1	d_2	Cross

Table 3.1: Input-output mapping for a quantum crosspoint.

$$|r_1, d_1, 1, d_2\rangle\,|0\rangle \xrightarrow[Cross]{crosspoint} |1, d_2, r_1, d_1\rangle\,|1\rangle \tag{3.6}$$

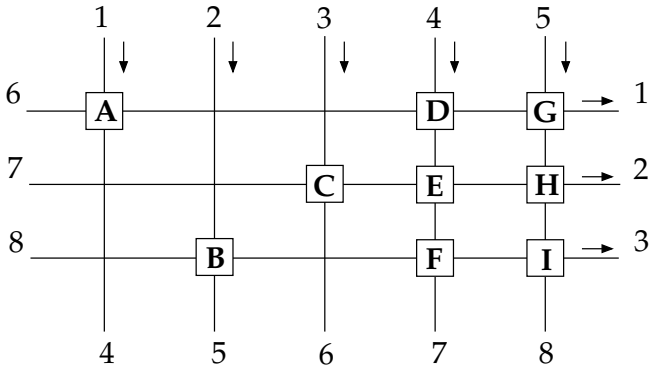
We can see that the auxiliary control qubit is always set to state $|r_2\rangle$, i.e., the packets are swapped when the routing qubit of the packet coming in from the left (r_2) is in state $|1\rangle$ and go through unchanged when this qubit is in state $|0\rangle$. Although the switch control is determined fully by just r_2, we cannot use its corresponding qubit by itself (without the auxiliary control qubit) to set the switch. This is because the qubit for r_2 is a part of the packet incident on the lower input and itself needs to be switched along with that packet. So although this qubit can act as a control qubit for all the rest of the qubits in the packet, it cannot be a control qubit for switching itself. Also, we are not using r_1 in the switch to determine the routing state, but this bit is still relevant as eventually at the output of the concentrator valid packets are identified by examining the routing bit values. Another reason to consider r_1 is the fact that crosspoint switches are interconnected to form the crossbar, and the packet from the upper input at one crosspoint switch may be incident at the lower input of a switch in later stages. In this scenario the qubit corresponding

to r_1 would be the auxiliary control for this later stage switch. As the state of the quantum crosspoint is determined fully by just using the information about the routing bits from the two input packets, the paths in the sparse crossbar are determined in a self-routing fashion.

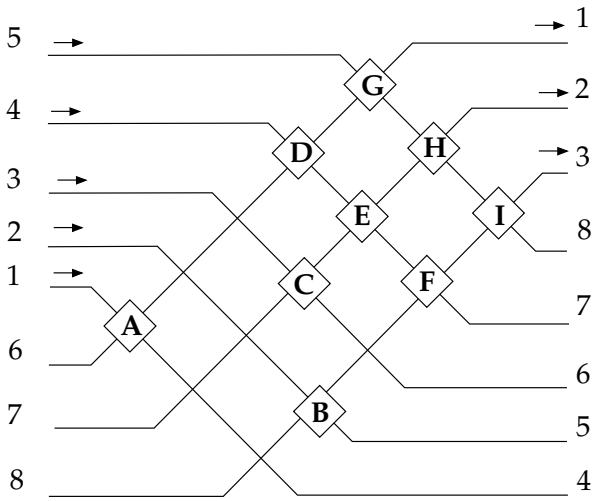
3.5.1 The Self-Routing Scheme

Starting from input 1, the routing of packets proceeds from top to bottom in a column for an input and then to next higher numbered input in the next column. The control qubit is not restored to its original state immediately, we give a method to restore the control qubit at the output of the concentrator in Section 3.9.

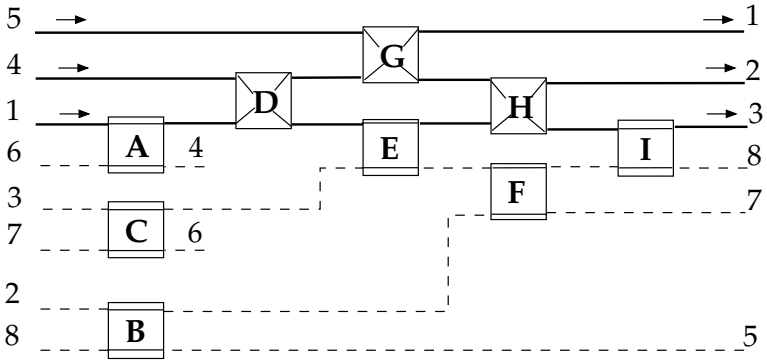
Note that unlike in the classical case, all n inputs (plus the m additional inputs from the left) have quantum bit strings incident on them. We distinguish the subset of inputs having packets for concentration by the setting the routing qubits in the headers of packets at these inputs to $|1\rangle$. A self-routing quantum concentrator derived from a classical fat-slim $(5, 3)$-sparse crossbar concentrator is given in Figure 3.4. The square boxes in Figure 3.4 indicated by letters A–I are quantum crosspoints. In this concentrator valid packets (packets with routing bit 1) come only on inputs 1–5, and if the input pattern is capacity achieving, they exit only on outputs 1–3. These inputs and outputs are indicated by arrows in the figure. In the process of self-routing, the crosspoints are traversed from top to bottom in a column and from left to right in a row. The crossbar in Figure 3.4(a) is redrawn in Figure 3.4(b) to show the sequence in which the crosspoints are traversed during routing. Any



(a) Crossbar structure.



(b) Crosspoint sequence in routing.



(c) Example for self-routing: Inputs 1, 4 and 5 are concentrated.

Figure 3.4: Self-routing fat-slim quantum sparse crossbar concentrator, QSC(5, 3).

sparse crossbar can be redrawn in this way if we rotate it counter-clockwise to make the diagonal vertical and write down the sequence of crosspoints encountered as we traverse from left to right. As a result of this redrawing we can see that all crosspoints lying on the diagonals which are oriented from top right to bottom left in the original crossbar form one vertical stage in the rotated version, e.g., G, E and H, F which lie on a diagonal in Figure 3.4(a) form independent stages in the crossbar in Figure 3.4(b). Note that D, C and B also lie on a single diagonal but since B and C are disjoint with each other and with D, they can be shifted from the second stage to the first, as shown in Figure 3.4(c). Figure 3.4(c) is Figure 3.4(b) with the inputs and outputs reordered to clearly show the planar and multi-stage structure of the sparse crossbar.

We now give an example to elucidate the self-routing process. Consider an input concentration pattern with valid packets at inputs 1, 4 and 5 for the crossbar shown in Figure 3.4(c), i.e., only packets appearing at inputs 1, 4 and 5 have routing bits set to 1. At all the other inputs the incident packets have routing bits set to 0. At crosspoint A the upper input, i.e., input 4 has a valid packet and the lower input has a packet with routing bit set to 0. Thus, this situation corresponds to the $r_1 = 1, r_2 = 0$ case in Table 3.1 and A is set to pass the packets through unchanged. Proceeding to the next stage, at crosspoint D, the routing bit of the packets at both the lower and upper inputs is 1, hence D swaps its input packets onto its outputs. Continuing in this way, the packet from input 1 takes the path A→D→G to output 1. Similarly the packets from input 4 and input 5 take the paths D→E→H and G→H→I to outputs 2 and 3 respectively. The switch settings and the packet routes

are shown in Figure 3.4(c). The routes taken by the valid packets are shown by solid lines while the routes taken by packets with routing bit equal to 0 are shown by dashed lines. Note that the path lengths between all the inputs and outputs are not equal, hence we need to introduce appropriate delay to make them all equal, accordingly, all output lines are extended till the end. Output lines 4 and 6 are not extended just to maintain clarity in the drawing but may be considered to extend till the last output stage.

The intuition behind the routing scheme is as follows: In the rotated crossbar as shown in Figure 3.4(b) the upper and lower input lines to a switch correspond to the lines coming in to the same switch from the top and left respectively in the original unrotated version. Similarly the upper and lower outputs in the rotated switches correspond to the lines going out to the right and bottom respectively in the original unrotated switches. Hence, giving routing priority to the packet at the lower input in a quantum crosspoint is equivalent to routing according to the packet coming in from the left in the unrotated crossbar. If the routing bit of the lower packet is 1 then the switch is set in a cross state, this is equivalent to passing a valid packet coming in from the left to the right irrespective of the packet incident from the top. Thus, once a valid packet is routed from the top to the right, it goes through unimpeded to the end of the row, in other words, once an input is matched to an output this decision remains unchanged for the subsequent duration of the routing process. The crosspoint is set in a through state when the packet at the lower input in the rotated switch has its routing bit equal to 0, this corresponds to a turn from left to the bottom in the unrotated crossbar. This observation combined

with the previous argument means that in a column of the unrotated crossbar a packet gets passed from top to bottom from one row to the next until it encounters a crosspoint where the packet from the left has a routing bit set to 0.

This self-routing scheme can be used for any sparse crossbar structure, it is not limited to concentrators. An interesting question to ask is whether all known optimal sparse crossbar concentrator structures allow concentration when a self-routing scheme of the form described above is used to route the packets. In the subsequent sections we prove that while this is not true for all optimal sparse crossbar concentrators, it holds for fat-slim and banded crossbars.

3.6 Self-Routing on Quantum Sparse Crossbar Concentrators

We prove that the self-routing scheme described in Section 3.5 concentrates packets up to the output capacity of the concentrator, for two families of sparse crossbar structures, namely the fat-slim and banded crossbars. All the proofs are given for a quantum input assignment consisting of a single concentration pattern. This is sufficient as the linearity of quantum systems implies that, for any general input quantum concentration assignment, these results apply to every concentration pattern contained in its superposition and hence to the entire input assignment. We first introduce some notation that will be used in the rest of the chapter.

3.6.1 Notation

For a quantum (n, m)-sparse crossbar network ($n \geq m$) in which packets are routed using our self-routing scheme, we use the following notation:

1. The set of inputs is denoted by $\mathcal{I} = \{1, 2, \ldots, n\}$ and the output set by $\mathcal{O} = \{1, 2, \ldots, m\}$ where $1 \leq m \leq n$.

2. The $n \times m$ adjacency matrix, $\boldsymbol{A}$, is given by:

$$\boldsymbol{A} = \{a_{ij}\} = \begin{cases} 0, & \text{no crosspoint between input } i \text{ and output } j, \\ 1, & \text{crosspoint between input } i \text{ and output } j. \end{cases}$$

3. $\mathcal{A}_i = \{j : \text{for all } a_{ij} = 1, 1 \leq i \leq n\}$, is the neighbor set for input i, i.e., the set of outputs which can be connected to input i.

4. $\mathcal{X}_r = \{x_1, x_2, \ldots, x_r\}$ is an ordered set of r distinct inputs, i.e., $x_i \in \mathcal{I}$, $1 \leq i \leq r$, and $x_1 < x_2 < \cdots < x_r$ for all $r = 1, \ldots, m$.

5. $\mathcal{Y}_r = \{y_1, y_2, \ldots, y_r\}$ is the set of outputs to which the inputs in $\mathcal{X}_r$ are matched using self-routing with output y_i being matched to input x_i, where, $y_i \in \mathcal{A}_{x_i}$, $i = 1, \ldots, r$. If input x_i can not be matched to any output then $y_i = \emptyset$, the empty set.

6. We denote subsets of $\mathcal{Y}_r$ as follows: $\mathcal{Y}_0 = \emptyset$, the empty set, $\mathcal{Y}_b = \{y_1, y_2, \ldots, y_b\}$, and $\mathcal{Y}_a^b = \{y_a, y_{a+1}, \ldots, y_b\}$, $\forall\, a \leq b \leq r$.

We now show that using the self-routing scheme described in Section 3.5, certain families of sparse crossbar concentrators, namely fat-slim and banded crossbar

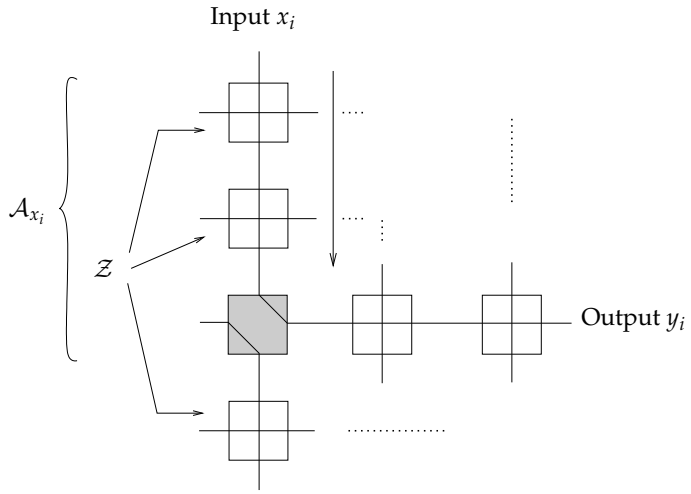


Figure 3.5: Output matching y_i for input x_i.

concentrators can correctly route any capacity achieving input pattern.

