Properties and Algorithms of the Hyper-Star Graph and its Related Graphs

by

© Fan Zhang

A Thesis Submitted to the

Department of Computer Science

in Partial Fulfilment of the

Requirements for the Degree of

Master of Science

Department of Computer Science

Brock University

August 2009

St. Catharines Ontario

Approved for the Committee:

Dr. K. Qiu (Supervisor)

Dr. S. Houghten

Dr. V. Wojcik

Department of Computer Science

Abstract

The hyper-star interconnection network was proposed in 2002 to overcome the drawbacks of the hypercube and its variations concerning the network cost, which is defined by the product of the degree and the diameter. Some properties of the graph such as connectivity, symmetry properties, embedding properties have been studied by other researchers, routing and broadcasting algorithms have also been designed. This thesis studies the hyper-star graph from both the topological and algorithmic point of view. For the topological properties, we try to establish relationships between hyper-star graphs with other known graphs. We also give a formal equation for the surface area of the graph. Another topological property we are interested in is the Hamiltonicity problem of this graph.

For the algorithms, we design an all-port broadcasting algorithm and a single-port neighbourhood broadcasting algorithm for the regular form of the hyper-star graphs. These algorithms are both optimal time-wise.

Furthermore, we prove that the folded hyper-star, a variation of the hyper-star, to be maixmally fault-tolerant.

Acknowledgements

First of all, I want to thank my supervisor Dr. Ke Qiu for giving me this precious opportunity to study at Brock. Also, big thanks to him for always being so positive and considerate.

I thank the members of my supervisory committee (Dr. K. Qiu, Dr. S. Houghten, Dr, V. Wojcik) for their comments, criticisms and suggestions.

I am grateful to all my fellow graduate students for the help they offered.

Last but not least, nothing could be done without my supportive family.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Parallel Computational Models

Parallel computing has been studied during the past few decades due to two major motivations:

1. Saving time that needs to be spent on solving one specific problem. Intuitively, making multiple computing units to co-operate simultaneously ought to outperform a conventional sequential computer.

2. Some problems can not be solved by a sequential computer however much time one is willing to spend. Due to the nature of these problems, they have to be dealt with in parallel.

The parallel computational model plays a major role in parallel computation. The parallel algorithms, without which no problems can be solved in parallel, have to be designed specifically for one parallel computational model. The reason for this is, unlike the sequential computer, data has to traverse between different processors. So

the communication between processors becomes a new concern in parallel computing. Based on how the processors communicate with each other, the parallel computational models [5] can be divided into two classes: the shared-memory machines and the interconnection networks. These two different models are introduced in the next two sections.

1.2 Shared-Memory Machine

Before talking about the shared-memory machines, we first introduce a basic classification of computer architectures, proposed by Michael J. Flynn [16]. The four classes are based upon the number of concurrent instructions and data streams available in the architecture: seen by the processor during program execution. Depending on whether there is one or several of these streams, computers can be divided into four classes:

- Single Instruction, Single Data Stream (SISD)

  A SISD computer is a general sequential machine. There is no parallelism in either the instruction or data streams in this class of computers.

- Multiple Instruction, Single Data Stream (MISD)

  In computing, a MISD computer contains N processors, each has its own control unit and all processors share a common memory unit. In this kind of computers, parallelism is achieved by using many functional units to perform different operations on the same data. But in practice, there is no known implementation of this class of computers so far.

- Single Instruction, Multiple Data Stream (SIMD)

  A SIMD computer consists of N identical processors, each with its own local memory to store data. All processors work under the control of a single instruction stream issued by a central control unit. The processors operate synchronously: at each step, all processors execute the same instruction on a different data element. SIMD computers are much more versatile than MISD computers.

- Multiple Instruction, Multiple Data Stream (MIMD)

  In a MIMD computer, multiple autonomous processors simultaneously executing different instructions on different data, where each processor has its own control unit and local memory. Therefore, processors are potentially all executing different programs on different data while solving different subporblems of a single problem. This makes MIMD computers more powerful than other three classes of computers.

In a shared-memory machine, all processors communicate with each other through a shared memory. The shared-memory machine, which is also known as Parallel Random Access Machine (PRAM), is shown in Figure 1.1. It consists of a bunch of identical processors and a common memory which is shared by these processors. Depending whether multiple processors can read from or write to the same memory location at the same time is allowed or not, the shared-memory machines can be divided into four groups:

- **Exclusive Read, Exclusive Write (EREW):**

  Only one processor can read from any one memory location at a time and only

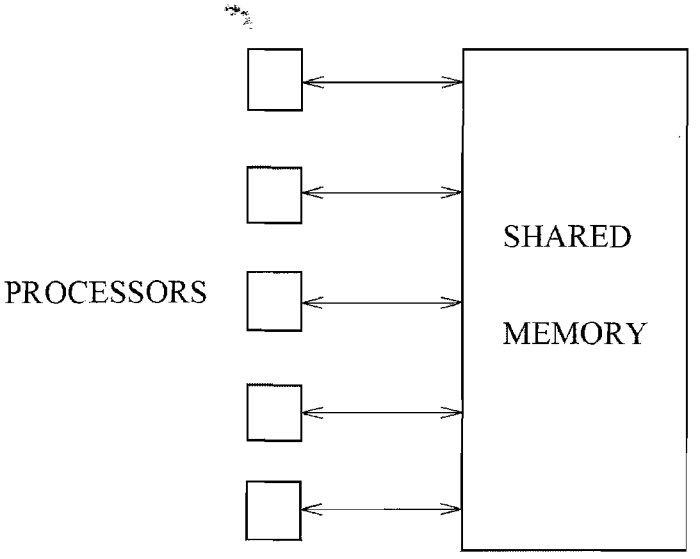

Figure 1.1: Shared-Memory Machine

one processor can write to any one memory location at a time. In this scenario, no conflict will occur when processors are accessing their data.

- **Exclusive Read, Concurrent Write (ERCW):**

  Only one processor can read from a memory location while multiple processors can write to a memory location at the same time. This kind of PRAM model has very little practicle value.

- **Concurrent Read, Exclusive Write (CREW):**

  Multiple processors can read from one memory location at one time but only one processor can write to a memory location at one time.

- **Concurrent Read, Concurrent Write (CRCW):**

  Multiple processors are allowed to either read from or write into the same location at the same time. In theory, this model is the most powerful among the

four.

When concurrent write (CW) instruction is allowed in a PRAM, conflicts are unavoidable. In order to decide what ends up in the specific memory location when several processors attempt to write into that location simultaneously, some methods need to be adopted when executing a CW instruction. These methods include:

1. **Priority CW:** Here the processors are assigned certain priorities. Of all the processors attempting to write in a memory location, only the one with the highest priority is allowed to do so.

2. **Common CW:** Here the processors attempting to write in a memory location are allowed to do so only if they are trying to write the same value. In this case, one processor that is selected arbitrarily succeeds. The instruction is further specified as follows:

   (a) **Fail Common:** If the values to be written by the processors into a memory location are not all equal, then the contents of the memory location are unchanged.

   (b) **Collision Common:** This instruction requires a "failure" label to be stored in the memory location in case the **CW** does not succeed.

   (c) **Fail-Safe Common:** Here failure is not tolerated. The algorithm must be designed in such a way that whenever more than one processor wishes to write into the same memory location, they must be trying to write the same value.

3. **Arbitrary CW:** Of all the processors attempting to write simultaneously in a memory location, any one can succeed without affecting the correctness of the algorithm.. However, the algorithm must specify exactly how the successful processor is to be selected.

4. **Random CW:** Here the processor that succeeds in writing is chosen by a random process.

5. **Combining CW:** In this model of concurrent writing, all the values that those processors wishes to write simultaneously into the same memory location are combined into a single value, which is then stored in that location. This instruction takes several forms, depending on which function the algorithm needs to use to combine the values before storing the result. The following variants are available:

   (a) *Arithmetic functions:* The values to be written are either added up or multiplied.

   (b) *Logical functions:* A set of Boolean values can be combined using **AND, OR** or **EXCLUSIVE-OR**. The negations of these functions are also available.

   (c) *Selection function:* The largest or smallest of the values to be written is selected.

1.3 Interconnection Network

Unlike the shared-memory machines, processors in an interconnection network communicate with each other via direct links. Each processor has its own local memory, and the links between a pair of processors are *two-way* communication lines, in other words, two processors connected by a link can exchange data simultaneously.

The interconnection network is usually modeled as an undirected graph $G = (V, E)$. The vertex set V of G represents the processors of the interconnection network, and the edges in E represent the links between pairs of processors. There is an edge between two vertices in G if and only if there is a direct link between the two corresponding processors in the interconnection network. Throughout this thesis, we will use the terms "interconnection network" and "graph" interchangeably.

Thus, the criteria used to evaluate an undirected graph are also suitable for the analysis of the corresponding interconnection network. These criteria are defined formally as follows.

Definition 1 *Two processors directly connected by a link are said to be* **neighbours**.

Definition 2 *If an edge $e = (u, v) \in E$, then the nodes u and v are said to be* **adjacent** *and the edge e is said to be* **incident** *on these nodes.*

Definition 3 *The* **degree** *of a node $v \in V$ is equal to the number of edges in G which are incident on v. The* **degree** *of a graph is the maximum of all node degrees.*

Definition 4 *A graph is* **regular** *if all its nodes have the same degree. Such a graph with degree n is called n-***regular**.

Definition 5 *The* **distance** *between a pair of nodes u and v in G, is equal to the length (in the number of edges) of a shortest path joining u and v. The* **diameter** *of G is the maximum distance between two nodes in G over all pairs of nodes in V.*

Definition 6 *The* **node connectivity** *of a graph is the minimum number of nodes that must be removed to disconnect the graph or reduce it to a solitary point.*

Definition 7 *An* **automorphism** *of a graph is a mapping from the vertices of the given graph G back to vertices of G such that the resulting graph is isomorphic with G.*

Definition 8 *A graph is said to be* **vertex symmetric** *if for every pair of vertices u and v, there exists an automorphism of the graph that maps u into v. A graph is said to be* **edge symmetric** *if for every pair of edges, a and b, there exists an automorphism of the graph that maps a into b.*

A special class of graphs, namely the ***Cayley Graphs*** are all vertex and edge symmetric [3][24][19].

Definition 9 *A graph G is f-**fault tolerant** if whenever f or less than f nodes are deleted from G, the remaining graph is still connected. The* **fault tolerance** *of the graph G is said to be the maximum number of f for which it is f-fault tolerant.*

The interconnection networks are further classified into two models based on how many neighbours one processor can communicate with at one time. In the ***weak*** model (also known as the ***single-port*** model), a processor can only communicate with one of its neighbours at a time. In the ***strong*** model (also known as the ***all-port*** model), a processor can communicate with all of its neighbours at the same

time. Communication problems on the interconnection networks are often considered separately for these two different models.

A lot of interconnection networks have been proposed and studied during the last few decades [25][34][33], we will introduce some of the most popular ones in the rest of this section.

1.3.1 Linear Array

In the linear array interconnection network, each processor is connected to two neighbours except for the two end processors, which have only one neighbour. Thus, all the processors in this interconnection network form a one-dimensional array. It is easy to see that the degree of the linear array of size N is 2 and the diameter is $O(N)$. If the two end processors are connected by a direct link, then it forms a special case of the linear array, namely the ***ring*** interconnection network. A linear array interconnection network is shown in Figure 1.2.

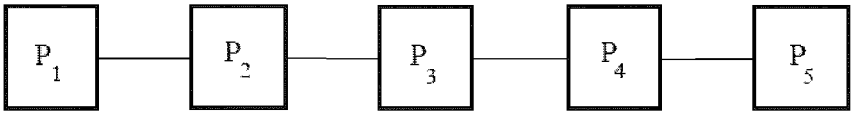

Figure 1.2: A Linear Array with 5 Processors

1.3.2 Mesh

The corresponding undirected graph of the mesh interconnection network is a two-dimensional array. The processor in row i and column j is denoted by $P_{i,j}$, the neighbours of $P_{i,j}$ will be $P_{i-1,j}$, $P_{i+1,j}$, $P_{i,j-1}$ and $P_{i,j+1}$ if they exist. Except for the processors on the boundary rows and columns, all processors have four neighbours. Processors on the boundary rows and columns have less than four neighbours. Thus the degree of mesh is 4. It can also be proved easily that the degree of a mesh with m rows and n columns is $O(m+n)$. Figure 1.3 shows a 3×3 mesh, the row and column indices are shown in the graph.

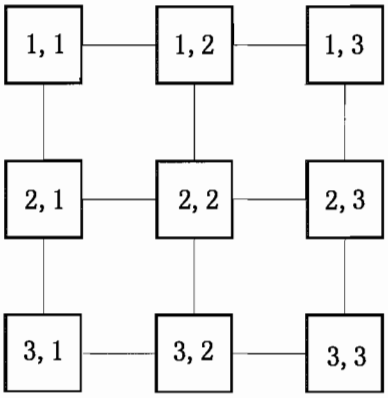

Figure 1.3: A 3×3 Mesh

1.3.3 Hypercube

An n-dimensional hypercube Q_n consists of $N = 2^n$ processors interconnected as follows. Each processor is labeled by a different n-bit binary number. Two processors are connected to each other if and only if their binary labels differ in exactly one bit

position. The degree and diameter of an n-dimensional hypercube are both n. The left part of Figure 1.4 shows a 3-cube. The hypercube graph is arguably the most popular graph ever proposed to be used as a computing network [32], many variations of hypercube have been proposed and studied [13][9]. For the rest of this subsection we will give a brief introduction to three of those variations of the hypercube.

1.3.3.1 Cube-Connected Cycles

The design of the cube-connected cycles aims at not only implementing algorithms efficiently, but also complying with the technological constraints, so that the cube-connected cycles can actually be used in the layout of many specialized large scale integrated circuits (VLSI) [29].

To obtain a cube-connected cycles network, we begin with a n-dimensional hypercube and then replace each of its 2^n corners with a ring of n processors. Each processor in a ring is connected to a processor in a neighboring ring in the same dimension. The number of processors is $N = 2^n \times n$

1.3.3.2 Twisted Cube

Let C be any shortest cycle (i.e., a 4-cycle) in Q_n. Also, let (u, x) and (v, y) be any two independent (do not share an endpoint) edges in C. The twisted n-cube graph [15][11] TQ_n is then constructed as follows. Delete edges (u, x) and (v, y) from Q_n. Then, connect, via an edge, vertex u to vertex y, and vertex v to vertex x. That is, $TQ_n = Q_n - \{(u, x), (v, y)\} + \{(u, y), (v, x)\}$. TQ_n is n-regular just as Q_n is. Also, TQ_n has two disjoint Q_{n-1}'s as subgraphs.

The cube can be twisted around any 4-cycle, the canonically twisted Q_n is the one

whose vertices u,v,x, and y have the labels $b(u) = 000\ldots0$, $b(v) = 010\ldots0$, $b(x) = 100\ldots0$, $b(y) = 110\ldots0$. Q_3 and TQ_3 are shown in Figure 1.4.

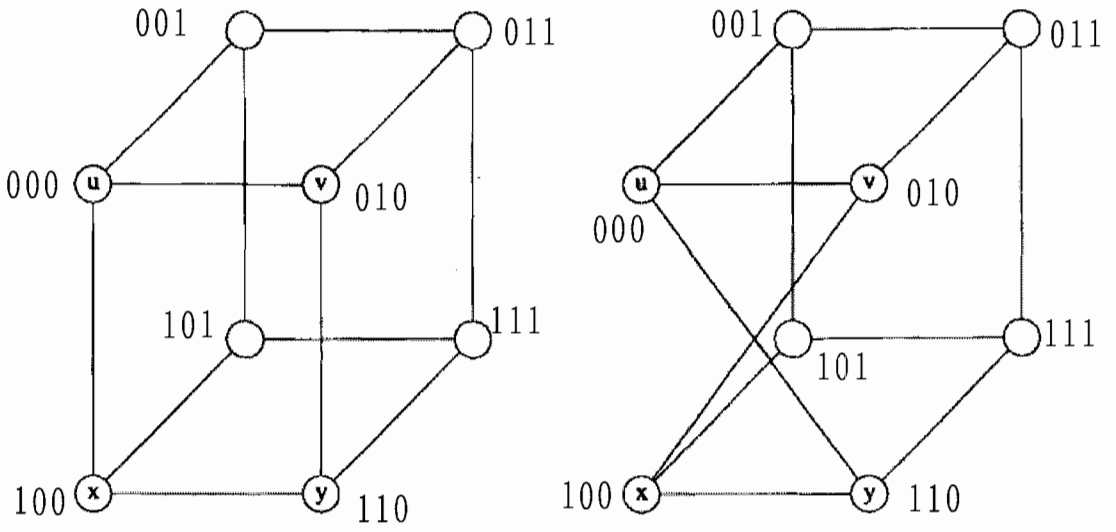

Figure 1.4: A 3-Cube and its Corresponding Twisted Cube

1.3.3.3 Folded Hypercube

The folded hypercube was proposed to further reduce the diameter and traffic congestion, with little hardware overhead [14]. A folded hypercube of dimension n, $FHC(n)$ can be constructed from a standard binary n-cube by connecting each node to the unique node that is farthest from it.

Formally, an n-dimensional folded hypercube can be modeled as a graph $F(V, E)$. It has the same set of vertices as in the hypercube with the same dimension, and its edge set is a superset of the hypercube's. And the hypercube is a spanning subgraph of the corresponding folded hypercube. In addition, an edge (u, v) is in F iff $H(u$

XOR $v) = 1$ or n (Hamming weight). In other words, two nodes are connected if and only if their binary representation differ in 1 bit or are complement to each other.

1.3.4 Star

For a positive integer $n > 1$, an n-star interconnection network S_n has $n!$ vertices, the label v of each processor P_v is a distinct permutation of the symbols $\{1, 2, \ldots, n\}$. Processor P_v is directly connected by an edge to each of $n - 1$ processors P_u, where u is obtained by interchanging the first and ith symbols of v.

S_n is a regular graph with degree $n - 1$ and diameter $O(n)$, i.e., sub-logarithmic in the number of vertices, while hypercube has a degree and diameter logarithmic in the number of vertices. Figure 1.5 shows a 4-star.

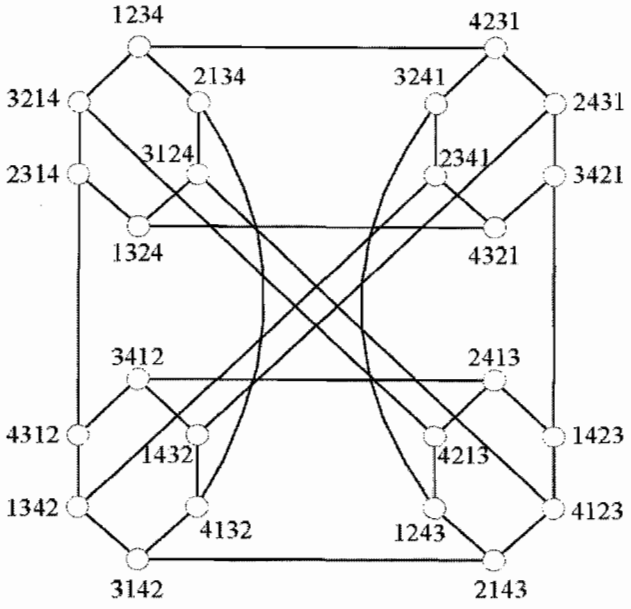

Figure 1.5: A 4-Star Graph

The star interconnection network is an attractive alternative to the hypercube parallel model [1]. The star graphs compare favorably with the hypercube regarding properties as symmetry properties, fault tolerance, etc. [2][12][31]. Many efficient algorithms have been designed for the star graphs as well [30][7][6].

1.4 The Hyper-Star Interconnection Network

For interconnection networks, we prefer small degree and diameter. But there is a trade-off between degree, related to hardware cost, and diameter, related to transmission time of messages. Thus, the network cost, defined as *degree* $\times$ *diameter* seems to be a reasonable measure for interconnection networks. And the ***hyper-star*** graph was first proposed in 2002 [26] as an interconnection network with small network cost. The hyper-star graph is formally defined as follows:

Definition 10 *Let* $HS(n,k) = (V(HS(n,k)), E(HS(n,k)))$ *be a* ***hyper-star*** *graph, where a node is represented by the string of* n *bits* $s_1 s_2 \dots s_i \dots s_n$, $s_i \in \{0,1\}$, *and* k $(n > k)$ *bits are "1". Two nodes* $u = s_1 s_2 \dots s_{i-1} s_i s_{i+1} \dots s_n$ *and* $v = s_i s_2 \dots s_{i-1} s_1 s_{i+1} \dots s_n$ *are connected by an edge* $(u,v) \in E(HS(n,k))$ *if and only if* s_i *is a complement of* s_1, *and the bit string of* v *is obtained by exchanging* s_1 *and* s_i, $2 \leq i \leq n$, *in the bit string of* u. *That is,* $HS(n,k)$ *is defined as*

$$V(HS(n,k)) = s_1 s_2 \dots s_i \dots s_n, s_i \in \{0,1\}, where\ |s_i = 1| = k,$$

$$E(HS(n,k)) = (u,v), where\ u = s_1 s_2 \dots s_i \dots s_n, v = s_i s_2 \dots s_1 \dots s_n, s_1 = \overline{s_i}$$

From the above definition, since nodes of $HS(n,k)$ can be represented by k-combinations of the set of n bit strings such that the number of 1's is k and the number of "0" is

$n-k$, the number of nodes in $HS(n,k)$ is $\begin{pmatrix} n \\ k \end{pmatrix} = \frac{n!}{k!(n-k)!}$. Figure 1.6 shows a hyper-star with $n=6$ and $k=3$.