Lemma 3.1. *Let $\mathcal{Z} \subseteq \mathcal{O}$ be the subset of outputs to which packets have already been matched, using self-routing, before routing begins on input x_i, $1 \leq i \leq r$. Then, the output y_i, to which input x_i is matched is given by*

$$y_i = \min_z \{z \in \mathcal{A}_{x_i} \cap \mathcal{Z}'\}$$

where $\mathcal{Z}' = \mathcal{O} \setminus \mathcal{Z}$ is the complement of the set $\mathcal{Z}$.

Proof. All outputs which have been concentrated to (matched) before routing the packet on input x_i, (i.e., in $\mathcal{Z}$) correspond to rows which have a packet with routing bit $r = 1$ coming from the left on them. We know from the table for the auxiliary qubit (Table 3.1) that all crosspoints in rows corresponding to set $\mathcal{Z}$ will be set to the "cross" state, see Figure 3.5. This means that only a crosspoint from the set of rows corresponding to $\mathcal{Z}'$ will be set to the "through" state. This in turn means $y_i \in \mathcal{Z}'$. Also, obviously, $y_i \in \mathcal{A}_{x_i}$. Thus, $y_i \in \mathcal{A}_{x_i} \cap \mathcal{Z}'$. Since the routing process proceeds

from the top to bottom, i.e., in increasing order of row/output number, the lowest numbered available output is matched. Hence, $y_i = \min_z\{z \in \mathcal{A}_{x_i} \cap \mathcal{Z}'\}$. □

Lemma 3.2. *Let y_i be the output to which input x_i is matched, $1 \leq i \leq r$. Then*

$$y_i = \min_z \{z \in \mathcal{A}_{x_i} \cap (\mathcal{Y}_{i-1})'\}$$

Proof. As the routing of packets at inputs is initiated in increasing order of input index from left to right, the packets at inputs $x_1, \ldots, x_{i-1}$ are routed before routing starts on input x_i. Thus, the set of outputs matched before routing starts on input x_i is $\mathcal{Y}_{i-1}$. Substituting $\mathcal{Z} = \mathcal{Y}_{i-1}$ in Lemma 3.1 we get the desired result. □

Lemma 3.2 essentially asserts that, at any input, the first or lowest numbered unmatched output is selected as a match for that input during self-routing. If an input x_i cannot be matched to any output in its neighbor set $\mathcal{A}_{x_i}$ then $y_i = \emptyset$.

Lemma 3.3. *If $i \neq j$ and*

$$y_i = \min_z \{z \in \mathcal{A}_{x_i} \cap (\mathcal{Y}_{i-1})'\} \neq \emptyset$$

$$y_j = \min_z \{z \in \mathcal{A}_{x_j} \cap (\mathcal{Y}_{j-1})'\} \neq \emptyset.$$

then $y_i \neq y_j$, $1 \leq i, j \leq r \leq m$.

Proof. Without loss of generality, assume $i < j$, then $y_i \in \{y_1, \ldots, y_{j-1}\} = \mathcal{Y}_{j-1}$. Also $y_j = \min_z\{z \in \mathcal{A}_{x_j} \cap (\mathcal{Y}_{j-1})'\}$ implies that $y_j \in (\mathcal{Y}_{j-1})'$. Hence, $y_i \neq y_j$. □

Lemma 3.3 implies that the non-empty elements of $\mathcal{Y}_r = \{y_1, \ldots, y_r\}$ are all distinct. Thus, if all the elements of $\mathcal{Y}_r$ exist, i.e., they are non-empty, then $\mathcal{Y}_r$ forms a matching for the inputs in the set $\mathcal{X}_r = \{x_1, \ldots, x_r\}$. Moreover, if such a matching

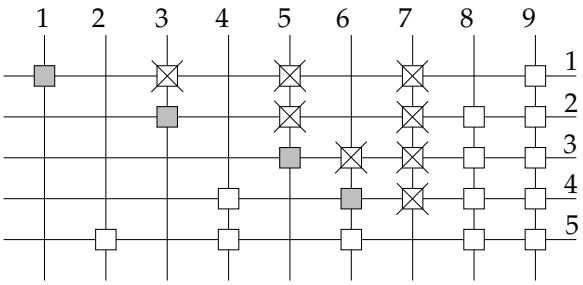


Figure 3.6: Conflict in self-routing.

exists for every r-input subset of a (n, m)-sparse crossbar network, $r \leq m$, then the network is an (n, m)-concentrator. From Lemma 3.2 the matching $\mathcal{Y}_r$ corresponds to the outputs chosen by self-routing, therefore, the concentrator thus obtained is self-routable. The theorem below follows:

Theorem 3.1. *An (n, m)-sparse crossbar network is a self-routing concentrator if*

$$y_i = \min_z \{z \in \mathcal{A}_{x_i} \cap (\mathcal{Y}_{i-1})'\} \neq \emptyset$$

for all $i = 1, \ldots, r$, and for every r-input ordered subset $\mathcal{X}_r = \{x_1, \ldots, x_r\}$, $1 \leq r \leq m$.

A quantum concentrator derived by replacing classical crosspoints by quantum crosspoints in a classical sparse crossbar concentrator is not always self-routable by the algorithm described above. One such scenario is shown in Figure 3.6. For the sparse crossbar in this figure the union of any k columns (or equivalently input neighbor sets) contains crosspoints in least k distinct rows (or outputs) for all $k = 1, \ldots, 5$, thus by Hall's theorem it is a $(9, 5)$-concentrator. The set of inputs with packets to be concentrated is given by $\mathcal{X}_5 = \{1, 3, 5, 6, 7\}$. The output to which an input is matched by following the self-routing scheme is indicated by the shaded crosspoints, therefore, input 1 is matched to output 1, input 3 to output 2 and

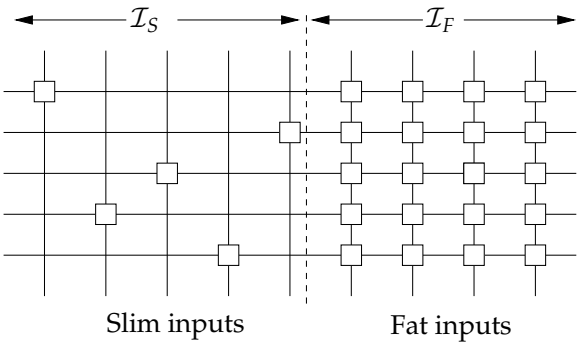


Figure 3.7: Partitions of fat-slim crossbar.

so on. The crossed out crosspoints indicate the outputs to which an input is not matched during self-routing. The matched outputs for the first four inputs in $\mathcal{X}_5$ form the set $\mathcal{Y}_4 = \{1, 2, 3, 4\}$. Thus for input 7 (which is the fifth input in $\mathcal{X}_5$) we get $y_5 = \min_z \{z \in \mathcal{A}_7 \cap (\mathcal{Y}_4)'\} = \min_z \{z \in \{1, 2, 3, 4\} \cap \{5\}\} = \emptyset$. Hence, this concentrator can not self-route all input subsets using the algorithm we described.

We now show that fat-slim and banded sparse crossbar concentrators can self-route by showing that for these concentrators Theorem 3.1 is always satisfied. As part of the proofs we give explicit expressions for the input-output mapping realized for concentration on these structures.

3.6.2 Self-Routing Fat-Slim QSC(n, m)

Definition 3.3 (Fat-Slim Crossbar)**.** An (n, m)-sparse crossbar network is called fat-slim if we can partition the input set $\mathcal{I}$ into two subsets: $\mathcal{I}_S$ (slim inputs) and $\mathcal{I}_F$ (fat inputs) as shown in Figure 3.7 with neighbor sets for input i described as follows:

$$i \in \mathcal{I}_S \Leftrightarrow 1 \leq i \leq m;\ \mathcal{A}_i = \pi(i) \tag{3.7a}$$

$$i \in \mathcal{I}_F \Leftrightarrow m < i \leq n;\ \mathcal{A}_i = \{1, \ldots, m\} = \mathcal{O} \tag{3.7b}$$

where π is a permutation on the elements of the set $\{1, \ldots, m\}$.

Every fat-slim (n, m)-sparse crossbar is an optimal (n, m)-concentrator and has $m(n - m + 1)$ crosspoints [24]. We now show that any capacity achieving input pattern can be self-routed on a fat-slim sparse crossbar concentrator.

Theorem 3.2. *For the fat-slim QSC(n, m) let $\mathcal{X}_r = \{x_1, x_2, \ldots, x_r\}$ be any ordered r-input subset where $x_1 < x_2 < \cdots < x_r$, $\forall\, r, 1 \leq r \leq m$. Then, there exists an output matching, $\mathcal{Y}_r = \{y_1, y_2, \ldots, y_r\}$ for $\mathcal{X}_r$ obtained as result of self-routing the fat-slim QSC(n, m), it is given by*

$$y_i = \begin{cases} \pi(x_i), & x_i \in \mathcal{I}_S, \\ b_{i-(m-a)}, & x_i \in \mathcal{I}_F. \end{cases}$$

where $\mathcal{B} = \{b_1, \ldots, b_a\} = (\{\pi(x_i) \in \mathcal{I}_S\})'$ such that $b_1 < b_2 < \cdots < b_a$, $a = |\mathcal{B}|$, $i = 1, 2, \ldots, r$.

Proof. In the ordered r-input set $\mathcal{X}_r$, let the first k ($k \leq r$) inputs belong to the slim section and the rest to the fat section.

If $x_1 \in \mathcal{I}_S$, we get

$$\begin{aligned} y_1 &= \min_z\{z \in \mathcal{A}_{x_1} \cap \mathcal{Y}_0'\} \quad \text{(from Lemma 3.2)} \\ &= \min_z\{z \in \mathcal{A}_{x_1} \cap \mathcal{O}\} \quad (\text{as } \mathcal{Y}_0 = \emptyset) \\ &= \min_z\{z \in \mathcal{A}_{x_1}\} = \pi(x_1) \quad \text{(from Eqn. (3.7a))} \end{aligned} \tag{3.8}$$

Hence, $y_1 \neq \emptyset$ and $\mathcal{Y}_1 = \{\pi(x_1)\}$ is the set of matched outputs after routing on the first input.

Similarly for $x_2 \in \mathcal{I}_S$ we get

$$\begin{aligned} y_2 &= \min_z\{z \in \mathcal{A}_{x_2} \cap (\mathcal{Y}_1)'\} \quad \text{(from Lemma 3.2)} \\ &= \min_z\{z \in \{\pi(x_2)\} \cap \{y_1\}'\} \quad \text{(from Eqn. (3.7a))} \\ &= \min_z\{z \in \{\pi(x_2)\} \cap \{\pi(x_1)\}'\} \qquad (3.9) \\ &= \min_z\{z \in \{\pi(x_2)\}\} \quad (\text{as } \pi(x_1) \neq \pi(x_2)) \\ &= \pi(x_2) \qquad (3.10) \end{aligned}$$

Continuing this way we get

$$\mathcal{Y}_k = \{\pi(x_1), \pi(x_2), \ldots, \pi(x_k)\} \qquad (3.11)$$

$\mathcal{X}_r$ is an ordered set of distinct inputs, which means that all the elements of $\mathcal{Y}_k$ are distinct and hence form a matching for the k slim inputs. If $k = r$ then the proof is complete, else

For input $x_{k+1} \in \mathcal{I}_F$ we get

$$\begin{aligned} y_{k+1} &= \min_z\{z \in \mathcal{A}_{x_{k+1}} \cap (\mathcal{Y}_k)'\} \quad \text{(from Lemma 3.2)} \\ &= \min_z\{z \in \mathcal{O} \cap \{\pi(x_1), \ldots, \pi(x_k)\}'\} \quad \text{(from Eqn. (3.11))} \\ &= \min_z\{z \in \{\pi(x_1), \ldots, \pi(x_k)\}'\} \qquad (3.12) \end{aligned}$$

Note $|\{\pi(x_1), \ldots, \pi(x_k)\}'| = m - k$.