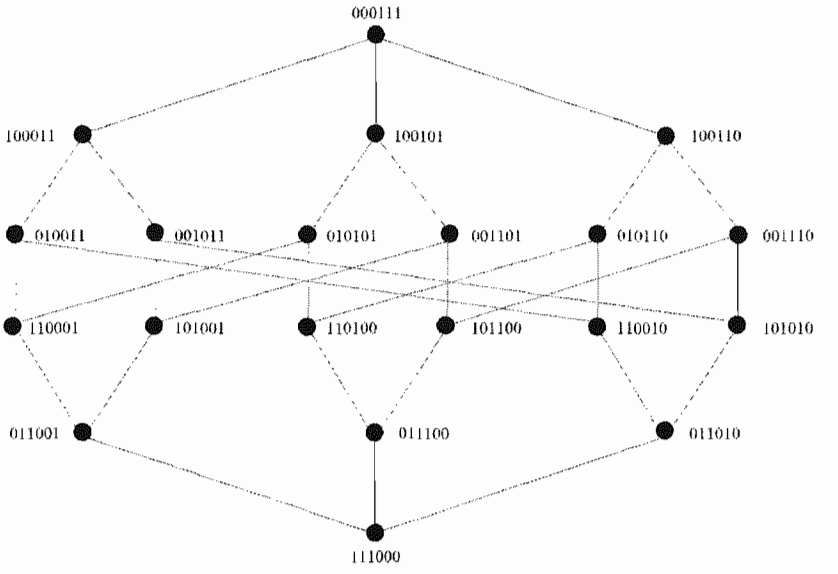

Figure 1.6: Hyper-Star Grpah $HS(6,3)$

1.5 Analysis of Parallel Algorithms

The most important three criteria we use to analyze a parallel algorithm are: **running time, the number of processors in the computational model** and **cost** [4]. The *running time* of a parallel algorithm is defined as the time taken by this algorithm to solve a problem on a parallel computer. Specifically, we are interested in the *worst* time, which means the time required for solving the most difficult instance of the problem using this algorithm. Usually, we count how many *elementary* steps are

performed by an algorithm when solving a problem (worst case) as a measure of running time. There are two different kinds of elementary steps:

1. *Computational steps*: A computational step is an arithmetic or logic operation performed on a datum within a processor, like adding two numbers.

2. *Routing steps*: A routing step takes place when a datum of constant size is transmitted from one processor to another processor via shared memory or an interconnection network.

Each step (computational or routing) takes a constant number of time units, and the running time of a parallel algorithm is a function of the size of input. For a problem of size N, we use $t(N)$ to denote the worst case number of time units required by the parallel algorithm.

We use $p(N)$ to denote the *number of processors* used by a parallel algorithm to solve a problem of size N. And the *cost* $c(N)$ of a parallel algorithm for a problem of size N is then defined as $c(N) = t(N) \times p(N)$. The cost of a parallel algorithm is an upper bound on the total number of elementary steps executed.

1.6 Organization of the Thesis

This thesis is organized as follows:

- **Chapter 1** gives an overall introduction to parallel computing, including the parallel computational models and parallel algorithms.

- **Chapter 2** is the literature review of the interconnection network this thesis is focused on, the hyper-star graph. This review is done in two aspects, the

topological properties of the graph, including degree, diameter, symmetry, scalability, and connectivity. The other aspect is the algorithms designed to run on a hyper-star network, which includes a shortest path routing algorithm and a broadcasting algorithm on single-port model.

- **Chapter 3** presents several properties of the hyper-star graph, which are the relation to other graphs, the surface area of the graph and the Hamiltonicity problem of the graph.

- **Chapter 4** focuses on the communication problems of the hyper-star graph. We restrict our discussion to the regular hyper-star graph $HS(2n, n)$. Two algorithms are designed for $HS(2n, n)$, namely the all-port model broadcasting algorithm and the single-port model neighbourhood broadcasting algorithm.

- **Chapter 5** briefly studies a variation of the hyper-star graph, which is called the folded hyper-star graph. We first review some work that has been done by other researchers, and then prove the maximum fault tolerance of the graph.

- **Chapter 6** concludes the thesis and list future open problems and research directions in this network.

Chapter 2

Literature Review of the Hyper-Star

2.1 Introduction

In this chapter we provide a literature review of the hyper-star graph. This review will cover some topological properties of the hyper-star graph, including scalability, connectivity, diameter, and symmetry properties. How to embed a ring and hypercube into the hyper-star is also reviewed. Then, a broadcasting scheme using a spanning tree is introduced. At last, we give a brief introduction to a variation of the hyper-star graph, namely the folded hyper-star graph.

2.2 Topological Properties

From the definition of the hyper-star graph, we can easily see that in $HS(n,k)$, a node has degree $n-k$ if the leftmost bit in its label is "1" and k if the leftmost bit is

"0". Thus, the hyper-star graph is irregular in general and only regular when $n = 2k$.

2.2.1 Scalability

By grouping the nodes in $HS(n,k)$ based on whether the least significant bit of its label is "1" or "0", we can decompose $HS(n,k)$ into two subgraphs $HS(n-1,k)$ and $HS(n-1,k-1)$. It has been proved that:

Proposition 1 *A hyper-star graph $HS(n,k)$ is isomorphic to a hyper-star graph $HS(n,n-k)$ [26].*

This means if we decompose a regular hyper-star graph $HS(2k,k)$, we can end up with two isomorphic subgraphs $HS(2k-1,k)$ and $HS(2k-1,k-1)$. A decomposition of $HS(6,3)$ is shown in Figure 2.1.

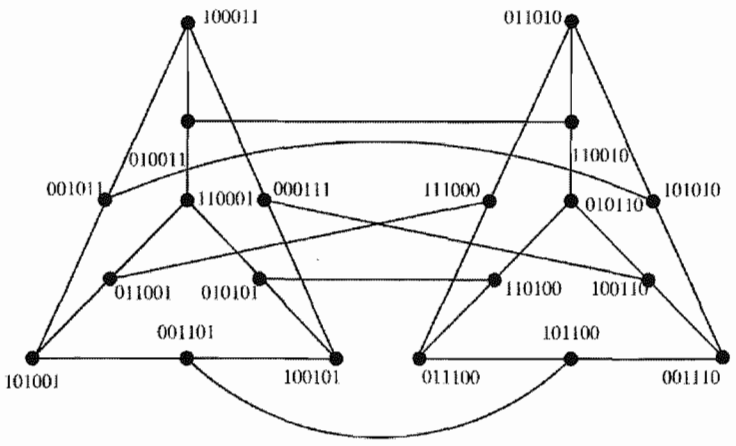

Figure 2.1: Decomposing $HS(6,3)$ into $HS(5,3)$ and $HS(5,2)$

2.2.2 Connectivity

In interconnection networks, node connectivity or edge connectivity is an important measurement to evaluate the network as still functional. For a graph G, let its node connectivity, edge connectivity, and degree be $\kappa(G)$, $\lambda(G)$, and $\delta(G)$, respectively, then it has been established that $\kappa(G) \leq \lambda(G) \leq \delta(G)$. It has been shown that the hyper-star graph $HS(2k, k)$ is maximally fault tolerant.

Proposition 2 *For a hyper-star graph $HS(2n, n)$, $\kappa(HS(2n, n)) = \lambda(HS(2n, n)) = n$ [26].*

Proof. Let P be a set of $n-1$ nodes that will be removed from the $HS(2n, n)$.

Case 1 The leftmost bit s_1 of all nodes ($\in P$) is "0"

If there is a node u in $HS(2n, n)$ that has all $n-1$ neighbours in P, there exists at least one non-faulty neighbour of u whose s_1 is "0". If some neighbours of u are not in P, it is easy to see that $HS(2n, n)$ is connected. Similarly, we can show the case where the first bit string s_1 of all nodes $\in P$ is "1".

Case 2 The leftmost bit s_1 of some nodes ($\in P$) is "1"

The degrees of all nodes in $HS(2n, n)$ are n and n nodes incident on a node u are connected to u if their first bit of labels are the complement of the first bit of u. Thus, if $n-1$ nodes in P are consisting of nodes having their first bit as "1" and nodes having their first bit as "0", $HS(2n, n)$ is always connected.

From the above discussion, we can see that the $HS(2n, n)$ is connected even if we remove at most $n-1$ nodes from it. It shows that $\kappa(HS(2n, n)) \geq n$. Also, since $HS(2n, n)$ is a regular graph of degree n, $\kappa(HS(2n, n)) \leq n$. Therefore, $\kappa(HS(2n, n)) = n$. Similarly, $\lambda(HS(2n, n)) = n$. □

2.2.3 Diameter

Let $d(HS(n,k))$ be the diameter of a hyper-star graph $HS(n,k)$.

Proposition 3 *The diameter of a hyper-star graph $HS(n,k)$ $(k < \frac{n}{2})$ is $n-1$ [26].*

Proof. Depending on k, we can have different diameters as follows.

$$d(HS(n,k)) = \begin{cases} 2 & \text{if } k = 1 \\ \text{n-1} & \text{if } k = \frac{n}{2} \\ \text{2k} & \text{if } k < \frac{n}{2} \end{cases}$$

All hyper-star graphs $HS(n,k)$ whose k is 1 are single stars. That is, all other nodes are incident on one node. Thus the diameter of $HS(n,k)$ is 2. Suppose for two nodes u and v in a regular hyper-star graph such that $k = \frac{n}{2}$, the number of bits these two nodes are different is at most n. And it takes $n-1$ steps to correct all the different bits. Thus, the diameter of $HS(n,\frac{n}{2})$ is $n-1$. For an irregular hyper-star graph such that $k < \frac{n}{2}$, the maximum number of different bits between two nodes is $2k$, which is less than $n-1$. From the above discussion, the diameter of a hyper-star graph $HS(n,k)$ is $n-1$. □

2.2.4 Symmetry Properties

For two nodes u and v in $HS(2k,k)$, a node w on a path from u to v is said to be in the *level* L_m if the distance between u and w is m. One major symmetry property has been proved, which is stated as follows.

Proposition 4 *A hyper-star graph $HS(2n,n)$ is node-symmetric [21].*

Consequently, from a node in $HS(2n,n)$, a subgraph consisting of nodes in L_i, $0 \leq i \leq n-1$, and a subgraph consisting of nodes in L_j, $n \leq j \leq 2n-1$, are symmetric.

2.3 Embedding Other Graphs into Hyper-Stars

Efficient embedding is another favorable property for an interconnection network. If a network A can be embedded in a network B, then all the algorithms developed for parallel processing with network A can be easily transported onto another network B.

Definition 11 *Let $G = (V, E)$ and $G' = (V', E')$ be graphs and $P(G')$ denote the set of paths in G'. We say (ϕ, ρ) is an* **embedding** *of G into G' if $\phi : V \to V'$ and $\rho : E \to P(G')$ are injective functions such that $(u, v) \in E$ if and only if $\rho(u, v)$ is a path between $\phi(u)$ and $\phi(v)$, the paths in $\rho(E)$ (that is, the image of ρ) are internally vertex-disjoint, and the set of internal vertices on paths in $\rho(E)$ and $\phi(V)$ are disjoint.*

Definition 12 *The* ***dilation*** *of edge e in graph G is the length of $\rho(e)$ and the* ***dilation*** *of the embedding is the maximum value among all the dilations of edges.*

Proposition 5 *Let $n \geq 2$. Then hypercube Q_n can be embedded into hyper-star $HS(2n, n)$ with dilation 2 [23].*

Proof. For each vertex $v \in V(Q_n)$, define $\phi(v) = v\overline{v}$. Then $\phi(v)$ is a binary string of length $2n$ with exactly n 1's. Hence $\phi : V(Q_n) \to V(HS(2n, n))$ is an injection. Now consider the edges of Q_n. Let $e \in E(Q_n)$. We have two cases:

The two adjacent vertices differ in the leftmost position. Then we may assume that they are of the form $0s$ and $1s$. Define $\rho(e)$ to be the path of length 1: $(0s1\overline{s}, 1s0\overline{s})$ in $HS(2n, n)$, where s is a binary string of length $n - 1$.

The two adjacent vertices do not differ in the first position. Then we may assume that they are of the form $as_1 0 s_2$ and $as_1 1 s_2$, where s_1, s_2 are (possibly empty) binary strings. If $a = 0$, define $\rho(e)$ to be the path of length 2: $(0s_1 0 s_2 1\overline{s_1} 1\overline{s_2}, 1s_1 0 s_2 1\overline{s_1} 0\overline{s_2}, 0s_1 1 s_2 1\overline{s_1} 0\overline{s_2})$. If $a = 1$, define $\rho(e)$ to be the path of length 2: $(1s_1 0 s_2 0\overline{s_1} 1\overline{s_2}, 0s_1 1 s_2 0\overline{s_1} 1\overline{s_2}, 1s_1 1 s_2 0\overline{s_1} 0\overline{s_2})$. Note that the internal vertex does not belong to $\phi(V(Q_n))$.

Then ρ is the required injective function. By construction, every vertex in Q_n is mapped to a vertex in $HS(2n, n)$ with the property that the first half of the string is the complement of the second half. Moreover, the vertex in the center of a length 2 path in the image of ρ has the following property: the first and the second halves of the string match up in exactly two positions including the first position. Hence it uniquely identifies the preimage of the path. Thus the image of ρ consists of internally vertex-disjoint paths. □

2.4 Routing Algorithm

A routing algorithm has been designed for the regular hyper-star graph $HS(2n, n)$ [26]. Let a node S be the source node and a node D be the destination node in a hyper-star graph $HS(2n, n)$, then the shortest path between S and D can be regarded as the process of changing the bit string of S to that of D. A routing scheme to construct the shortest path is as follows.

For two nodes $S = s_1 s_2 \dots s_n$ and $D = d_1 d_2 \dots d_n$, denote by $R = r_1 r_2 \dots r_n$ the bit string obtained by applying the $Exclusive - OR$ operation between S and D.

We use $\oplus$ to represent the Exclusive-OR operator. Each i-dimensional edge, which leads to two identical ith bits from bits where $r_i = 1$, belongs to a shortest path. A path is continuously extended, provided there is a bit '1' in R. Figure 2.2 shows the algorithm that constructs a shortest path between any two nodes. Let P_s be the set of nodes on a shortest path from S to D. Initially, $P_s = \{S\}$. In each iteration

```
Shortest_path_1(S, D, P_S)
begin
    if(S = D) then
        return P_s;
    obtain R = r_1 r_2 ··· r_i ··· r_n , where r_i = s_i ⊕ d_i, 1 ≤ i ≤ n;
    let p = {i | r_i = 1, 2 ≤ i ≤ n} and q = {j | (S, S') ∈ E(HS(n, k)), 2 ≤ j ≤ n},
        where jth bits in S and S' complement;
    if (| p |> 0) then
        find a node S' such that (S, S') is an i-dimensional edge, where
            i = min{p ∩ q};
        P_s = P_s ∪ {S'};
        S = S';
    call Shortest_path_1(S, D, P_S);
end
```

Figure 2.2: The Shortest Path Routing Algorithm for Hyper-Star

satisfying the condition $r_i = 1$, one bit from the source node S is fixed to be identical to its corresponding bit in the destination node D, and the number of different bits between S and D is reduced exactly by 1. Thus, it is easy to see that the algorithm is optimal in terms of the length of a path.

Chapter 3

Properties of the Hyper-Star Graph

3.1 Relation to Other Graphs

In this section we introduce two graphs that are closely related to the hyper-star graph, namely the odd graphs and the even graphs. We also present their relations to the hyper-star graphs. These relation properties were proved by Jong-Seok Kim, Eddie Cheng, Laszlo Liptak and Hyeong-Ok Lee in 2008 [23], and then rediscovered independently by us in the same year.

3.1.1 Odd Graph

3.1.1.1 Introduction

The class of odd graphs was first introduced in the context of graph theory, and then studied as a potential interconncetion network in 1991 [18]. The odd graphs are

formally defined as follows:

Definition 13 *Let n be an odd integer and $n = 2\delta - 1$, $\delta \geq 2$. The vertices of odd graph O_δ are the binary stings of length n with exactly δ 1's. Two vertices are adjacent if and only if their Hamming distance is $2\delta - 2$.*

That is, two vertices are connected if and only if they differ in all but one bit. Figure 3.1 shows the odd graph O_3. Below are some of the important topological prop-

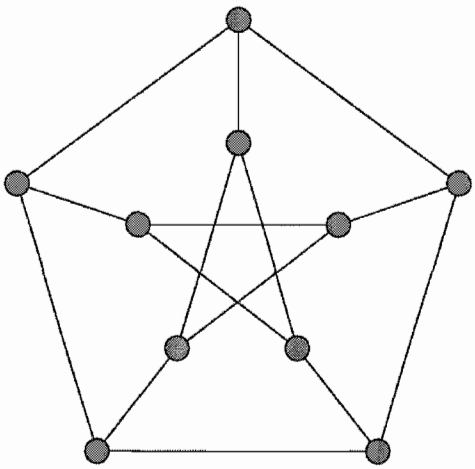

Figure 3.1: Odd Graph with Degree 3 (Petersen Graph)

erties of these graphs.

Proposition 6 *An O_δ graph is regular with $N = \begin{pmatrix} 2\delta - 1 \\ \delta \end{pmatrix}$ nodes, degree δ, and diameter $\delta - 1$.*

Proposition 7 *The degree and the diameter of O_δ are of asymptotic order $log_2\sqrt{N}$, where N is the number of nodes in the network.*

3.1.1.2 Relation to the Hyper-Star

Since the odd graph O_δ has the same number of vertices as the hyper-star graph $HS(2\delta-1,\delta)$, and their representation strings for the vertices are all the same, we study the two graphs together hoping to establish a relationship between them.

In the odd graph O_δ, a vertex u with the first bit being '1' is in the form of $1b_1b_2\ldots b_i\ldots b_{2\delta-2}$. It has only one neighbour whose first bit of its representation string is '1', and $\delta-1$ neighbours with '0' as their first bits. These $\delta-1$ neighbours are in the form of $0\overline{b_1b_2}\ldots b_i\ldots\overline{b_{2\delta-2}}$, where $2\leq i\leq 2\delta-2$ and $b_i=1$. If we take the complement of $\overline{b_1b_2}\ldots b_i\ldots\overline{b_{2\delta-2}}$, we get $b_1b_2\ldots\overline{b_i}\ldots b_{2\delta-2}$. We observe that $0b_1b_2\ldots\overline{b_i}\ldots b_{2\delta-2}$, $2\leq i\leq 2\delta-2$ and $b_i=1$ are the $\delta-1$ neighbours of $1b_1b_2\ldots b_i\ldots b_{2\delta-2}$ in the hyper-star graph $HS(2\delta-1,\delta)$.

A vertex u with the first bit '0' in O_δ has δ neighbours all in the form of 1α. As stated above, $1\overline{\alpha}$ is a vertex in $HS(2\delta-1,\delta)$, and it is a neighbour of the node in the hyper-star graph with the same representation string as vertex u. Thus, we have the following theorem:

Theorem 1 *The hyper-star graph $HS(2\delta-1,\delta)$, $\delta\geq 1$, can be constructed from the odd graph O_δ by removing all the edges between vertices whose first bits are '1's.*

Immediately we get:

Theorem 2 *The hyper-star graph $HS(2\delta-1,\delta)$ is a spanning graph of the odd graph O_δ, where $\delta\geq 1$.*

3.1.2 Even Graph

3.1.2.1 Introduction

The even graph was first introduced as a class of efficient interconnection networks in 1989 [17]. It possesses some attractive characteristics such as capability of maximal fault-tolerance, higher density, admitting simple distributed-routing algorithms both for the faulty and fault-free network, and ease of self-diagnosis. The formal definition of the even graph is given as follows.

Definition 14 *Let n be an odd integer of the form $n = (2\delta - 3)$ for $\delta > 1$. Let W_1, W_2 be two sets of binary strings of length n, such that the Hamming weight of every string in W_1 is $\delta - 1$ and the Hamming weight of every string in W_2 is $\delta - 2$. Then the vertex set V of the even graph E_δ is $V = W_1 \bigcup W_2$, the edge set $E = \{(x, y) | x \in W_1, y \in W_2, h(x, y) = 1 \text{ or } n\}$.*

That is, the vertices of an even graph are binary strings of length $2\delta - 3$, all these strings have either $\delta - 1$ 1's or $\delta - 2$ 1's. Two vertices are connected if they differ in one bit or they are complementary to each other. Figure 3.2 shows an even graph with 20 vertices. Some topological properties have been studied and proved, which are given in the following two propositions.

Proposition 8 *An E_δ network is regular with degree δ and has* $\begin{pmatrix} 2\delta - 2 \\ \delta - 1 \end{pmatrix}$ *nodes. Its diameter is $\delta - 1$.*

Proposition 9 *The degree and the diameter of E_δ is proportional to $log_2\sqrt{N}$, where N is the number of nodes in the network.*

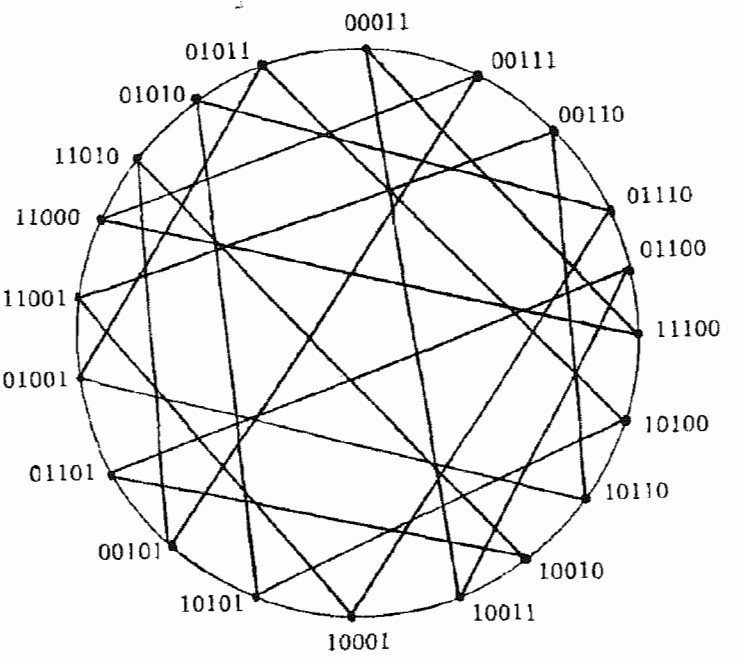

Figure 3.2: An Even Graph E_4 with 20 Vertices, Degree 4, and Diameter 3

3.1.2.2 Relation to the Hyper-Star

Theorem 3 *The even graph E_δ, $\delta \geq 2$, is isomorphic to the folded hyper-star graph $FHS(2\delta - 2, \delta - 1)$.*

Proof. Define a mapping $\phi : V(E_\delta) \rightarrow V(FHS(2\delta - 2, \delta - 1))$ as follows. If α is a vertex of E_δ, and $\alpha \in W_1$, (the Hamming weight of α is $\delta - 1$), then define $\phi(\alpha) = 0\alpha$; If β is a vertex of E_δ, and $\beta \in W_2$, (the Hamming weight of β is $\delta - 2$), then define $\phi(\beta) = 1\beta$. Now consider α, a vertex of E_δ, $\alpha \in W_1$. Then the neighbour of α must be in the form of β, $\beta \in W_2$. If α and β are adjacent, then we have two cases:

1. The Hamming distance between α ane β is 1, they differ in one bit (say position p). Furthermore, that different bit is 1 in α, and is 0 in β. Therefore, 0α and 1β both have $\delta - 1$ 1's and $\delta - 1$ 0's, which means they are both vertices in

$FHS(2\delta-2,\delta-1)$. And 1β can be obtained from 0α by exchanging the first '0' with the p'th bit (1). Thus, 0α and 1β are adjacent in $FHS(2\delta-2,\delta-1)$;

2. The Hamming distance between α ane β is $2\delta-3$, they are complementary to each other. Thus 0α and 1β are still complementary to each other. Which means they are adjacent in $FHS(2\delta-2,\delta-1)$.