Let $\mathcal{B} = \{b_1, \ldots, b_{m-k}\}$ where $b_i \in (\mathcal{Y}_k)'$, $i = 1, \ldots, m-k$, such that $b_1 < b_2 < \cdots < b_{m-k}$. Thus, $\mathcal{B}$ is an ordered version of the set $(\mathcal{Y}_k)' = \{\pi(x_1), \ldots, \pi(x_k)\}'$

with elements arranged in increasing order of magnitude, $|\mathcal{B}| = m - k$. From Eqn. (3.12),

$$y_{k+1} = \min_z\{z \in \mathcal{B}\} = b_1$$

Also, $k \leq r \leq m$ and if $k = r = m$, i.e., all the slim inputs are concentrated then $y_{k+1} = \emptyset$ and $\mathcal{Y}_k$ is the matching corresponding to the concentration. If $k < r$ then clearly $y_{k+1} = b_1 \neq \emptyset$ as then $\mathcal{B} \neq \emptyset$. Thus $\mathcal{Y}_{k+1} = \mathcal{Y}_k \cup \{b_1\} = \{\pi(x_1), \ldots, \pi(x_k), b_1\}$. For input $x_{k+2} \in \mathcal{I}_F$ we get

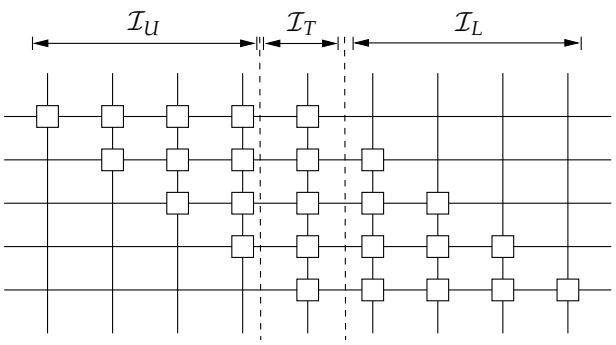
$$\begin{aligned}
y_{k+2} &= \min_z\{z \in \mathcal{A}_{x_{k+2}} \cap (\mathcal{Y}_{k+1})'\} \\
&= \min_z\{z \in \mathcal{O} \cap (\mathcal{Y}_k \cup \{b_1\})'\} \\
&= \min_z\{z \in (\mathcal{Y}_k \cup \{b_1\})'\} \\
&= \min_z\{z \in (\mathcal{Y}_k)' \cap \{b_1\}'\} \\
&= \min_z\{z \in \mathcal{B} \cap \{b_1\}'\} \\
&= \min_z\{z \in \mathcal{B} \setminus \{b_1\}\} \\
&= b_2
\end{aligned}$$

Continuing in the same fashion we get $\mathcal{Y}_{k+1}^r = \{b_1, \ldots, b_{r-k}\}$. Now, $|\mathcal{B}| = m - k$, i.e., $k = m - |\mathcal{B}|$. Thus

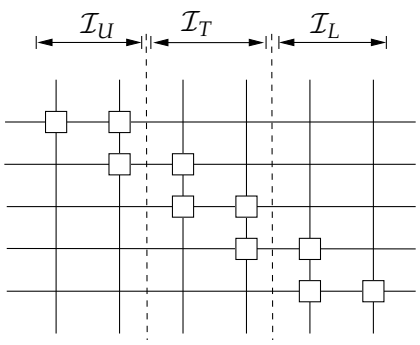
$$\begin{aligned}
y_{k+i} &= b_i = b_{(k+i)-k} \\
&= b_{k+i-(m-|\mathcal{B}|)}, \quad i = 1, \ldots, r-k
\end{aligned}$$

hence $y_i = b_{i-(m-a)}$, $i = k+1, \ldots, r$, $a = |\mathcal{B}|$.

Therefore, by Theorem 3.1 the fat-slim QSC(n, m) is self routing. □



(a) Partitions of banded crossbar, $n = 9, m = 5$, ($n \geq 2m - 1$).



(b) Partitions of banded crossbar, $n = 6, m = 5$, ($m \leq n < 2m - 1$).

Figure 3.8: Partitions of banded sparse crossbar concentrator.

3.6.3 Self-Routing Banded QSC(n, m)

We show that for banded sparse crossbar concentrators our self-routing scheme can find an r-output matching for any r input subset ($r \leq m$).

Definition 3.4 (Banded Crossbar)**.** An (n, m)-sparse crossbar is called banded if the set of inputs, $\mathcal{I}$, can be partitioned into three sets $\mathcal{I}_U, \mathcal{I}_T$ and $\mathcal{I}_L$ as shown in Figure 3.8 with the corresponding neighbor sets for the inputs as follows:

If $n \geq 2m - 1$ then:

$$i \in \mathcal{I}_U \Leftrightarrow 1 \leq i \leq m - 1;\ \mathcal{A}_i = \{1, 2, \ldots, i\} \tag{3.13a}$$

$$i \in \mathcal{I}_T \Leftrightarrow m \leq i \leq n - m + 1;\ \mathcal{A}_i = \{1, 2, \ldots, m\} = \mathcal{O} \tag{3.13b}$$

$$i \in \mathcal{I}_L \Leftrightarrow n - m + 2 \leq i \leq n;\ \mathcal{A}_i = \{i - n + m, \ldots, m\} \tag{3.13c}$$

If $m \leq n < 2m - 1$ then:

$$i \in \mathcal{I}_U \Leftrightarrow 1 \leq i \leq n - m + 1;\ \mathcal{A}_i = \{1, 2, \ldots i\} \tag{3.14a}$$

$$i \in \mathcal{I}_T \Leftrightarrow n - m + 2 \leq i \leq m - 1;\ \mathcal{A}_i = \{i - n + m, \ldots, i\} \tag{3.14b}$$

$$i \in \mathcal{I}_L \Leftrightarrow m \leq i \leq n;\ \mathcal{A}_i = \{i - n + m, \ldots, m\} \tag{3.14c}$$

Note that for $n = 2m - 2$, $n - m + 2 = m > m - 1$. Hence in this case, from Eqn. (3.14b) $\mathcal{I}_T$ does not exist, but this does not affect the proof below. Also, equations Eqn. (3.13a)-(3.14c) can be written more succinctly as:

$$\mathcal{A}_i = \{\max(1,\ i - n + m), \ldots, \min(i,\ m)\}, \quad i = 1, \ldots, n \tag{3.15}$$

Every banded (n, m)-sparse crossbar is an optimal (n, m)-concentrator with $m(n - m + 1)$ crosspoints [25].

Theorem 3.3. *For the banded QSC(n,m) let $\mathcal{X}_r = \{x_1, x_2, \ldots, x_r\}$ be any ordered r-input subset where $x_1 < x_2 < \cdots < x_r$, $\forall\ r, 1 \leq r \leq m$. Then, there exists an output matching, $\mathcal{Y}_r = \{y_1, y_2, \ldots, y_r\}$ for $\mathcal{X}_r$ obtained as result of self-routing the banded QSC(n, m), it is given by*

if $n \geq 2m - 1$

$$y_i = \begin{cases} i, & x_i \in \mathcal{I}_U \cup \mathcal{I}_T, \\ \max(i,\ x_i - n + m), & x_i \in \mathcal{I}_L. \end{cases}$$

if $m \leq n < 2m - 1$

$$y_i = \begin{cases} i, & x_i \in \mathcal{I}_U, \\ \max(i,\ x_i - n + m), & x_i \in \mathcal{I}_T \cup \mathcal{I}_L. \end{cases}$$

where $i = 1, 2, \ldots, r$.

Proof. In $\mathcal{X}_r$ let k_1 inputs belong to $\mathcal{I}_U$, k_2 inputs belong to $\mathcal{I}_T$ and the rest $r - (k_1 + k_2)$ inputs belong to $\mathcal{I}_L$, i.e., $\{x_1, \ldots, x_{k_1}\} \subseteq \mathcal{I}_U$, $\{x_{k_1+1}, \ldots, x_{k_1+k_2}\} \subseteq \mathcal{I}_T$ and $\{x_{k_1+k_2+1}, \ldots, x_r\} \subseteq \mathcal{I}_U$.

Case I: $n \geq 2m-1$

For input $x_1 \in \mathcal{I}_U$:

$$\begin{aligned} y_1 &= \min_z\{z \in \mathcal{A}_{x_1} \cap \mathcal{Y}_0'\} \quad \text{(from Lemma 3.2)} \\ &= \min_z\{z \in \mathcal{A}_{x_1} \cap \mathcal{O}\} \quad (\text{as } \mathcal{Y}_0 = \emptyset) \\ &= \min_z\{z \in \mathcal{A}_{x_1}\} \\ &= \min_z\{z \in \{1, 2, \ldots, x_1\}\} \quad \text{(from Eqn. (3.13a))} \\ &= 1 \end{aligned}$$

Similarly for $x_2 \in \mathcal{I}_U$

$$\begin{aligned} y_2 &= \min_z\{z \in \mathcal{A}_{x_2} \cap (\mathcal{Y}_1)'\} \\ &= \min_z\{z \in \{1, \ldots, x_2\} \cap \{1\}'\} \\ &= \min_z\{z \in \{2, \ldots, x_2\}\} = 2 \end{aligned}$$

Proceeding this way for all inputs in $\mathcal{I}_U$ we get $\mathcal{Y}_{k_1} = \{1, 2, \ldots, k_1\}$.

For input $x_{k_1+1} \in \mathcal{I}_T$:

$$\begin{aligned} y_{k_1+1} &= \min_z\{z \in \mathcal{A}_{x_{k_1+1}} \cap (\mathcal{Y}_{k_1})'\} \quad \text{(from Lemma 3.2)} \\ &= \min_z\{z \in \{1, \ldots, m\} \cap \{1, \ldots, k_1\}'\} \quad \text{(from Eqn. (3.13b))} \\ &= \min_z\{z \in \{k_1+1, \ldots, m\}\} = k_1 + 1 \end{aligned}$$

We can get similar results for all other inputs in $\mathcal{I}_T$. Thus, $\mathcal{Y}_{k_1+k_2} = \{1, \ldots, k_1+k_2\}$.

For input $x_{k_1+k_2+1} \in \mathcal{I}_L$:

$$y_{k_1+k_2+1} = \min_z\{z \in \mathcal{A}_{x_{k_1+k_2+1}} \cap (\mathcal{Y}_{k_1+k_2})'\}$$

$$= \min_z\{z \in \{x_{k_1+k_2+1} - n + m, \ldots, m\} \cap \{1, \ldots, k_1 + k_2\}'\} \tag{3.16}$$

$$= \min_z\{z \in \{x_{k_1+k_2+1} - n + m, \ldots, m\} \cap \{k_1 + k_2 + 1, \ldots, m\}\}$$

$$= \min_z\{z \in \{\max(x_{k_1+k_2+1} - n + m,\ k_1 + k_2 + 1), \ldots, m\}\} \tag{3.17}$$

$$= \max(k_1 + k_2 + 1,\ x_{k_1+k_2+1} - n + m) \tag{3.18}$$

where Eqn. (3.16) follows from Eqn. (3.13c) and Eqn. (3.17) follows from the fact that we are taking the intersection of 2 sets both of which cover continuous intervals of outputs up to output m. Eqn. (3.18) is obvious as the lower boundary of the interval is smaller than the rest of the numbers in the interval. We now use induction to prove the rest of the theorem.

Induction assumption:

$$y_i = \max(i,\ x_i - n + m), \quad \text{for all } i = k_1 + k_2 + 1, \ldots, j - 1 \text{ where } j \leq r. \tag{3.19}$$

Need to prove: $y_j = \max(j,\ x_j - n + m)$

Proof for Induction: We have already proved the base case for $i = k_1 + k_2 + 1$. We will first show that, for y_i's chosen according to Eqn. (3.19), $y_{i-1} < y_i$.