Hence the proof is complete. □

Since the hyper-star graph can be constructed by removing all the edges in the corresponding folded hyper-star that connect two vertices whose binary strings are complementary to each other, the theorem below immediately follows.

Theorem 4 *The hyper-star graph $HS(2\delta-2,\delta-1)$ is a spanning graph of the even graph E_δ, where $\delta \geq 2$.*

3.2 Surface Area

Definition 15 *The quantity of $|\{v|d(u,v)=k\}|$, where $d(u,v)$ stands for the distance between nodes u and v, is referred as* ***Whitney numbers of the second kind of the poset*** *[28], or* ***surface area of a node with radius*** *k [20].*

This quantity of surface area is especially well defined for the node symmetric graphs, as the surface area for any node in a node symmetric graph G equals that for any other node in G. We can thus discuss the *surface area of such a graph G.*

In this section, we derive a formula for the regular hyper-star graph $HS(2n,n)$. Since $HS(2n,n)$ is node-symmetric, we focus on the surface area of one specific node, in

this case, the *identity node*, $I = 0^n1^n$.

A regular hyper-star graph $HS(2n, n)$ is an undirected graph consisting of $\begin{pmatrix} 2n \\ n \end{pmatrix}$ nodes, where a node is represented by the string of $2n$ bits $s_1 s_2 \ldots s_{2n}$ such that the cardinality of the set $\{i | 1 \leq i \leq 2n, s_i = 1\}$ is n.

Definition 16 *Let σ_i be an operation that exchanges s_1 and s_i, $2 \leq i \leq 2n$, where s_i is the complement of s_1. Then two nodes u and v are connected when v is obtained from the operation $\sigma_i(u)$, $2 \leq i \leq 2n$.*

Definition 17 *For a node u, we denote by $[k_1, k_2, \ldots, k_t]$ a* **path** *obtained by applying operations $\sigma_{k_1}, \sigma_{k_2}, \ldots, \sigma_{k_t}$ in sequence from the node u.*

For example, there is a path $[3, 2, 4]$ or $[4, 2, 3]$ from 0011 to 1100.

Since a shortest path from I can be constructed by applying unique operations σ_i, $n+1 \leq i \leq 2n$, and σ_j, $2 \leq j \leq n$, alternately, we have the following lemma.

Lemma 1 *Any shortest path $[k_1, k_2, \ldots, k_t]$ from the identity node I to a specific node in regular hyper-star graph $HS(2n, n)$ has the same set of numbers $(k_1, k_3, \ldots, k_i, \ldots)$ (i is odd) and $(k_2, k_4, \ldots, k_j, \ldots)$ (j is even).*

We refer to the set $(k_1, k_3, \ldots, k_i, \ldots)$ (i is odd) as S_{odd}, and the set $(k_2, k_4, \ldots, k_j, \ldots)$ (j is even) as S_{even}. Notice here that every number in set S_{odd} is between $n+1$ and $2n$, and every number in set S_{even} is between 2 and n.

Thus we can see that the number of nodes of a certain distance from I is determined by the number of different combinations of S_{odd} and S_{even}. This leads us to the following theorem.

Theorem 5 *The surface area of regular hyper-star $HS(2n, n)$ is:*

$$N_i = \frac{n(n-1)^2(n-2)^2 \ldots (n - \lfloor \frac{i}{2} \rfloor)^2}{P(\lfloor \frac{i}{2} \rfloor) P(\lceil \frac{i}{2} \rceil)}$$

where $P(n)$ is the number of permutations on n distinct symbols.

Table 3.1 shows the surface area of the hyper-star $HS(2n, n)$ for $n = 2$ to $n = 6$.

Table 3.1: Surface Area of Hyper-Star $HS(2n, n)$

Radius	$HS(4,2)$	$HS(6,3)$	$HS(8,4)$	$HS(10,5)$	$HS(12,6)$
1	2	3	4	5	6
2	2	6	12	20	30
3	1	6	18	40	75
4		3	18	60	150
5		1	12	60	200
6			4	40	200
7			1	20	150
8				5	75
9				1	30
10					6
11					1

3.3 Hamiltonicity

The Hamiltonicity problem is yet another important problem often studied for interconnection networks.

Definition 18 *Suppose that W is an interconnection network. A path (or cycle) in W is called a* **Hamiltonian path** *(or* **Hmiltonian cycle***) if it contains every node of W exactly once. W is called* **Hamiltonian** *if there is a Hamiltonian cycle in W.*

3.3.1 The Middle Cube and the Revolving Door Conjecture

The interconnection network middle cube is defined as:

Definition 19 *Let Q_n be the n-dimensional hypercube. If $n = 2k + 1$, then the subgraph M_n of Q_n induced by the nodes having exactly k or $k + 1$ 1's is called the* ***middle cube*** *of dimension n.*

The highlighted part in Figure 3.3 shows a middle cube M_3 in the corresponding hypercube Q_3. The middle cube is first studied as a potential interconnection network for parallel computation in 1990 [27], there is one well-known conjecture concerning the middle cube, namely the ***Revolving Door Conjecture***, which is stated as follows:

Conjecture 1 *All middle cubes M_n, $n > 1$, are Hamiltonian.*

This conjecture has been verified for $n \leq 17$, but is still open in general.

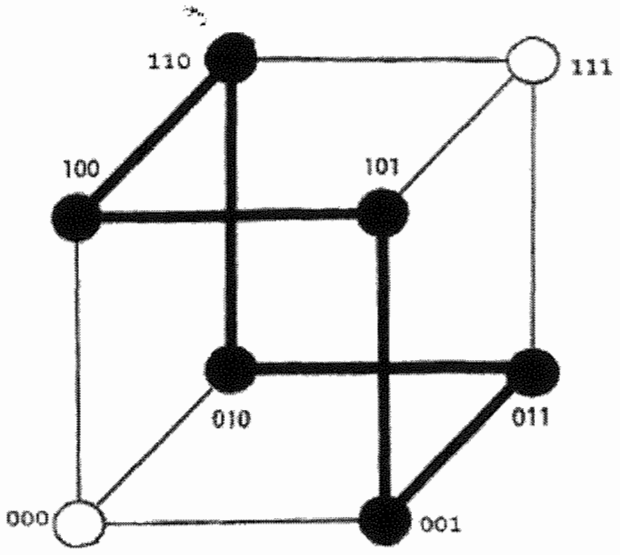

Figure 3.3: Middle Cube M_3 in Hypercube Q_3

3.3.2 Middle Cube and Hyper-Star

In the course of tryng to solve the Hamiltonicity problem of the hyper-star graph, we try to establish a relationship between the hyper-star and the middle cube due to the resemblance between their representation strings. And this leads us to the following theorem:

Theorem 6 *The hyper-star graph* $HS(2k+2, k+1)$ *is isomorphic to the middle cube* M_{2k+1}, $k \geq 0$.

Proof. Define a mapping $\phi : V(M_{2k+1}) \rightarrow V(HS(2k+2, k+1))$ as follows. If α is a vertex of M_{2k+1}, and the Hamming weight of α is $k+1$, then define $\phi(\alpha) = 0\alpha$; If β is a vertex of $M_(2k+1)$, and the Hamming weight of β is k, then define $\phi(\beta) = 1\beta$. Now consider α, a vertex of M_{2k+1}, $H(\alpha) = k+1$. Then the neighbours of α must be in the form of β, $H(\beta) = k$. And α and β differ in exactly one bit, say bit position p. Futhermore, bit p of α is 1, bit p of β is 0. Then, 0α and 1β differ in two bits,

the first bit of course, and the bit at position $p+1$. The bit at position $p+1$ is 1 in 0α and is 0 in 1β. This means 1β can be obtained from 0α by exchanging the first bit with the $p+1$'th bit, which implies $\phi(\alpha)$ and $\phi(\beta)$ are two adjacent nodes in the hyper-star graph $HS(2k+2, k+1)$. Hence the proof is complete. □

By this theorem, we claim the Hamiltonicity problem of the hyper-star graph is as hard as the Hamiltonicity problem of the middle cube graph, and hence the following conjecture:

Conjecture 2 *All regular hyper-star graphs $HS(2k, k)$, $k \geq 1$, are Hamiltonian.*

3.3.3 Tracing Hamiltonian Cycles in the Hyper-Star

As established in the previous subsection, proving the hyper-star graphs are Hamiltonian in general is a hard problem despite the fact that several low dimensional regular hyper-stars are proved to be Hamiltonian. In this subsection, we provide a stategy for finding the Hamiltonian cycles in the hyper-star graphs. This technique was first used in 1990 to trace the Hamiltonian cycles in modified even networks [27].

Let N be the number of nodes in the hyper-star graph $HS(2n, n)$. Let $\Sigma = \{0, 1\}$ and Σ^n denote the set of all binary strings of length n. Define the operation $\oplus$ to be the *bitwise* OR for two binary strings. Define two sets W_0 and W_1 as: $W_0 = \{x \mid x = 0\alpha, \alpha \in \Sigma^{2n-1}, H(\alpha) = n\}$; $W_1 = \{y \mid y = 1\beta, \beta \in \Sigma^{2n-1}, H(\beta) = n-1\}$. That is, $W_0 \bigcup W_1$ contains the binary strings of all the nodes in $HS(2n, n)$. Then define a traversal S_i of the set W_i, $i = 0, 1$ as follows: $S_0 = \{x_0, x_1, \ldots, x_{N/2-2}, x_{N/2-1}\}$, $S_1 = \{y_0, y_1, \ldots, y_{N/2-2}, y_{N/2-1}\}$ such that $x_i = 0\alpha_i$, $x_i \in W_0$ and $y_i = 1\beta_i$, $y_i \in W_1$

for all $0 \leq i \leq N/2 - 1$ and

$$h(x_i, x_{(i+1) \bmod N/2}) = 2,\ 0 \leq i \leq N/2 - 1 \tag{3.1}$$

$$h(y_i, y_{(i+1) \bmod N/2}) = 2,\ 0 \leq i \leq N/2 - 1 \tag{3.2}$$

$$\alpha_i = \beta_i \oplus \beta_{(i+1) \bmod N/2},\ 0 \leq i \leq N/2 - 1 \tag{3.3}$$

and

$$(x_i, y_i), (x_i, y_{(i+1) \bmod N/2}) \in E(HS(2n, n)) \tag{3.4}$$

The Hamiltonian cycle in $HS(2n, n)$ is then given by the sequence

$$\{y_0, x_0, y_1, x_1, \ldots, y_{N/2-2}, x_{N/2-2}, y_{N/2-1}, x_{N/2-1}\} \tag{3.5}$$

Because $W_0 \bigcap W_1 = \emptyset$, $W_0 \bigcup W_1 = V(HS(2n, n))$, if condition 3.4 is satisfied, then sequence 3.5 is indeed one Hamiltonian cycle in the hyper-star graph $HS(2n, n)$. Conditions 3.1 and 3.2 are necessary because the Hamming distance between two nodes in a hyper-star graph equals the graphical distance between them.

For condition 3.3, since $h(y_i, y_{(i+1) \bmod N/2}) = 2,\ 0 \leq i \leq N/2 - 1$, the Hamming distance between β_i and $\beta_{(i+1) \bmod N/2}$ is also 2, which means two bits are different, say bits at position p_1 and p_2. Also, $H(\beta_i) = H(\beta_{(i+1) \bmod N/2}) = n - 1$, bits p_1 and p_2 in y_i and $y_{(i+1) \bmod N/2}$ have to be $(0, 1)$, $(1, 0)$ or $(1, 0)$, $(0, 1)$. Either way, after applying bitwise OR for y_i and $y_{(i+1) \bmod N/2}$, p_1 and p_2 in α_i are both 1. And all other bits in α_i are equal to those in β_i and $\beta_{(i+1) \bmod N/2}$, this is necessary for (x_i and y_i, x_i and $y_{(i+1) \bmod N/2}$ to be connected (y_i, $y_{(i+1) \bmod N/2}$ can be obtained from x_i by exchanging the first bit 0 in x_i with the bit p_1 or p_2).

This strategy gives us a way of finding Hamiltonian cycles in the hyper-star graph $HS(2n, n)$, which is, if we can find the traversal S_1 that meets condition 3.2, then

we can compute the corresponding traversal S_0 using 3.3, and the resulting S_0 is guaranteed to satisfy condition 3.1, hence we get one Hamiltonian cycle given by sequence 3.5.

Another observation is, all the nodes in the set W_1 have '1' as their first bit, so for $y_i \in W_1$, $0 \leq i \leq N/2-1$ and $y_i = 0\beta_i$, $h(y_j, y_k) = h(\beta_j, \beta_k)$, $0 \leq j, k \leq N/2-1$. Further, all β_i, for $0 \leq i \leq N/2-1$ are the nodes of the odd graph O_n.

This observation gives us one convenient way of finding the traversal S_1 because odd graphs O_2 and O_4 to O_8 are known to be Hamiltonian. If we have a Hamiltonian cycle $\{\beta_0, \beta_1, \ldots, \beta_{N/2-2}, \beta_{N/2-1}\}$ in O_n, then S_1 is given by

$$\{1\beta_0, 1\beta_2, 1\beta_4, \ldots, 1\beta_{N/2-1}, 1\beta_1, 1\beta_3, \ldots, 1\beta N/2-2\}$$

What is worth mentioning is that the use of Hamiltonian cycles in the odd graphs is only a convenience but not necessary, for example, it is well known that the odd graph O_3 is not Hamiltonian, but the hyper-star graph $HS(6,3)$ is Hamiltonian.

The Hamiltonian cycle in hyper-star $HS(8,4)$, using this approach, is given in Table 3.2.

Table 3.2: One Hamiltonian Cycle in $HS(8,4)$

10000111	01000111	11000110	01001110
11001010	01101010	11101000	01101001
10101001	00101101	10001101	01001101
11000101	01100101	10100101	00100111
10100011	00110011	10010011	01010011
11000011	01001011	11001001	01011001
11010001	01010101	11010100	01110100
11100100	01100110	11100010	01100011
11100001	01110001	11110000	01111000
10111000	00111001	10110001	00110101
10010101	00010111	10010110	01010110
11010010	01110010	10110010	00111010
10011010	01011010	11011000	01011100
11001100	01101100	10101100	00111100
10110100	00110110	10100110	00101110
10101010	00101011	10001011	00011011
10011001	00011101	10011100	00011110
10001110	00001111		

Chapter 4

Communication Problems on the Hyper-Star

Since the processors in an interconnection network exchange information via direct links, the underlying graph of the network plays a major role in the communication aspects of the network. Many communication problems have been introduced and studied for various interconnection networks, such as routing, neighbourhood broadcasting, broadcasting, gossiping... [8][22].

In this chapter, we design two algorithms for the regular hyper-star graph $HS(2n, n)$, the broadcasting algorithm on all-port model, and the neighbourhood broadcasting algorithm on single-port model. Our algorithms are both asymptotically optimal.

4.1 Broadcasting

4.1.1 Introduction

The ***broadcasting problem*** (***BP***) is the problem of disseminating a message from the source node to all the nodes in the network. For a single-port model, the lower bound for BP is given in the following theorem.

Theorem 7 *Any broadcasting algorithm of a network on single-port model with N nodes must require $\Omega(log\ N)$ time.*

Proof. At each time unit, one processor with the message can only send it to one of its neighbours, so after every step, the number of nodes which have received the message can at most double. There are N nodes in the network, thus the least time needed to solve BP is $\Omega(log\ N)$. □

When considering the broadcasting problem of a network on all-port model, the least time required is bounded by the diameter of the graph, because that is the least time required for the message to traverse from one node to another node that is the farthest away from it.

On all-port model, in addition to the time (the number of communication steps) required, one of the other criteria of the algorithm is the ***traffic***, i.e., the total number of messages exchanged. This means that it is desirable to minimize both the time and traffic. Minimizing the traffic is equivalent to minimizing the redundancy, i.e., the number of times a node receives the same message.

4.1.2 Broadcasting on the All-Port Model $HS(2n, n)$

For our all-port broadcasting algorithm on $HS(2n, n)$, we use a greedy approach in the sense that at step i, the leftmost node on level i will inform as many nodes as it can without introducing redundancy. And every node to the right will do the same thing, which is to inform as many nodes as they can without introducing redundancy. This approach will guarantee that every node on the next level will be informed, and only informed exactly once.

Figure 4.1 shows one example of how the all-port broadcasting is done on $HS(8, 4)$. All the nodes on the same level are informed in the same step due to the communication capability of the all-port model.

The number of steps this algorithm takes is equal to the diameter of the graph, which is optimal.

4.2 Neighbourhood Broadcasting

4.2.1 Introduction

The ***neighborhood broadcasting problem*** (***NBP***, for short) is the problem of disseminating a message from the source node to all the nodes adjacent to the source node. The neighborhood broadcasting problem was first introduced in 1991 [10] as a tool to simulate a single step of the all-port model by the single-port model in a given network.

On the all-port model, a node can communicate with all of its neighbours at the same

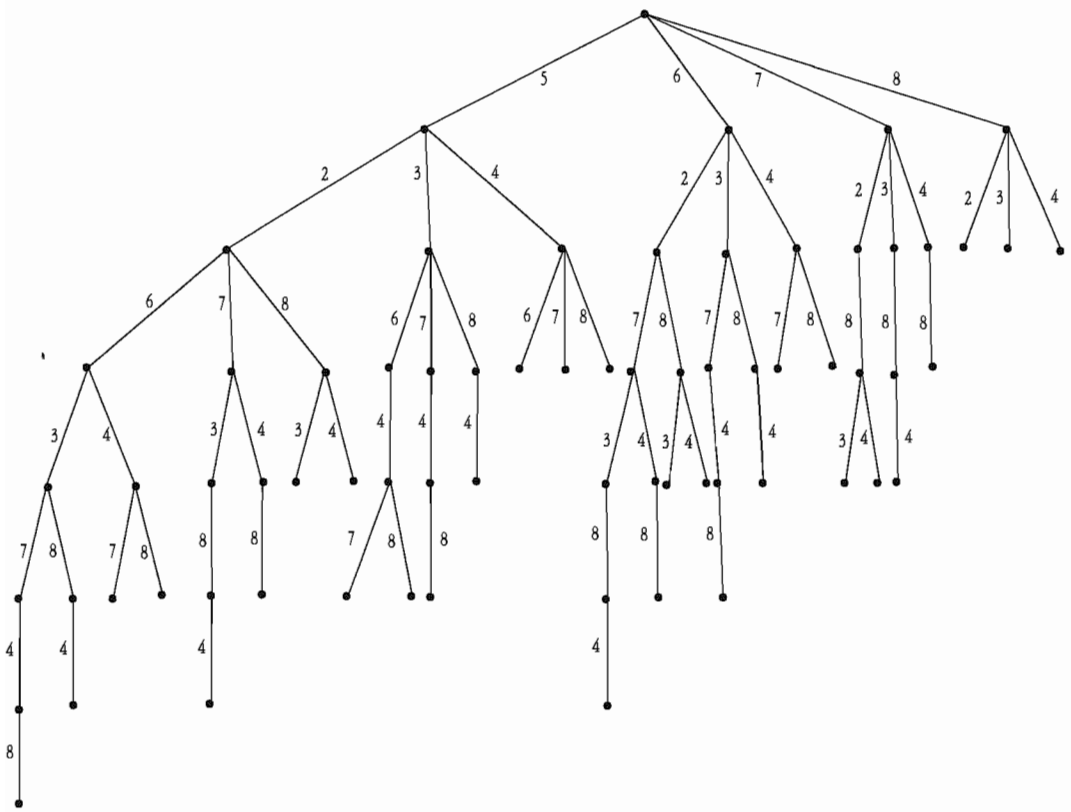

Figure 4.1: All-Port Broadcasting on $HS(8,4)$

time. In this case, NBP becomes a trivial problem and takes 1 time unit. From now on, we consider NBP on single-port model. On the single-port model, a node can only communicate with one of its neighbours at a time. The lower bound of NBP is stated in the following theorem.

Theorem 8 *Any neighbourhood broadcasting algorithm on a network with degree d must require* $\Omega(log\ d)$ *time.*

Proof. At each time unit, one processor with the message can only send it to one of its neighbours, so after every step, the number of neighbours which have received the message can at most double. The maximum number of neighbours of a node in the network is d, thus the least time needed to solve NBP is $\Omega(log\ d)$ $\square$.

4.2.2 Neighbourhood Broadcasting on the Single-Port Model $HS(2n, n)$

Being a regular graph, $HS(2n, n)$ has a degree of n. Thus the lower bound for NBP on $HS(2n, n)$ is $\Omega(log\ n)$. Also, as proved in the previous chapter, the regular hyper-star graphs are node-symmetric. Without loss of generality, we assume the source node for NBP is the identity node $I = 0^n1^n$.

We develop our algorithm for neighbourhood broadcasting problem on $HS(2n, n)$ based on several observations. First we give some definitions.

Definition 20 *A* ***bipartite graph*** *(or* ***bigraph****) is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to*

one in V; that is, U and V are independent sets. Equivalently, a bipartite graph is a graph that does not contain any odd-length cycles. If $|U| = |V|$, then it is called a ***balanced bipartite graph.***

Then we have:

Theorem 9 *The regular hyper-star graph $HS(2n, n)$ is a balanced bipartite graph.*

Proof. We partition the nodes of $HS(2n, n)$ into two sets U and V. U contains all the nodes whose first bit is 1, and V contains all the nodes whose first bit is 0. Apparently every edge of $HS(2n, n)$ connects a node in U to one in V and $|U| = |V| = \begin{pmatrix} 2n-1 \\ n \end{pmatrix}$, thus $HS(2n, n)$ is a balanced bipartite graph. □

Figure 4.2 shows the hyper-star graph $HS(6, 3)$, its nodes are arranged in such a way that nodes on the left all have first bit 1 and those on the right all have first bit 0.