Note $y_{i-1} = \max(i - 1, x_{i-1} - n + m)$ and $y_i = \max(i, x_i - n + m)$, thus we get the following cases:

1. $y_{i-1} = i - 1$: We get the followings series of inequalities:

$$\max(i, x_i - n + m) \geq i$$

$$\Rightarrow y_i \geq i \quad (\text{as } y_i = \max(i, x_i - n + m))$$

$$> i - 1 = y_{i-1}$$

2. $y_{i-1} = x_{i-1} - n + m$: We get the followings series of inequalities:

$$\begin{aligned} \max(x_i - n + m, i) &\geq x_i - n + m \\ \Rightarrow y_i &\geq x_i - n + m \quad (\text{as } y_i = \max(i, x_i - n + m)) \\ &> x_{i-1} - n + m = y_{i-1} \end{aligned}$$

Thus $y_i > y_{i-1},\ i = k_1 + k_2 + 2, \ldots, j-1$.

We know that $\mathcal{Y}_{k_1+k_2} = \{1, \ldots, k_1 + k_2\}$ is monotonically increasing, $\mathcal{Y}_{k_1+k_2+1}^{j-1}$ is monotonically increasing and

$$y_{k_1+k_2+1} = \max(k_1 + k_2 + 1, x_{k_1+k_2+1} - n + m)) > k_1 + k_2 = y_{k_1+k_2}$$

Thus $\mathcal{Y}_{j-1} = \mathcal{Y}_{k_1+k_2} \cup \mathcal{Y}_{k_1+k_2+1}^{j-1}$ is monotonically increasing, i.e., $y_1 < y_2 < \cdots < y_{j-1}$.

By Lemma 3.2

$$y_j = \min_z \{z \in \mathcal{A}_{x_j} \cap (\mathcal{Y}_{j-1})'\} \tag{3.20}$$

By induction assumption $y_{j-1} = \max(j-1, x_{j-1} - n + m)$.

Case 1: $y_{j-1} = x_{j-1} - n + m$

By monotonicity of $\mathcal{Y}_{j-1}$ we get $\max_z(z \in \mathcal{Y}_{j-1}) = x_{j-1} - n + m$. Thus

$$(\mathcal{Y}_{j-1})' = \mathcal{Z} \cup \{x_{j-1} - n + m + 1, \ldots, m\}$$

$$\text{where } \mathcal{Z} \subseteq \{k_1 + k_2 + 1, \ldots, x_{j-1} - n + m - 1\}$$

Therefore,

$$\begin{aligned}
\mathcal{A}_{x_j} \cap (\mathcal{Y}_{j-1})' &= \{x_j - n + m, \ldots, m\} \cap \left[\mathcal{Z} \cup \{x_{j-1} - n + m + 1, \ldots, m\}\right] \\
&= \emptyset \cup \{x_j - n + m, \ldots, m\} \cap \{x_{j-1} - n + m + 1, \ldots, m\} \\
&= \{(\max(x_j - n + m,\ x_{j-1} - n + m + 1)), \ldots, m\} \\
&= \{x_j - n + m, \ldots, m\} \quad (\text{as } x_j \geq x_{j-1} + 1) \qquad (3.21)
\end{aligned}$$

Substituting in Eqn. (3.20) we get

$$\begin{aligned}
y_j &= \min_z \{z \in \{x_j - n + m, \ldots, m\}\} \\
&= x_j - n + m \qquad (3.22)
\end{aligned}$$

Now $y_{j-1} = \max(x_{j-1} - n + m,\ j - 1) = x_{j-1} - n + m$. Thus

$$\begin{aligned}
& x_{j-1} - n + m \geq j - 1 \\
\Rightarrow\ & x_{j-1} - n + m + 1 \geq j \\
\Rightarrow\ & x_j - n + m \geq j \quad (\text{as } x_j \geq x_{j-1} + 1) \qquad (3.23)
\end{aligned}$$

From Eqns. (3.22) and (3.23)

$$y_j = \max(x_j - n + m,\ j) \qquad (3.24)$$

Case 2: $y_{j-1} = j - 1$

Since $\mathcal{Y}_{j-1}$ is monotonically increasing, i.e., $y_1 < y_2 < \cdots < y_{j-1}$ and $y_{j-1} = j - 1$

$$\begin{aligned}
& \Rightarrow \mathcal{Y}_{j-1} = \{1, 2, \ldots, j - 1\} \\
& \Rightarrow (\mathcal{Y}_{j-1})' = \{j, \ldots, m\}
\end{aligned}$$

Substituting in Eqn. (3.20) we get

$$\begin{aligned} y_j &= \min_z\{\{x_j - n + m, \ldots, m\} \cap \{j, \ldots, m\}\} \\ &= \min_z\{\max(x_j - n + m,\ j), \ldots, m\} \\ &= \max(x_j - n + m,\ j) \end{aligned} \tag{3.25}$$

From Eqn. (3.24) and Eqn. (3.25) $y_j = \max(x_j - n + m,\ j)$ and proof for the induction is complete.

Case II: $m \leq n < 2m - 1$

k_1 inputs in S belong to $\mathcal{I}_U$, i.e., $\{x_1, \ldots, x_{k_1}\} \subseteq \mathcal{I}_U$.

For $y_1, \ldots, y_{k_1}$ the proof is exactly the same as for the case $n \geq 2m - 1$ and we get $\mathcal{Y}_{k_1} = \{1, 2, \ldots, k_1\}$.

The next k_2 inputs in $\mathcal{X}_r$ belong to $\mathcal{I}_T$, i.e., $\{x_{k_1+1}, \ldots, x_{k_1+k_2}\} \subseteq \mathcal{I}_T$.

For y_{k_1+1} we get

$$\begin{aligned} y_{k_1+1} &= \min_z\{z \in \mathcal{A}_{x_{k_1+1}} \cap (\mathcal{Y}_{k_1})'\} \quad \text{(from Lemma 3.2)} \\ &= \min_z\{z \in \{x_{k_1+1} - n + m, \ldots, x_{k_1+1}\} \cap \{1, \ldots, k_1\}'\} \text{(from Eqn. (3.14b))} \\ &= \min_z\{z \in \{x_{k_1+1} - n + m, \ldots, x_{k_1+1}\} \cap \{k_1 + 1, \ldots, m\}\} \end{aligned}$$

Both the sets in the intersection contain a continuous series of outputs. Now obviously, $x_{k_1+1} \leq m$. Also, $k_1 + 1 \leq x_{k_1+1}$ as

$$\begin{aligned} & k_1 \leq n - m + 1 \quad \text{from Eqn. (3.14a)} \\ \Rightarrow\ & k_1 + 1 \leq n - m + 2 \\ & \leq x_{k_1+1} \quad \text{(from Eqn. (3.14b))} \end{aligned}$$

Thus

$$y_{k_1+1} = \min_z\{z \in \{\max(k_1+1, x_{k_1+1} - n + m), \ldots, x_{k_1+1}\}\}$$
$$= \max(k_1+1, x_{k_1+1} - n + m)$$

We prove the output matches for the rest of the inputs in $\mathcal{I}_T$ using induction.

Induction Assumption:

$$y_i = \max(i, x_i - n + m), \;\; k_1 + 1 \leq i \leq j-1 \text{ where } j \leq k_1 + k_2 \tag{3.26}$$

Need to prove: $y_j = \max(j,\; x_j - n + m)$.

Proof: We have already proved the base case for $i = k_1 + 1$. Similar to the case for $n \geq 2m - 1$ we can show $y_{k_1+1} < \cdots < y_{j-1}$ as below:

By induction assumption $y_i = \max(i, x_i - n + m)$ and $y_{i-1} = \max(i-1, x_{i-1} - n + m)$, $k_1 + 1 < i < j$. Comparing y_{i-1} and y_i we get the following cases

1. $y_{i-1} = i - 1$: We get the followings series of inequalities:

$$\max(i, x_i - n + m) \geq i$$
$$\Rightarrow y_i \geq i \quad (\text{as } y_i = \max(i, x_i - n + m))$$
$$> i - 1 = y_{i-1}$$

2. $y_{i-1} = x_{i-1} - n + m$: We get the followings series of inequalities:

$$\max(x_i - n + m, i) \geq x_i - n + m$$
$$\Rightarrow y_i \geq x_i - n + m \quad (\text{as } y_i = \max(i, x_i - n + m))$$
$$> x_{i-1} - n + m = y_{i-1}$$

Hence, $y_{k_1+1} < \cdots < y_{j-1}$. Also, $\mathcal{Y}_{k_1} = \{1, \ldots, k_1\}$ and $y_{k_1+1} = \max(k_1 + 1,\ x_{k_1+1} - n + m) > k_1$, hence $y_1 < y_2 < \cdots < y_{j-1}$.

By Lemma 3.2

$$y_j = \min_z \{z \in \mathcal{A}_{x_j} \cap (\mathcal{Y}_{j-1})'\} \tag{3.27}$$

By induction assumption $y_{j-1} = \max(j-1, x_{j-1} - n + m)$.

Case 1: $y_{j-1} = x_{j-1} - n + m$

Monotonicity of $\mathcal{Y}_{j-1}$ implies $\max_z(z \in \mathcal{Y}_{j-1}) = x_{j-1} - n + m$. Thus $(\mathcal{Y}_{j-1})' = \mathcal{Z} \cup \{x_{j-1} - n + m + 1, \ldots, m\}$ where $\mathcal{Z} \subseteq \{k_1 + 1, \ldots, x_{j-1} - n + m - 1\}$.

Then

$$\begin{aligned}
\mathcal{A}_{x_j} \cap (\mathcal{Y}_{j-1})' &= \{x_j - n + m, \ldots, x_j\} \cap \left[\mathcal{Z} \cup \{x_{j-1} - n + m + 1, \ldots, m\}\right] \\
&= \emptyset \cup \{x_j - n + m, \ldots, x_j\} \cap \{x_{j-1} - n + m + 1, \ldots, m\} \\
&= \{\max(x_j - n + m,\ x_{j-1} - n + m + 1), \ldots, x_j\} \\
&= \{x_j - n + m, \ldots, x_j\} \quad (\text{as } x_j \geq x_{j-1} + 1)
\end{aligned} \tag{3.28}$$

Substituting in Eqn. (3.27) we get

$$\begin{aligned}
y_j &= \min_z \{z \in \{x_j - n + m, \ldots, x_j\}\} \\
&= x_j - n + m
\end{aligned} \tag{3.29}$$

Now $y_{j-1} = \max(x_{j-1} - n + m,\ j - 1) = x_{j-1} - n + m$. Thus

$$\begin{aligned}
& x_{j-1} - n + m \geq j - 1 \\
\Rightarrow\ & x_{j-1} - n + m + 1 \geq j \\
\Rightarrow\ & x_j - n + m \geq j \quad (\text{as } x_j \geq x_{j-1} + 1)
\end{aligned} \tag{3.30}$$

From Eqn. (3.29) and Eqn. (3.30)

$$y_j = \max(x_j - n + m,\ j) \tag{3.31}$$

Case 2: $y_{j-1} = j - 1$

As $y_1 < y_2 < \cdots < y_{j-1}$ and $y_{j-1} = j - 1$

$$\Rightarrow \mathcal{Y}_{j-1} = \{1, 2, \ldots, j-1\}$$

$$\Rightarrow (\mathcal{Y}_{j-1})' = \{j, \ldots, m\}$$

Substituting in Eqn. (3.27) we get

$$\begin{aligned} y_j &= \min_z\{\{x_j - n + m, \ldots, x_j\} \cap \{j, \ldots, m\}\} \\ &= \min_z\{\max(x_j - n + m,\ j), \ldots, x_j\} \\ &= \max(x_j - n + m,\ j) \end{aligned} \tag{3.32}$$

From Eqn. (3.31) and Eqn. (3.32) $y_j = \max(x_j - n + m,\ j)$ and proof for the induction is complete.

For input $x_{k_1+k_2+1}$ we get

$$\begin{aligned} y_{k_1+k_2+1} &= \min_z\{z \in \mathcal{A}_{x_{k_1+k_2+1}} \cap (\mathcal{Y}_{k_1+k_2})'\} \quad \text{(from Lemma 3.2)} \\ &= \min_z\{z \in \{x_{k_1+k_2+1} - n + m, \ldots, m\} \cap (\mathcal{Y}_{k_1+k_2})'\} \end{aligned} \tag{3.33}$$

As $y_{k_1+k_2} = \max(k_1 + k_2,\ x_{k_1+k_2} - n + m)$, we get the following two cases:

Case 1: $y_{k_1+k_2} = k_1 + k_2$. Since $y_1 < y_2 < \cdots < y_{k_1+k_2}$, and $y_{k_1+k_2} = k_1 + k_2$

$$\mathcal{Y}_{k_1+k_2} = \{1, 2, \ldots, k_1 + k_2\}$$

$$\Rightarrow (\mathcal{Y}_{k_1+k_2})' = \{k_1 + k_2 + 1, \ldots, m\} \tag{3.34}$$

Substituting Eqn. (3.34) in Eqn. (3.33) we get

$$\begin{aligned} y_{k_1+k_2+1} &= \min_z\{z \in \{x_{k_1+k_2+1} - n + m, \ldots, m\} \cap \{k_1 + k_2 + 1, \ldots, m\}\} \\ &= \min_z\{z \in \{\max(x_{k_1+k_2+1} - n + m,\ k_1 + k_2 + 1), \ldots, m\}\} \\ &= \max(x_{k_1+k_2+1} - n + m,\ k_1 + k_2 + 1) \end{aligned} \tag{3.35}$$

Case 2: $y_{k_1+k_2} = x_{k_1+k_2} - n + m$. Since $y_1 < \cdots < y_{k_1+k_2}$, $\max_z(z \in \mathcal{Y}_{k_1+k_2}) = y_{k_1+k_2} = x_{k_1+k_2} - n + m$. Thus

$$(\mathcal{Y}_{k_1+k_2})' = \mathcal{Z} \cup \{x_{k_1+k_2} - n + m + 1, \ldots, m\}$$

where $\mathcal{Z} \subseteq \{k_1 + k_2 + 1, \ldots, x_{k_1+k_2} - n + m - 1\}$. Substituting this in Eqn. (3.35) we get

$$\begin{aligned} y_{k_1+k_2+1} &= \min_z\{z \in \{x_{k_1+k_2+1} - n + m, \ldots, m\} \cap \\ &\qquad (\mathcal{Z} \cup \{x_{k_1+k_2} - n + m + 1, \ldots, m\})\} \\ &= \min_z\{z \in \emptyset \cup (\{x_{k_1+k_2+1} - n + m, \ldots, m\} \cap \\ &\qquad \{x_{k_1+k_2} - n + m + 1, \ldots, m\})\} \\ &= \min_z\{z \in \{\max(x_{k_1+k_2} - n + m + 1,\ x_{k_1+k_2+1} - n + m), \ldots, m\}\} \\ &= \max(x_{k_1+k_2} - n + m + 1,\ x_{k_1+k_2+1} - n + m) \\ &= x_{k_1+k_2+1} - n + m \quad (\text{as } x_{k_1+k_2+1} \geq x_{k_1+k_2} + 1) \end{aligned} \tag{3.36}$$

Also,

$$\begin{aligned} x_{k_1+k_2+1} - n + m &\geq x_{k_1+k_2} + 1 - n + m \\ &= y_{k_1+k_2} + 1 \\ &\geq k_1 + k_2 + 1 \end{aligned} \tag{3.37}$$

where Eqn. (3.37) follows from the fact that $y_{k_1+k_2} = \max(x_{k_1+k_2} - n + m,\ k_1 + k_2)$. From Eqn. (3.36) and Eqn. (3.37)

$$y_{k_1+k_2+1} = \max(k_1 + k_2 + 1,\ x_{k_1+k_2+1} - n + m)$$

We can now use an induction argument for rest of the inputs in $\mathcal{I}_L$ similar to the case for $n \geq 2m - 1$ to show that

$$\begin{aligned} y_i &= \max(i,\ x_i - n + m), \quad k_1 + k_2 + 1 \leq i \leq r \\ \Rightarrow y_i &= \max(i,\ x_i - n + m), \quad x_i \in \mathcal{I}_L. \end{aligned}$$

Thus, by Theorem 3.1 the banded QSC(n, m) is self-routing. □

3.7 An Example

We give an example to illustrate self-routing on a fat-slim QSC$(5, 3)$. This concentrator is shown in Figure 3.9. The quantum packets present at inputs $1, 3$ and 4 are $|Q_1\rangle = \frac{1}{\sqrt{2}}(|1, d_{11}\rangle + |1, d_{12}\rangle)$, $|Q_3\rangle = |1, d_3\rangle$ and $|Q_4\rangle = \frac{\sqrt{3}}{2}|1, d_{41}\rangle + \frac{1}{2}|1, d_{42}\rangle$ respectively. Inputs 2 and 5 do not have any packets. Inputs $6, 7$ and 8 correspond to the three dummy inputs on the left hand side from top to bottom. Thus, in this case, the input quantum concentration assignment is given by

$$\begin{aligned} &|Q_1\rangle \otimes |0, d_2\rangle \otimes |Q_3\rangle \otimes |Q_4\rangle \otimes |0, d_5\rangle \bigotimes_{i=6}^{8} |0, d_i\rangle \\ &= \left(\frac{1}{\sqrt{2}}|1, d_{11}\rangle + \frac{1}{\sqrt{2}}|1, d_{12}\rangle\right) \otimes |0, d_2\rangle \otimes |1, d_3\rangle \otimes \left(\frac{\sqrt{3}}{2}|1, d_{41}\rangle + \frac{1}{2}|1, d_{42}\rangle\right) \\ &\qquad \otimes |0, d_5\rangle \otimes |0, d_6\rangle \otimes |0, d_7\rangle \otimes |0, d_8\rangle \end{aligned}$$

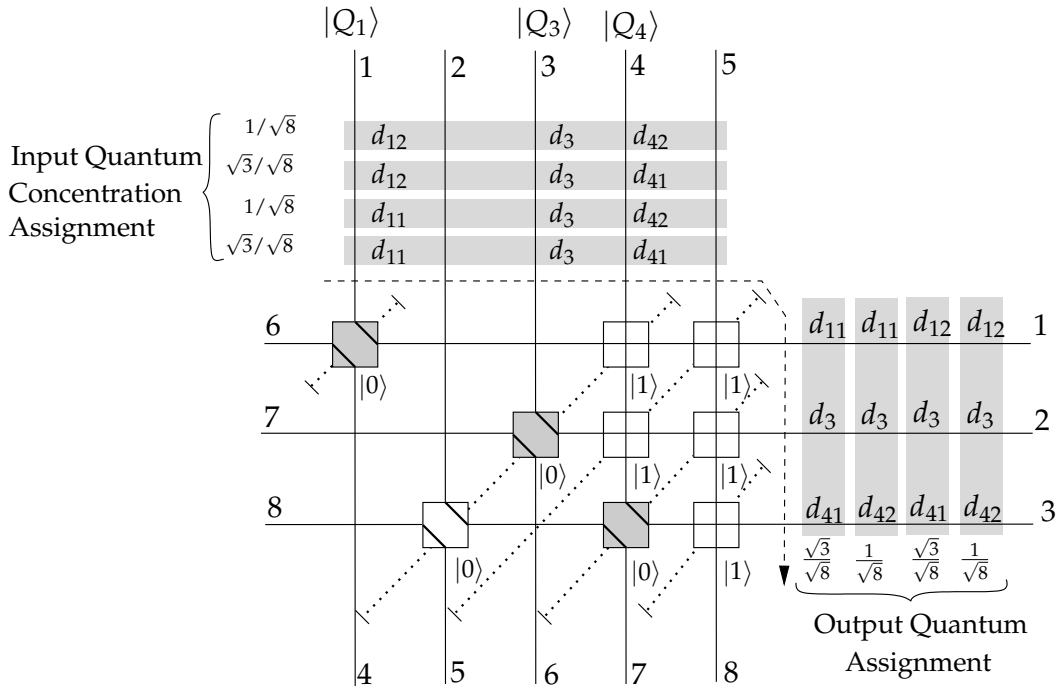


Figure 3.9: Self-Routing on fat-slim QSC$(5,3)$.

$$
\begin{aligned}
&= \frac{\sqrt{3}}{2\sqrt{2}} \left|(1,d_{11}),(0,d_2),(1,d_3),(1,d_{41}),(0,d_5),(0,d_6),(0,d_7),(0,d_8)\right\rangle \\
&\quad + \frac{1}{2\sqrt{2}} \left|(1,d_{11}),(0,d_2),(1,d_3),(1,d_{42}),(0,d_5),(0,d_6),(0,d_7),(0,d_8)\right\rangle \\
&\quad + \frac{\sqrt{3}}{2\sqrt{2}} \left|(1,d_{12}),(0,d_2),(1,d_3),(1,d_{41}),(0,d_5),(0,d_6),(0,d_7),(0,d_8)\right\rangle \\
&\quad + \frac{1}{2\sqrt{2}} \left|(1,d_{12}),(0,d_2),(1,d_3),(1,d_{42}),(0,d_5),(0,d_6),(0,d_7),(0,d_8)\right\rangle
\end{aligned}
\tag{3.38}
$$

Thus, the input is a superposition of four concentration patterns with coefficients $\sqrt{3}/\sqrt{8}$, $1/\sqrt{8}$, $\sqrt{3}/\sqrt{8}$ and $1/\sqrt{8}$ respectively, shown by grey horizontal bars. Since all four patterns are capacity achieving, the quantum assignment is also capacity achieving. The state of the crosspoints is also shown. The shaded crosspoints route the valid packets on inputs $1,3$ and 4. Measurement at the output will result in one out of the four patterns shown at the output being observed with

probabilities $3/8, 1/8, 3/8$ and $1/8$ respectively. Therefore, data packets d_{11} and d_{12} are observed on output 1 with probability $1/2$. Data packet d_3 is observed on output 2 with probability 1 and data packets d_{41} and d_{42} are observed with probability $3/4$ and $1/4$ respectively on output 3. This output state can be explicitly written as:

$$\begin{aligned}
&\frac{\sqrt{3}}{2\sqrt{2}}\,|(1,d_{11}),(1,d_3),(1,d_{41}),(0,d_6),(0,d_8),(0,d_7),(0,d_2),(0,d_5)\rangle \\
&\quad+\frac{1}{2\sqrt{2}}\,|(1,d_{11}),(1,d_3),(1,d_{42}),(0,d_6),(0,d_8),(0,d_7),(0,d_2),(0,d_5)\rangle \\
&\quad+\frac{\sqrt{3}}{2\sqrt{2}}\,|(1,d_{12}),(1,d_3),(1,d_{41}),(0,d_6),(0,d_8),(0,d_7),(0,d_2),(0,d_5)\rangle \\
&\quad+\frac{1}{2\sqrt{2}}\,|(1,d_{12}),(1,d_3),(1,d_{42}),(0,d_6),(0,d_8),(0,d_7),(0,d_2),(0,d_5)\rangle
\end{aligned} \tag{3.39}$$

The packets in the concentration patterns are written in increasing order of outputs, with outputs 1–3 on the right and outputs 4–8 located on the bottom. The dashed arrow shows the order in which the crosspoints are traversed during routing, and the crosspoints in one stage are indicated by the dotted diagonals. The initial state of the auxiliary qubits (control qubits) is $|000000000\rangle$ which is a string of nine zeros, each corresponding to one crosspoint in the crossbar. Recall that control qubits are set to 1 for the cross state and to 0 for the through state. The output state of the auxiliary qubits is indicated beside the crosspoints. Thus, the output state of the auxiliary qubits is $|000110111\rangle$ where the bits are written in order from top to bottom and left to right, e.g., the third crosspoint for input 4 is set to a through state and this is the sixth crosspoint in traversal order, so the sixth bit in the output state is 0.

3.8 Output for Capacity Exceeding Input Patterns

So far we have shown that in a fat-slim or banded QSC(n, m), self-routing can be used to concentrate any capacity achieving input assignment pattern. We now present the case when the input pattern exceeds the capacity of the crossbar.

For a self-routing QSC(n, m), consider a capacity exceeding input concentration pattern with r, $(r > m)$ valid packets. The ordered set of inputs with packets to concentrate is $\mathcal{X}_r = \{x_1, \ldots, x_m, x_{m+1}, \ldots, x_r\} = \mathcal{X}_m \cup \mathcal{X}_{m+1}^r$ where $\mathcal{X}_m = \{x_1, \ldots, x_m\}$ and $\mathcal{X}_{m+1}^r = \{x_{m+1}, \ldots, x_r\}$. Since inputs are routed in increasing order, all inputs in $\mathcal{X}_m$ are concentrated to the m outputs, i.e., $\mathcal{Y}_m = \{y_1, \ldots, y_m\} = \mathcal{O}$. For input x_{m+1}: $y_{m+1} = \min_z\{z \in \mathcal{A}_{x_{m+1}} \cap (\mathcal{Y}_m)'\} = \min_z\{z \in \mathcal{A}_{x_{m+1}} \cap \emptyset\} = \emptyset$. Similarly for the other inputs in $\mathcal{X}_{m+1}^r$, the matching output is $\emptyset$, i.e., $\{y_{m+1}, \ldots, y_r\} = \emptyset$. If $y_i = \emptyset$, then all crosspoints in the column for the corresponding input x_i are set to cross state and the packet comes out on the bottom, which is at output $x_i + m$. Hence, the m lowest numbered inputs are concentrated and the rest are connected to corresponding output at the bottom.