We will use the notations introduced in last chapter, which is for a node u, we denote by $[k_1, k_2, \ldots, k_t]$ a path obtained by applying operations $\sigma_{k_1}, \sigma_{k_2}, \ldots, \sigma_{k_t}$ in sequence from the node u.

Theorem 10 *The smallest cycle in $HS(2n, n)$ has length* 6.

Proof. Because $HS(2n, n)$ is a bipartite graph, it cannot contain odd cycles. If there is a cycle of length 4 in $HS(2n, n)$, then it must be in the form of $[i, j, i, j]$ because operation σ_i flips the ith bit of the node. Assume the first bit of our source node is 0, then the ith bit must be 1, the jth bit must be 0. After applying operation σ_i, σ_j

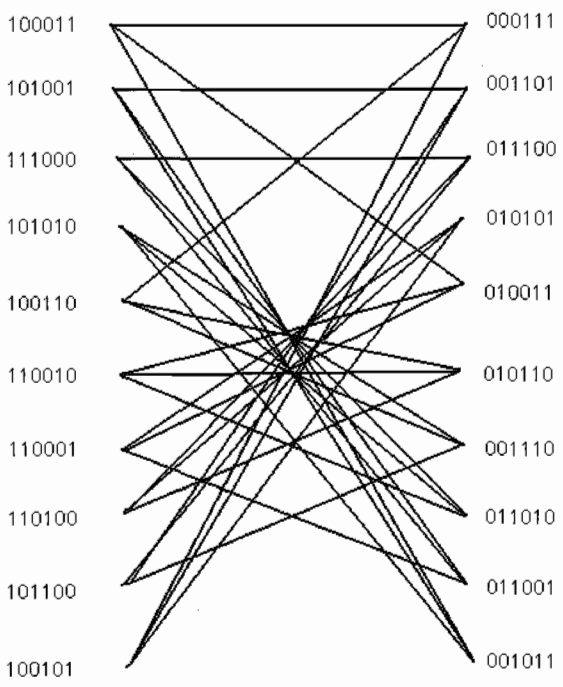

Figure 4.2: $HS(2n, n)$ are Balanced Bipartite Graphs

to the source node, we end up with a node with first bit 0, ith bit 0, jth bit 1. So for the next operation, we cannot apply σ_i, which is a contradiction. Thus there is no cycle of length 4 in $HS(2n, n)$. Our next choice is cycle of length 6, and this kind of cycles do exist, for example, in $HS(8, 4)$, we have:

$$00001111 \leftrightarrow 10000111 \leftrightarrow 01000111 \leftrightarrow 11000110$$

$$\leftrightarrow 01001110 \leftrightarrow 10001110 \leftrightarrow 00001111$$

Thus completes the proof. □

Our next theorem is crucial to our algorithm.

Theorem 11 *Any pair of neighbours of the source node 0^n1^n is connected by a path of length 4, and these paths between different pairs of neighbours are node-disjoint.*

Proof. Any neighbour of the source node 0^n1^n is obtained from exchanging the first 0 of the source node with another 1 of the node. Thus any two neighbours of it are different in exactly 2 bits at position p and q, where $n + 1 \leq p, q \leq 2n$. Then these two neighbours are connected by the path $[i, q, p, i]$, where $2 \leq i \leq n$. And these two neighbours and the source node are on the same cycle $[p, i, q, p, i, q]$, because two different pairs of neighbours have different p and q, these cycles are bound to be node-disjoint except they share the source node 0^n1^n. Thus the 4-length paths connecting pairs of neighbours are node-disjoint. □

The above theorem allows us to view the source node together with its n neighbours as a de facto complete graph in the sense that any two nodes are connected by a path

of constant length.

Based on the technique of ***recursive doubling*** where at each step, we double the number of neighbours with the message by using a set of node-disjoint paths with constant length between neighbours, a simple neighbourhood braodcasting algorithm for $HS(2n, n)$ can be designed. We denote the neighbour obtained by switching the first bit of the source node with the $n+i$'th bit, $1 \leq i \leq n$, the i'th neighbour of the source node.

- Initially, only the source node 0^n1^n in $HS(2n, n)$ has the message to be sent to all of its neighbours.
- For the first step, the source node sends the message to its first neighbour via direct link.
- At step i, $i \geq 2$, the source node sends the message to its 2^{i-1}'th neighbour. If neighbour j has the message, then at this step, it sends the message to neighbour $j + 2^{i-1}$ via the 4-length path $[2, j + 2^{i-1}, j, 2]$.
- The process continues until all the neighbours of the source node have the message.

For example, in $HS(14, 7)$, the neighbourhood broadcasting is done in the following pattern.

- **Step1**:

$$(S \rightarrow 1)00000001111111 \rightarrow 10000000111111$$

- **Step2**:

$$(S \rightarrow 2)00000001111111 \rightarrow 10000001011111$$

$$(1 \rightarrow 3)10000000111111 \rightarrow 01000000111111 \rightarrow 11000000101111$$

$$\rightarrow 01000001101111 \rightarrow 10000001101111$$

- **Step3**:

$$(S \rightarrow 4)00000001111111 \rightarrow 10000001110111$$

$$(1 \rightarrow 5)10000000111111 \rightarrow 01000000111111 \rightarrow 11000000111011$$

$$\rightarrow 01000001111011 \rightarrow 10000001111011$$

$$(2 \rightarrow 6)10000001011111 \rightarrow 01000001011111 \rightarrow 11000001011101$$

$$\rightarrow 01000001111101 \rightarrow 10000001111101$$

$$(3 \rightarrow 7)10000001101111 \rightarrow 01000001101111 \rightarrow 11000001101110$$

$$\rightarrow 01000001111110 \rightarrow 1000000111110$$

Chapter 5

Folded Hyper-Star

5.1 Introduction

The formal definition of the folded hyper-star graph is [26]:

Definition 21 *A folded hyper-star graph $FHS(2n, n)$ is a graph where edges are added to connect any two nodes whose bit strings are complements in the hyper-star graph $HS(2n, n)$. That is, a folded hyper-star graph $FHS(2n, n)$ is defined as*

$$V(FHS(2n, n)) = V(HS(2n, n))$$

$$E(FHS(2n, n)) = E(HS(2n, n)) \cup e,$$

where

$$e = \{(u, v) | u = s_1 s_2 \dots s_{2n}, v = \overline{s_1 s_2 \dots s_{2n}}\}$$

An edge in the set of edges e is called a ***c-edge***. Figure 5.1 shows a folded hyper-star graph $FHS(4, 2)$.

Simillarly, a routing algorithm was designed for the folded hyper-star [26]. Let

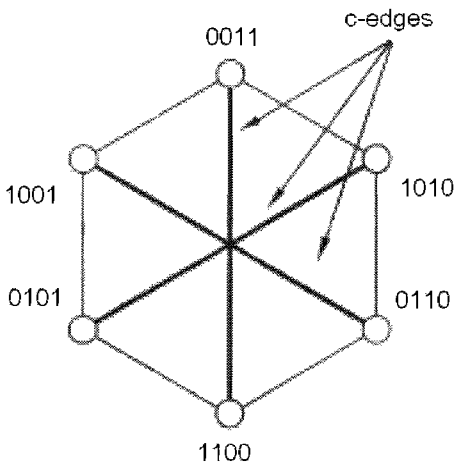

Figure 5.1: $FHS(4,2)$ Graph

$R = r_1 r_2 \ldots r_{2n}$ be a bit string obtained by applying Exclusive-OR operations between the source node S and the destination node D. Also, let M be the set of nodes whose number of 1's is greater than n. When a node belongs to the set M, the routing scheme for a folded hyper-star graph $FHS(2n,n)$ chooses a node first such as to utilize c-edges efficiently. The set of nodes P_s on a shortest path from S to D is constructed by the algorithm shown in Figure 5.2. Initially, $P_s = \{S\}$. We can easily see that the number of nodes in $FHS(2n,n)$ is the same as that of $FS(2n,n)$, while the degree of $FHS(2n,n)$ is $n+1$. The diameter of a hyper-star graph $HS(2n,n)$ is $2n-1$ if bit strings of two nodes are complement to each other. On the other hand, such nodes are connected by an edge in $FHS(2n,n)$. Thus, the diameter of $FHS(2n,n)$ is n. Since the network cost is defined as degree×diameter, the network cost of a folded hyper-star graph $FHS(2n,n)$ is n^2+n. Thus, the folded hyper-star graph is superior to the hypercube and its variations with the same number of nodes in terms of the network cost. Comparisons between folded hyper-star and other

```
Shortest_path_2(S, D, P_s)
begin
    k=0;
    if (S = D) then
        return P_s;
    obtain R = r_1 r_2 ... r_i ... r_2n, where r_i = s_i ⊕ d_i, 1 ≤ i ≤ 2n;
    let p = {i | r_i = 1, 2 ≤ i ≤ 2n}, q_1 = (S, S')_e, and q_2 = {j | (S, S'')
      ∈ E(FHS(2n, n)) − q_1}, where jth bits in S and S'' complement;
    for i = 2 to 2n do
        if (r_i = "1") then k = k + 1;
        i = i + 1;
    if (k > n) then
        P_s = P_s ∪ {S'};
        S = S';
    else
        if (|p| > 0) then
          find a node S'' such that (S, S'') is an j-dimensional edge, where
            j = min{p ∩ q_2};
          P_s = P_s ∪ {S''};
          S = S'';
    call Shortest_path_2(S, D, P_s)
end
```

Figure 5.2: The Shortest Path Routing Algorithm for the $FHS(2n, n)$

interconnection networks are presented in Table 5.1.

Table 5.1: Comparisons Between FHS and Other Networks

Network Model	Size	Degree	Diameter	Network Cost
Hypercube	2^n	n	n	n^2
Folded Hypercube	2^n	$n+1$	$\lceil \frac{n}{2} \rceil$	$\approx 0.5n^2$
Multiply-Twisted Cube	2^n	n	$\lceil \frac{n+1}{2} \rceil$	$\approx 0.5n^2$
Folded $HS(2k, k)$	$\binom{2k}{k}$	$k+1$	k	k^2+k

5.2 Fault Tolerance

It is important to have node-disjoint (or parallel) paths between two nodes in an interconnection network to speed up transfers of large amounts of data and provide alternative route in cases of node failures. In this section we show the maximal fault-tolerance capability of the folded hyper-star $FHS(2n, n)$ by enumerating all the node-disjoint paths between any two nodes. Recall that we use σ_i to denote the operation that exchanges s_1 and s_i, $2 \leq i \leq 2n$, where s_i is the complement of s_1 in a binary string. Also, for a node u, we denote by $[k_1, k_2, \ldots, k_t]$ a path obtained by applying operations $\sigma_{k_1}, \sigma_{k_2}, \ldots, \sigma_{k_t}$ in sequence from the node u, the edge connecting two nodes which are complementary to each other is the c-edge, and the graphical distance between two nodes u and v is denoted by $dist(u, v)$. In this way, a path between any two nodes in the folded hyper-star graph is representable as a sequence of k_i's and c's. The enumeration of paths is given in terms of these notations.

Theorem 12 *The number of node-disjoint paths between any two nodes u, $v \in FHS(2n, n)$ is the maximum possible and is equal to $n + 1$. The lengths of such paths are:*

***Case 1**: $dist(u, v)$ is even*

***Case 1.1**: $dist(u, v) < n$*

There are $dist(u, v)/2$ paths of length $dist(u, v)$. The remaining paths are of equal length, which is $dist(u, v) + 2$.