3.9 Restoring Auxiliary Control Quantum Bits

Quantum information can be encoded in many different ways, such as the spin component of basic particles like electrons or protons, or in the polarization of photons. But, such particles can interact with the environment which leads to a corruption of their quantum state, a process known as *decoherence*. Decoherence can be viewed as a measurement of a superposed quantum state which collapses

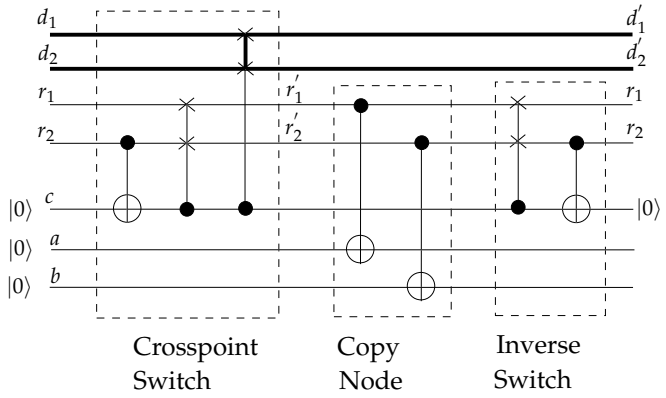


Figure 3.10: Circuit for restoring the control quantum bit c.

it to one of its basis states. This leads to a loss of information, but for a quantum circuit, this information loss can be overcome if the ancillary quantum bits used as control qubits are restored back to their original states, so that a corruption of their state does not affect the observed quantum data. We now give a method to restore the state of the auxiliary bits back to their original state, i.e., $|0\rangle$. For a single quantum crosspoint we can restore the control quantum bit back to the state $|0\rangle$ as shown in Figure 3.10. The mapping performed is:

$$|(r_1, d_1), (0, d_2)\rangle\, |0\rangle_c\, |00\rangle_{ab} \xrightarrow[Through]{Crosspoint} |(r_1, d_1), (0, d_2)\rangle\, |0\rangle_c\, |00\rangle_{ab} \tag{3.40}$$

$$\xrightarrow{Copy} |(r_1, d_1), (0, d_2)\rangle\, |0\rangle_c\, |r_1 0\rangle_{ab} \tag{3.41}$$

$$\xrightarrow{Inverse} |(r_1, d_1), (0, d_2)\rangle\, |0\rangle_c\, |r_1 0\rangle_{ab} \tag{3.42}$$

$$|(r_1, d_1), (1, d_2)\rangle\, |0\rangle_c\, |00\rangle_{ab} \xrightarrow[Cross]{Crosspoint} |(1, d_2), (r_1, d_1)\rangle\, |1\rangle_c\, |00\rangle_{ab} \tag{3.43}$$

$$\xrightarrow{Copy} |(1, d_2), (r_1, d_1)\rangle\, |1\rangle_c\, |1\, r_1\rangle_{ab} \tag{3.44}$$

$$\xrightarrow{Inverse} |(r_1, d_2), (1, d_1)\rangle\, |0\rangle_c\, |1\, r_1\rangle_{ab} \tag{3.45}$$

At the output of the quantum crosspoint the two CNOT gates in the copy circuit

copy the values of bits r_1 and r_2 onto a and b respectively. This can be seen in Eqn. (3.41) and Eqn. (3.44). The inverse switch then does a controlled swap of the two routing bits r_1 and r_2 before restoring c back to its original state as can be seen in Eqn. (3.42) and Eqn. (3.45). Note that when the bit c is 0 at the output of the quantum crosspoint switch then the restoring portion does not modify anything (Eqns. (3.40)-(3.42)) as the corresponding auxiliary qubit is already in the state $|0\rangle$.

On measurement at the output we determine valid (not valid) packets by observing their associated routing bit as 1 (0). But note that on final measurement in Figure 3.10, the routing bits may not correspond to the data part of their packets, this is seen in Eqn. (3.45) where r_1, d_2 and $1, d_1$ are together instead of being $1, d_2$ and r_1, d_1. But the copying operation ensures that we have a copy of the correct routing bits and can use these to distinguish the valid packets, for example, in Eqn. (3.45) the correct values of the routing bits at the output are 1 for the upper packet and r_1 for the lower packet and these are present in the correct order, $1, r_1$, on qubits a and b. Thus we can now consider a as the routing qubit for the packet at the upper output and b as the routing qubit for the packet at the lower output.

This circuit restores the control qubit for a single crosspoint. For the entire self-routing QSC we need a mirror image of the sparse crossbar concentrator concatenated with the QSC after the copying of the routing bits is done at the output to restore the control qubits. This is shown in Figure 3.11. Only the routing qubits are involved in restoring the state of the control qubits, hence only these qubits are forwarded to the next stage after the QSC and are shown by dashed lines. The dotted lines show the order of traversal of crosspoints and inverse switches.

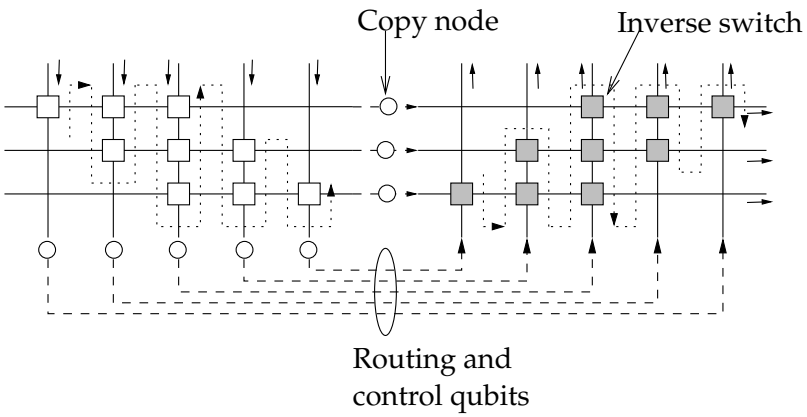


Figure 3.11: A Banded QSC(5,3) with additions for restoring the auxiliary control quantum bits.

3.10 Cost Analysis

We now give the gate count and bounds on the delay for a self-routing QSC(n, m).

3.10.1 Gate Count

We need, per crosspoint, one multi-qubit switch gate for swapping the $n_d + 1$ bit packets and one CNOT gate for setting the auxiliary control bit. A switch gate for swapping one quantum bit packets can be implemented using two CNOT gates and one CCNOT gate. Hence, we need $2(n_d + 1)$ CNOT and $n_d + 1$ CCNOT gates for the multi-qubit switch gate for a total of $2n_d + 3$ CNOT and $n_d + 1$ CCNOT gates per crosspoint. Therefore, each quantum concentrator design we presented uses $m(n - m + 1)(2n_d + 3)$ CNOT gates, $m(n - m + 1)(n_d + 1)$ CCNOT gates and $m(n - m + 1)$ auxiliary quantum bits.

For the restoring stage there are $m(n - m + 1)$ inverse switches and $n + m$

copy nodes. Each inverse switch has one switch gate for single qubits and one CNOT gate, which sums up to three CNOT and one CCNOT gate. Each copy node has two CNOT gates and two extra qubits. Thus the total cost for restoring the control qubits is $3m(n-m+1)+2(n+m)$ CNOT gates, $m(n-m+1)$ CCNOT gates and $2(m+n)$ extra qubits. Therefore, the overall cost for a QSC(n,m) is $m(n-m+1)(2n_d+6)+2(n+m)$ CNOT gates, $m(n-m+1)(n_d+2)$ CCNOT gates and $m(n-m+1)+2(n+m)$ auxiliary quantum bits.

3.10.2 Routing Delay

The depth of a QSC(n,m) is given by the maximum possible number of crosspoints between an input and an output. It is easy to see that the longest input-output path is between input 1 and output m. For the fat-slim QSC(n,m), this path length is $(n-m+1)+m-1=n$ crosspoints and for the banded QSC(n,m) the path length is $(n-m+1)+(m-1)+(m-1)=n+m-1$ crosspoints. Hence the depth of fat-slim QSC(n,m) is n and the depth of banded QSC(n,m) is $n+m-1$. The time required to self-route is upper bounded by the depth of the concentrator, thus, self-routing on a fat-slim QSC(n,m) has $O(n)$ delay and self-routing on a banded QSC(n,m) has $O(n+m)$ delay. The time required to self-route is upper bounded by the depth of the concentrator, thus, self-routing on a fat-slim QSC(n,m) has $O(n)$ delay and self-routing on a banded QSC(n,m) has $O(n+m)$ delay.

3.11 Conclusion

We defined the notion of quantum concentration, concentration patterns and quantum crossbar concentrators. We designed self-routing crosspoints and self-routing quantum crossbar concentrators. We proved that fat-slim and banded quantum crossbar concentrators are self routing and thus suitable for quantum implementation. Finally we gave a method to restore the state of auxiliary control qubits to prevent information loss due to decoherence.

In proving that fat-slim and banded crossbar concentrators are self-routable, we assumed as input a quantum assignment consisting of a single pattern of classical packets. Thus, this proves that classical fat-slim and banded crossbar concentrators are self-routable too. Other sparse crossbar concentrator structures apart from fat-slim and banded crossbars could be analyzed for self-routability. The density of (n, m)-sparse crossbar concentrators amongst all $n \times m$ crossbars is known [51], an interesting direction for further investigation would be to identify the class of sparse crossbar concentrators which are self-routable and to quantify the size of this class. In particular, it will be interesting to determine if there exist self-routing regular sparse crossbar concentrators, i.e., those with fixed out-degree inputs and in-degree outputs.

Chapter 4

Bounds on Size of Input Quantum Assignment

Classical switching is a two step process: scheduling of the incoming connection requests and the subsequent routing of the scheduled requests over the switching fabric. Scheduling is used to select a set of packets at the inputs such that the output addresses form a permutation, i.e., there is no contention for outputs. For packet switching, the packets incident on the inputs are queued in buffers while scheduling is done. A large number of scheduling algorithms have been developed which try to optimize different measures of switch performance like delay, throughput, queue lengths etc. [52, 53, 54]. In the simplest case of first-in-first-out (FIFO) service in the input queues, it is known that head of the line blocking limits throughput to 58.6% [55]. It was also shown in [55] that throughput can be increased by choosing a packet at random from among the first m packets, $m > 1$, at the head of each input queue. The parameter m is a system variable which can be optimized.

Now consider an $n \times n$ non-blocking quantum switching network which can route all non-contending quantum input assignments. We assume that classical packets come in at the inputs and get queued in quantum memory buffers which can store the incoming qubit packets. In this scenario, the random selection of a packet from among the first m classical packets is equivalent to creating a quantum

packet containing an equal superposition of the first m packets in the quantum buffer at an input. The input quantum assignment then contains all the possible patterns of n packets formed by choosing, at each input, one packet from the first m packets. The probability of observing each of these input assignment patterns is the same.

The non-blocking quantum network can route all non-contending assignment patterns without blocking, therefore, on measurement at the output we will observe n valid packets with non-zero probability only if there is at least one permutation assignment pattern present in the input quantum assignment. Thus, a natural question to ask is how large should m be such that the input quantum assignment contains at least one permutation pattern. We give bounds on m assuming the incoming traffic is uniform, i.e., a packet at input i is addressed to output j with probability $1/n$ independent of i and j, $1 \leq i, j \leq n$. These bounds are probabilistic in the sense that for a given switch size n, we can calculate a minimum m such that the probability of the input quantum assignment containing at least one permutation assignment is greater than any arbitrary threshold value.

4.1 Problem Statement

We formulate the problem of finding a minimum value of m in terms of density of certain matrices as follows. Let $A_{nm} = \{a_{ij}\}$ be a $n \times m$ matrix where a_{ij} is the output address of the jth packet at input i, $1 \leq i \leq n$, $1 \leq j \leq m$ and $1 \leq a_{ij} \leq n$. Let row i of A_{nm} be denoted by Y_i.

Remark 4.1 (SDR). The matrix A_{nm} is said to have a system of distinct representatives (SDR), if a subset $S = \{s_1, s_2, \ldots, s_n\}$ of n distinct elements can be chosen from A_{nm} such that $s_i \in$ row i, i.e., $s_i \in Y_i$, $1 \leq i \leq n$. The set S is called the system of distinct representatives (SDR).

The quantum input assignment formed by packets with the addresses as specified in A_{nm} contains a permutation assignment pattern only if A_{nm} has at least one SDR. A SDR corresponds to the ordering of packets in a permutation pattern. Since we are assuming uniform distribution on the output addresses, all the n^{nm} matrices are equally likely to occur and the density of $n \times m$ matrices with distinct representatives for the rows gives the probability that the corresponding quantum input assignment has a permutation assignment pattern. The following well-known theorem for distinct representatives by Hall will be very useful in our subsequent calculations

Hall's Theorem [50]. *Let O be a finite set and let $Y_1, Y_2, \ldots, Y_r$ be arbitrary subsets of O. There exist distinct elements $y_i \in Y_i$, $1 \leq i \leq r$ if and only if the union of any k of $Y_1, Y_2, \ldots, Y_r$ contains at least k elements.*

Let O be the set formed by the elements of A_{nm} and $Y_1, Y_2, \ldots, Y_r$ represent the sets of numbers in some r rows, $s_1, s_2, \ldots, s_r$ respectively, i.e., Y_i is the subset of outputs to which packets at input s_i are addressed. Then if the union of $Y_1, Y_2, \ldots, Y_r$ contains at least r outputs for any choices of $s_1, s_2, \ldots, s_r$ in the input set, and any r, $1 \leq r \leq n$, then Hall's theorem implies that the quantum assignment contains a permutation pattern.

Let $\rho(n, m)$ be the density of matrices for which distinct representatives for rows exist. Then

Theorem 4.1.

$$\rho(n, m) \leq \frac{n!\, \mathcal{S}_{nm}^{(n)}}{n^{nm}} \tag{4.1}$$

where

$$\mathcal{S}_{nm}^{(n)} = \frac{1}{n!} \sum_{i=0}^{nm} (-1)^{n-i} \binom{n}{i} i^n \tag{4.2}$$

is a Stirling number of the second kind and is equal to the number of ways an nm-set can be partitioned into n distinct non-empty sets. The bound in Eqn. (4.1) *holds with equality for* $n = 2$*, therefore,* $\rho(2, m) = 1 - 2^{-(2m-1)}$.

Proof. Consider the set of nm elements $\{a_{ij}\}$ partitioned into n subsets $X_1, \ldots, X_n$ such that $x_i \in X_i \Rightarrow x_i = i$, $1 \leq i \leq n$. If A_{mn} has a SDR then it can partitioned into n such subsets so that none of the subsets is empty. Hence, the set of matrices for which $X_1, X_2, \ldots, X_n$ are all non-empty contains the set of all matrices with SDRs. There are $\mathcal{S}_{nm}^{(n)}$ ways to partition a set of nm elements into n non-empty sets when order is not considered [56]. Hence there are $n!\, \mathcal{S}_{nm}^{(n)}$ distinct matrices which can be partitioned into n non-empty subsets in the way described above. Since all matrices A_{nm} with an SDR are contained in this set of matrices, the bound in Eqn. (4.1) follows.

When $n = 2$, if two non-empty partitions exist then the corresponding matrix

definitely has an SDR and hence the bound in Eqn. (4.1) is tight. Hence

$$\rho(2,m) = 2 \cdot \mathcal{S}_{2m}^{(2)} / 2^{2m} \tag{4.3}$$

$$= 2(2^{2m-1} - 1)/2^{2m} = 1 - 2^{-(2m-1)} \tag{4.4}$$

where we have used the fact that $\mathcal{S}_n^{(2)} = 2^{n-1} - 1$. This can be verified by direct evaluation from Eqn. (4.2). □

We now tighten the bounds on $\rho(n,m)$ for $n > 2$ by using Hall's theorem.

Theorem 4.2.

$$\frac{n!\,\mathcal{S}_{nm}^{(n)} - \sum_{k=2}^{n-1}\sum_{l=1}^{k-1}\sum_{p=n-l}^{n}\binom{n}{k}\binom{n}{l}\,l!\,\mathcal{S}_{mk}^{(l)}\,p!\,\mathcal{S}_{m(n-k)}^{(p)}}{n^{nm}} \leq \rho(n,m) \tag{4.5}$$

$$\rho(n,m) \leq \frac{n!\,\mathcal{S}_{nm}^{(n)} - \sum_{k=2}^{n-1}\sum_{l=1}^{k-1} n!\,\binom{n}{k}\,\mathcal{S}_{mk}^{(l)}\,\mathcal{S}_{m(n-k)}^{(n-l)}}{n^{nm}} \tag{4.6}$$

Proof. Consider all $n \times m$ matrices in which the union of some k rows contains exactly $l, l < k$ distinct numbers and the union of the rest of the $n-k$ rows contains only the remaining $n-l$ distinct numbers. Clearly this class of matrices can be partitioned into n non-empty sets $X_1, \ldots, X_n$, with set X_i consisting of all a_{bc} such that $a_{bc} = i$. But as the union of some k rows contains less than k distinct elements, by Hall's theorem these matrices do not have a SDR. The number of such matrices for fixed k and $l, l < k$ is given by:

$$\binom{n}{k}\binom{n}{l}\ \underbrace{l!\,\mathcal{S}_{mk}^{(l)}}_{I}\ \underbrace{(n-l)!\,\mathcal{S}_{m(n-k)}^{(n-l)}}_{II} \tag{4.7}$$

as we can choose k rows in $\binom{n}{k}$ ways, l elements out of n in $\binom{n}{l}$ ways and the term marked as I is the total number of $k \times m$ sub-matrices which have l distinct entries

and the term marked II is the number of $(n-k) \times m$ sub-matrices which have $n-l$ distinct entries. Summing up over all k and l we get

$$\sum_{k=2}^{n-1} \sum_{l=1}^{k-1} \binom{n}{k} \binom{n}{l} \; l! \mathcal{S}_{mk}^{(l)} \; (n-l)! \mathcal{S}_{(n-k)m}^{(n-l)} \tag{4.8}$$

$$= \sum_{k=2}^{n-1} \sum_{l=1}^{k-1} n! \binom{n}{k} \mathcal{S}_{mk}^{(l)} \mathcal{S}_{(n-k)m}^{(n-l)} \tag{4.9}$$

hence, the upper bound in Eqn. (4.6) follows.

Now consider all $n \times m$ matrices in which the union of some k rows contains l distinct elements $l < k$ and the remaining $n-l$ distinct elements appear *at least once* in the rest of the $n-k$ rows, i.e., their union contains at least $n-l$ distinct entries. These also violate the conditions of Hall's theorem and hence do not contain an SDR. The total number of such matrices for a fixed k and l is upper bounded by

$$\binom{n}{k} \binom{n}{l} \; l! \; \mathcal{S}_{mk}^{(l)} \sum_{p=n-l}^{n} p! \; \mathcal{S}_{m(n-k)}^{(p)} \tag{4.10}$$

this is an upper bound as we are over-counting matrices in the summation term for index p. Summing up over all values of k and l we get the lower bound in Eqn. (4.5). □

4.2 Some Results

These bounds are close to each other and give a fairly good estimate of the density, ρ. We can use these bounds to estimate, for a given switch size, n, the minimum value of m at which the density exceeds a certain threshold, say η. Then we know that, if the traffic is uniform, then with probability η, the output quantum assignment contains a permutation pattern with all n valid packets. Note that the proba-

n	Superposition size, m								
	1	2	3	4	5	6	7	8	9
2	0.5	0.875	0.969	0.992	0.998	0.999	$\simeq 1$	$\simeq 1$	$\simeq 1$
4	0.094	0.556	0.863	0.958	0.987	0.996	0.999	$\simeq 1$	$\simeq 1$
8	0.002	0.053	0.674	0.889	0.962	0.987	0.995	0.998	0.999
16	$\simeq 10^{-6}$	0.04	0.424	0.764	0.911	0.968	0.988	0.996	0.999
32	$\simeq 10^{-13}$	0.03	0.17	0.563	0.816	0.930	0.974	0.991	0.997
64	$\simeq 10^{-27}$	0.01	0.024	0.304	0.655	0.859	0.946	0.980	0.993

Table 4.1: Lower bound on $\rho(n, m)$.

bility of observing this particular pattern on measurement will be dependent on its associated probability amplitude. The minimum values $\rho(n, m)$ calculated using the lower bound in Eqn. (4.5) are given in Table 4.1. From these values it can be seen that the density increases rapidly and saturates very quickly at around 0.99 for $m = 9$. Thus a quantum packet consisting of a superposition of nine classical packets at the head of each input almost guarantees that the output quantum assignment of the quantum non-blocking network will have a permutation packet pattern.

Chapter 5

Conclusions and Future Work

We have defined the concepts of quantum packets, quantum assignments, quantum non-blocking networks and quantum concentrators.

5.1 Random Routing in Quantum Switching Networks

We have used the inherent random nature of quantum bits and the property of quantum parallelism to identify that random routing on any quantum switch network is equivalent to creating quantum superpositions of the input packets at the outputs of the individual switches which constitute the network. We have characterized the output state of the quantum Baseline network and shown an explicit relation between routing probabilities and the amplitudes of the packet patterns in the quantum output.

For classical $2^n \times 2^n$ Beneš networks certain classes of permutations like the bit-permute-complement class ($\mathcal{BPC}_n$) [45] and linear-complement class ($\mathcal{LC}_n$) [46] can be self-routed without blocking, but any permutation not belonging to either of these classes always encounters blocking if it is routed using bit-controlled routing. For the quantum Beneš network we have shown that for *any* arbitrary permutation input there is a finite non-zero probability that the output will correspond to packets being routed without blocking when packet routing is done using quantum

randomization and bit controlled routing. We have derived this result by utilizing results implied by Lee's routing algorithm for Beneš networks [48]. In addition to the lower bound on the probability we have found a new class of permutations for which this lower bound is equal to one. This implies that for any permutation pattern belonging to this class, *all* the packet patterns in the quantum output correspond to routing without blocking over the Beneš network. For the classical Beneš network this means that if an input permutation assignment belongs to the newly discovered class then the switches in the first $n-1$ stages can be set to any arbitrary state and the resulting output is still self-routable over the last n stages.

5.2 Self-Routing Quantum Sparse Crossbar Concentrators

We have given the design of quantum sparse crossbar (n, m)-concentrators. These concentrator structures are based on classical sparse crossbar concentrators where the classical crosspoint switches are replaced by quantum crosspoints. Many proposed quantum computing architectures are based on planar arrangement of solid-state components [17] and the low crosspoint count and the planar structure of such concentrators make them suitable for implementation in such architectures. In quantum systems all operations need to be reversible and hence a "rectangular" $n \times m$ structure can not be used. We have given a method to convert any "rectangular" crossbar into a "square" crossbar while preserving the concentration property. Since classical control for quantum systems is slow and interaction with the environment can lead to decoherence of the state of qubits, it is desirable that packets

on quantum switching networks are self-routed. Accordingly, we have given a self-routing scheme for concentration on suitable quantum sparse crossbar structures. In particular, we have rigorously proved that using our scheme fat-slim and banded sparse crossbars can self-route packets to realize any concentration pattern. We also give a method to restore the auxiliary control qubits back to their original state to prevent information loss due to decoherence.

5.3 Bounds on Size of Input Quantum Assignment

In the penultimate chapter we try to answer to following question in a meaningful way: If m packets are superposed to create a quantum packet at every input to an $n \times n$ quantum non-blocking network then how large should this value m be to get good performance? We assume uniform distribution on the output destinations and use the probability of getting a non-contending packet pattern in the output quantum state as a measure of performance. For this criterion, we have derived closed form upper and lower bounds by using combinatorial arguments to calculate the density of matrices which contain distinct representatives for the subsets defined by their rows. Numerical calculation of the bounds indicates that $m = 9$ is sufficient to ensure a probability greater than 0.99 for all n upto at least 64.

5.4 Future Work

Some future directions of investigation based on this work are given below.

1. For quantum Beneš networks, we used randomization to show that such net-

works are probabilistically non-blocking for any arbitrary permutation input. Lee's algorithm allows routing without any blocking on such networks but since it is not decentralized, it is not suitable for quantum switching networks. For 4×4 Beneš networks we have been able to design quantum switches based on comparators which implement Lee's algorithm in a decentralized fashion, i.e., we have the design of a fully non-blocking 4×4 quantum Beneš network. It will be interesting to investigate whether this approach can be used to get non-blocking switches for values of n larger than four.

2. We have proved that fat-slim and banded crossbar structures are self-routable and hence suitable for implementation in the quantum domain. Sparse crossbar concentrator structures other than fat-slim and banded crossbars could be analyzed for self-routability. An interesting direction for further investigation would be to identify the class of sparse crossbar concentrators which are self-routable and to quantify the size of this class. Our routing algorithm for these concentrators is biased towards the lower numbered switches which get routing priority over higher numbered switches. Thus for capacity exceeding input assignments the m lowest numbered input packets get concentrated. Methods to overcome this bias while maintaining self-routability can be explored. An interesting option would be to create superpositions at the internal crosspoints whenever there is a contention for an output by two valid packets.

3. For the investigation into the bound on the number of superposed packets in an input quantum packet we did not calculate the throughput of the non-blocking switch. That is, for all the input packet patterns which have contention, we did not calculate how many packets in the output pattern are routed correctly to their desired output destinations. We expect the results to mirror those for the classical case in which packets for routing are scheduled at random from a window of fixed size at each input queue [55].

The primary goal of this dissertation has been to explore the possible relations between classical packet switching networks and quantum computing. As part of this exploration, we have determined that there is a close relation between random routing and in classical packet switching and quantum superposition. This is turn has led us to develop quantum switching network models by which such networks can be methodically analyzed, designed and routed. Examples have been given to show the power and utility of quantum switching network model in each case. Yet, as is typically the case in any research, we conclude this work with more open questions than the ones with which we had begun.

Bibliography

[1] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, pp. 467–488, June 1982. 1

[2] P. W. Shor, "Scheme for reducing decoherence in quantum memory," *Physical Review A*, vol. 52, no. 4, pp. 2493–2496, 1995. 1

[3] C. H. Bennett, G. Brassard, and A. K. Ekert, "Quantum cryptography," *Scientific American*, vol. 267, 1992. 1

[4] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Review*, vol. 41, no. 2, pp. 303–332, Jun 1999. 1, 12

[5] A. M. Childs, E. Farhi, and J. Preskill, "Robustness of adiabatic quantum computation," *Physical Review A*, vol. 65, no. 1, Dec 2001. 2

[6] S. Hallgren, "Polynomial-time quantum algorithms for pell's equation and the principal ideal problem," in *STOC '02: Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM Press, 2002, pp. 653–658. 2

[7] W. van Dam and G. Seroussi. (2002) Efficient quantum algorithms for estimating gauss sums. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0207131 2

[8] C. Monroe, D. M. Meekhof, B. E. King, W. M. Itano, and D. J. Wineland, "Demonstration of a fundamental quantum logic gate," *Physical Review Letters*, vol. 75, p. 4714, 1995. 2

[9] D. Vion, A. Aassime, A. Cottet, P. Joyez, H. Pothier, C. Urbina, D. Esteve, and M. H. Devoret, "Manipulating the quantum state of an electrical circuit," *Science*, vol. 296, p. 886, 2002. 2

[10] Q. A. Turchette, C. J. Hood, W. Lange, H. Mabuchi, and H. J. Kimble, "Measurement of conditional phase shifts for quantum logic," *Physical Review Letters*, vol. 75, p. 4710, 1995. 2

[11] L. M. Vandersypen, M. Steffen, G. Breytta, C. S. Yannoni, R. Cleve, and I. L. Chuang, "Experimental realization of order-finding with a quantum computer," *Physical Review Letters*, December 15 2000a. 2

[12] B. Kane, "A silicon-based nuclear spin quantum computer," *Nature*, vol. 393, pp. 133–137, 1998. 2

[13] D. Copsey, M. Oskin, F. Impens, T. Metodiev, A. Cross, F. T. Chong, I. L. Chuang, and J. Kubiatowicz, "Toward a scalable, silicon-based quantum computing architecture," *IEEE Journal Of Selected Topics in Quantum Electronics*, vol. 9, no. 6, pp. 1552–1569, Nov-Dec 2003. 2

[14] M. Oskin, F. T. Chong, I. L. Chuang, and J. Kubiatowicz, "Building quantum wires: the long and the short of it," in *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*, June 2003, pp. 374–385. 2

[15] W. Wootters and W. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, pp. 802–803, 1982. 2, 15

[16] N. Isailovic, M. Whitney, Y. Patel, J. Kubiatowicz, D. Copsey, F. T. Chong, I. L. Chuang, and M. Oskin, "Datapath and control for quantum wires," *ACM Transactions on Architecture and Code Optimization*, vol. 1, no. 1, pp. 34–61, 2004. 2

[17] T. S. Metodi, D. D. Thaker, and A. W. Cross, "A quantum logic array microarchitecture: Scalable quantum data movement and computation," in *Proceedings of 38th Annual IEEE/ACM International Symposium on Microarchitecture, 2005. MICRO-38*, Nov 2005, pp. 305–318. 3, 119

[18] I. M. Tsai and S.-Y. Kuo, "Digital switching in the quantum domain," *IEEE Transactions on Nanotechnology*, vol. 1, no. 3, pp. 154–164, 2002. 3

[19] M. K. Shukla, R. Ratan, and A. Y. Oruç, "A quantum self-routing packet switch," in *Proceedings of the 38th Annual Conference on Information Sciences and Systems CISS'04*, Princeton, NJ, USA, March 2004, pp. 484–489. 3, 15

[20] ——, "The quantum baseline network," in *Proceedings of the 39th Annual Conference on Information Sciences and Systems CISS'05*, Johns Hopkins University, Baltimore, MD, March 2005. vii, 3, 15, 18, 38, 39, 43, 46

[21] S. T. Cheng and C. Y. Wang, "Quantum switching and quantum merge sorting," *IEEE Transactions on Circuits and Systems I: Regular Papers* [*see also IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*], vol. 53, no. 2, pp. 316–325, Feb 2006. 4

[22] R. Ratan and A. Y. Oruç, "Quantum switching networks with classical routing," in *Proceedings of the 41st Annual Conference on Information Sciences and Systems CISS'07*, Johns Hopkins University, Baltimore, MD, March 2007, pp. 789–793. 4

[23] S. Nakamura and G. M. Masson, "Lower bounds on crosspoints in concentrators," *IEEE Transactions on Computers*, vol. C-31, no. 12, pp. 1173–1179, 1982. 6, 70

[24] A. Yavuz Oruç and H. M. Huang, "Crosspoint complexity of sparse crossbar concentrators," *IEEE Transactions on Information Theory*, vol. 42, no. 5, pp. 1466–1471, 1996. 6, 64, 71, 88

[25] W. Guo and A. Y. Oruç, "Regular sparse crossbar concentrators," *IEEE Transactions on Computers*, vol. 47, no. 3, pp. 363–368, 1998. 6, 64, 65, 66, 71, 92

[26] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, September 2000. 9, 15

[27] D. Deutsch, "Quantum computational networks," *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 425, no. 1868, pp. 73–90, Sep. 8 1989. [Online]. Available: http://links.jstor.org/sici?sici=0080-4630%2819890908%29425%3A1868%3C7%3%3AQCN%3E2.0.CO%3B2-5 10

[28] J. Preskill. (1998) Physics 229: Advanced mathematical methods of physics – quantum computation and information. [Online]. Available: http://www.theory.caltech.edu/people/preskill/ph229 11

[29] C. E. Shannon, "Memory requirements in a telephone exchange," *Bell System Technical Journal*, vol. 29, pp. 343–349, 1950. 16

[30] C. Clos, "A study of non-blocking switching networks," *Bell System Technical Journal*, vol. 32, no. 2, pp. 406–424, March 1953. 16

[31] V. E. Beneš, *Mathematical Theory of Connecting Networks and Telephone Traffic*. Academic Press, 1965. 17, 41

[32] D. G. Cantor, "On nonblocking switching networks," *Networks*, vol. 1, pp. 367–377, 1971. 17

[33] C.-Y. Lee and A. Y. Oruç, "Design of efficient and easily routable generalized connectors," *IEEE Transactions on Communications*, vol. 43, no. 234, pp. 646–650, February 1995. 17

[34] J. H. Patel, "Performance of processor memory interconnections for multiprocessors," *IEEE Transactions on Computers*, vol. 30, no. 10, pp. 771–780, October 1981. 18, 52

[35] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Transactions on Computers*, vol. 25, pp. 1145–1155, 1976. 18

[36] M. K. Shukla and A. Y. Oruç, "Multicasting in quantum networks," 2008, working Draft—To be submitted to IEEE Transactions on Computers. 20

[37] M. K. Shukla, "Quantum unicast and multicast switching networks," Ph.D. dissertation, University of Maryland, College Park, in preparation. 20

[38] R. Ratan, M. K. Shukla, and A. Y. Oruç, "On random routing and its application to quantum interconnection networks," in *Proceedings of the 40th Annual Conference on Information Sciences and Systems CISS'06*, Princeton, NJ, USA, March 2006, pp. 1744–1749. 33

[39] D. S. Parker, "Notes on shuffle/exchange-type switching networks," in *Interconnection Networks for Multiprocessors and Multicomputers: Theory and Practice*, A. Varma and C. S. Raghavendra, Eds. IEEE Computer Society Press, 1994, ch. 4, pp. 124–133. 35, 52

[40] H. Çam, "Rearrangeability of (2n -1)-stage shuffle-exchange networks," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 557–585, March 2003. 38

[41] A. Waksman, "A permutation network," *Journal of the ACM*, vol. 15, no. 1, pp. 159–163, 1968. 42, 53

[42] D. Nassimi and S. Sahni, "Parallel algorithms to set up the Benes permutation network," *IEEE Transactions on Computers*, vol. 31, pp. 148–154, 1982. 42

[43] 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. 42

[44] G. F. Lev, N. Pippenger, and L. G. Valiant, "A fast parallel algorithm for routing in permutation networks," *IEEE Transactions on Computers*, vol. 30, no. 2, pp. 93–100, 1981. 42

[45] D. Nassimi and S. Sahni, "A self routing Benes network," in *ISCA '80: Proceedings of the 7th Annual Symposium on Computer Architecture*. New York, NY, USA: ACM Press, 1980, pp. 190–195. 43, 118

[46] C. S. Raghavendra and R. V. Boppana, "On self-routing in Benes and shuffle-exchange networks," *IEEE Transactions on Computers*, vol. 40, no. 9, pp. 1057–1064, 1991. 43, 118

[47] K. Y. Lee, "On the rearrangeability of $2(\log_2 N) - 1$ stage permutation networks," *IEEE Transactions on Computers*, vol. C-34, no. 5, pp. 412–425, May 1985. 43, 48

[48] ——, "A new Benes network control algorithm," *IEEE Transactions on Computers*, vol. C-36, no. 6, pp. 768–772, June 1987. 44, 48, 119

[49] W. Guo, "Design and optimization of switching fabrics for ATM networks and parallel computer systems," Ph.D. dissertation, Dept. of Electrical Engg., University of Maryland, College Park, MD, August 1996. 66

[50] P. Hall, "On representatives of subsets," *Journal of London Mathematical Society*, pp. 134–151, 1935. 70, 113

[51] E. Günduzhan and A. Y. Oruç, "Structure and density of sparse crossbar concentrators," in *DIMACS Series in Discrete Mathematics and Computer Science, Advances in Switching Networks*, 1998, vol. 42, pp. 169–180. 110

[52] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1260–1267, 1999. 111

[53] P. Giaccone, B. Prabhakar, and D. Shah, "Towards simple, high-performance schedulers for high-aggregate bandwidth switches," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2002, pp. 1160–1169 vol.3. 111

[54] C. S. Chang, W. J. Chen, and H. Y. Huang, "Birkhoff-von neumann input buffered crossbar switches," in *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, 2000, pp. 1614–1623. 111

[55] M. G. Hluchyj and M. J. Karol, "Queuing in high-performance packet switching," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1587–1597, Dec. 1988. 111, 122

[56] M. Abramowitz and I. A. Stegun, Eds., *Handbook of Mathematical Functions*. Washington D.C.: Courier Dover Publications, 1964, pp. 824–825. 114