***Case 1.2**: $dist(u, v) = n$*

There are $n + 1$ paths of length n

***Case 2**: $dist(u, v)$ is odd*

Case 2.1: *$dist(u,v) < n$*

There are $(dist(u,v)+1)/2$ paths of length $dist(u,v)$. There is one path of length $dist(u,v) = 2$. The remaining paths are of length $dist(u,v)+4$.

Case 2.2: *$dist(u,v) = n$*

There are $n+1$ paths of length n

Proof. We prove the theorem by constructing all the node-disjoint paths for the above two cases. Through this construction the lengths and the number of these paths can be verified.

Case 1: *$dist(u,v)$* **is even.**

Case 1.1: *$dist(u,v) < n$*. For the first path between nodes u and v with length *$dist(u,v)$*, we can use the shortest path routing algorithm shown in Figure 5.2 to find it. Assume the shortest path found this way is $[k_1, k_2, \ldots, k_{dist(u,v)}]$. Then the other $dist(u,v)/2 - 1$ node-disjoint paths with length *$dist(u,v)$* are listed as follows:

$$[k_3, k_4, \ldots, k_{dist(u,v)-1}, k_{dist(u,v)}, k_1, k_2]$$

$$[k_5, k_6, \ldots, k_1, k_2, k_3, k_4]$$

$$\ldots$$

$$[k_{dist(u,v)-1}, k_{dist(u,v)}, k_1, k_2, \ldots, k_{dist(u,v)-3}, k_{dist(u,v)-2}]$$

In other words, these paths are obtained by cyclically shifting the path $[k_1, k_2, \ldots, k_{dist(u,v)}]$ $2i$ $(1 \leq i \leq dist(u,v)/2 - 1)$ operations to the left.

Clearly these paths get to the same node because they all have the same set of operations on odd and even levels. We shall show that all these paths are node-disjoint. Consider two paths

$$P_1 = [k_1, k_2, \ldots, k_{dist(u,v)}]$$

and

$$P_i = [k_{2i-1}, k_{2i}, \ldots, k_{dist(u,v)-1}, k_{dist(u,v)}, k_1, k_2, \ldots, k_{2i-3}, k_{2i-2}]$$

where P_i can be obtained from P_1 by cyclically shifting all its operations $2i$ positions to the left. After applying an operation σ_j to a node in any path, the bit value at position j must remain the same as one moves along the nodes present in a path until another operation σ_j is carried out. Therefore, nodes in P_1 are different from nodes in P_i at least in the position k_1 for the first $dist(u,v) - 2i + 2$ nodes. Furthermore, the second to the $dist(u,v) - 2i + 3$'th node in these two paths are different in at least position k_2. A repeated use of this argument will lead us to the conclusion that all the nodes in these two paths are disjoint.

Assume there is another path $[p_1, p_2, \ldots, p_{dist(u,v)}]$ that is of length $dist(u,v)$ and node-disjoint with the $dist(u,v)/2$ paths listed above. Because every shortest path between two nodes must apply the same set of operations at even and odd levels, $p_1 \in \{k_1, k_3, \ldots, k_{dist(u,v)-1}\}$, which is a contradiction to the assumption that this path is node-disjoint with the above paths. Hence, the number of such shortest node-disjoint paths cannot exceed $dist(u,v)/2$.

The case when a c-edge is involved can be easily proved using the above argument. The shortest path distance between node u and v is $dist(u,v)$ and all such paths are already chosen, therefore, the alternate paths must be of length greater than $dist(u,v)$. The alternate paths cannot be of length $dist(u,v) + 1$, otherwise it will introduce a cycle of length $2dist(u,v) + 1$, which is odd. Since the folded hyper-star does not admit cycles with odd length, the next choice for the alternate paths is a path of length $dist(u,v) + 2$.

To construct the alternate paths of length $dist(u,v)+2$, we have to utilize the other neighbours of the source node that have not been used in the $dist(u,v)/2$ shortest paths. Which means the first operation cannot be in the set $\{k_1, k_3, \ldots, k_{dist(u,v)-1}\}$. Say operation k_n will go from the source node to such a neighbour, then the alternate paths are all given in the following form.

$$[k_n, k_{dist(u,v)}, k_{dist(u,v)-1}, \ldots, k_2, k_1, k_n]$$

Since $k_n \notin \{k_1, k_2, \ldots, k_{dist(u,v)}\}$, bit k_n will not be changed until the very last operation, which is σ_{k_n} again, and this operation changes bit k_n back to the value it was in the source node. Because the source and destination nodes only differ in bits $k_1, k_2, \ldots, k_{dist(u,v)}$, this sequence of operations will lead to the destination node v. Note the sequence between the two k_n on the ends is the reverse of one shortest path, this is to ensure the operations are taken on the right kind of levels (odd level or even level), the reverse of any shortest path will do, the argument is the same. All these paths utilize the same set of operations except the first and the last, thus all these paths are node-disjoint. The case when $[k_n]$ is a c-edge can be proved easily using the same argument.

To this point we have utilized every neighbour of the source node to find a parallel path, so the total number of node-disjoints paths between two nodes in this case is the degree of the graph, which is $n+1$.

Example: In $FHS(8,4)$, $u = 00001111$, $v = 00010111$, $dist(u,v) = 2$. The 1 path of length 2 is (the number in the parenthesis is the position of the bit being changed at each step):

$$00001111(5) \rightarrow 10000111(4) \rightarrow 00010111$$

The 4 node-disjoint paths of length 4 are:

$$00001111(c) \rightarrow 11110000(5) \rightarrow 01111000(4) \rightarrow 11101000(c) \rightarrow 00010111$$

$$00001111(6) \rightarrow 10001011(4) \rightarrow 00011011(5) \rightarrow 10010011(6) \rightarrow 00010111$$

$$00001111(7) \rightarrow 10001101(4) \rightarrow 00011101(5) \rightarrow 10010101(7) \rightarrow 00010111$$

$$00001111(8) \rightarrow 10001110(4) \rightarrow 00011110(5) \rightarrow 10010110(8) \rightarrow 00010111$$

Case 1.2: $dist(u, v) = n$. For this case, we first get the $dist(u, v)/2 = n/2$ shortest paths as we did in Case 1.1. Assume the bits (excluding the first bit) at positions $e_1, e_2, \ldots, e_{n-1}$ are the same in u and v. If we exchange the first bit with any of these $n - 1$ bits, we get one bit "further" from the destination node. However, the use of c-edge will go from a node at distance d from destination to a node at distance $2n-1-d$ from the destination. Thus, we can change bits at positions $e_1, e_2, \ldots, e_{n-1}$, to make the node of distance $n + (n - 1) = 2n - 1$ from node v, and then use the c-edge, to make the resulting node $(2n-1)-(2n-1) = 0$ edge from v, which is v. As a matter of fact, we can change bits $e_1, e_2, \ldots, e_{n-1}$ and use the c-edge in any order, we still get to v after n edges.

Example: In $FHS(8, 4)$, $u = 0000111$, $v = 00110011$, $dist(u, v) = 4$. The 5 node-disjoint paths of length 4 are:

$$00001111(5) \rightarrow 10000111(3) \rightarrow 00100111(6) \rightarrow 10100011(4) \rightarrow 00110011$$

$$00001111(6) \rightarrow 10001011(4) \rightarrow 00011011(5) \rightarrow 10010011(3) \rightarrow 00110011$$

$$00001111(c) \rightarrow 11110000(7) \rightarrow 01110010(2) \rightarrow 10110010(8) \rightarrow 00110011$$

$$00001111(7) \rightarrow 10001101(2) \rightarrow 01001101(8) \rightarrow 11001100(c) \rightarrow 00110011$$

$$00001111(8) \rightarrow 10001110(\overset{*}{c}) \rightarrow 01110001(2) \rightarrow 10110001(7) \rightarrow 00110011$$

Case 2: $dist(u, v)$ **is odd.**

Case 2.1: $dist(u, v) < n$. We still use the shortest path routing algorithm to find our first shortest path, say the path is $[k_1, k_2, \ldots, k_{dist(u,v)}]$, then the other shortest node-disjoint paths between node u and v are listed as follows:

$$[k_3, k_2, k_5, k_4, \ldots, k_{dist(u,v)}, k_{dist(u,v)-1}, k_1]$$

$$[k_5, k_2, k_7, k_4, \ldots, k_1, k_{dist(u,v)-1}, k_3]$$

$$\ldots$$

$$[k_{dist(u,v)}, k_2, k_1, k_4, \ldots, k_{dist(u,v)-4}, k_{dist(u,v)-1}, k_{dist(u,v)-2}]$$

That is, we only cyclically shift the operations applied on odd levels to the left, one operation per time. The operations applied on the even levels are the same for all the paths.

It is easy to see that they are all of the same length, which is the distance between node u and v, and they are all legible paths from u to v. The number of these paths is equal to the number of operations applied on odd levels, which is $(dist(u, v) + 1)/2 = (n + 1)/2$. These paths being node-disjoint can be proved by similar argument as used in Case 1.

Then one alternate path of length $dist(u, v) + 2$ is:

$$[c, k_1, k_2, \ldots, k_{dist(u,v)}, c]$$

The justification is the same as in **Case 1.1**.

The other $n - (dist(u,v)+1)/2$ parallel paths are in the form:

$$[p, q, k_1, k_2, \ldots, k_{dist(u,v)}, p, q]$$

The edge $[p]$ connects node u to a neighbour that is not utilized in the above shortest paths. Thus changes the bit p that is the same in nodes u and u. But the second appearance of p in the sequence changes this bit back. The edge $[q]$ is one "redundant" step to make sure edges $[k_1], [k_2] \ldots$ are on the right kind of level (odd or even level). Also, if there is no such edge as $[q]$, then it is impossible to make these paths node-disjoint. And all the bits at positions $k_1, k_2, \ldots, k_{dist(u,v)}$ are changed by this sequence of operations.

Example: In $FHS(8,4)$, $u = 00001111$, $v = 10010011$, $dist(u,v) = 3$. The 2 node-disjoint paths of length 3 are:

$$00001111(5) \rightarrow 10000111(4) \rightarrow 00010111(6) \rightarrow 10010011$$

$$00001111(6) \rightarrow 10001011(4) \rightarrow 00011011(5) \rightarrow 10010011$$

The one path using the c-edge is:

$$00001111(c) \rightarrow 11110000(5) \rightarrow 01111000(4) \rightarrow 11101000(6)$$

$$\rightarrow 01101100(c) \rightarrow 10010011$$

The 2 node-disjoint paths of length 7 are:

$$00001111(7) \rightarrow 10001101(2) \rightarrow 01001101(5) \rightarrow 11000101(4) \rightarrow 01010101(6)$$

$$\rightarrow 11010001(7) \rightarrow 01010011(2) \rightarrow 10010011$$

$$00001111(8) \rightarrow 10001110(2) \rightarrow 01001110(5) \rightarrow 11000110(4) \rightarrow 01010110(6)$$

$$\rightarrow 11010010(8) \rightarrow 01010011(2) \rightarrow 10010011$$

Case 2.2: $dist(u, v) = n$. The first $(dist(u, v) + 1)/2 = (n + 1)/2$ shortest parallel paths are the same as in **Case 2.1**. The other shortest paths can be constructed in the same way as **Case 1.2**, which is combine changing the bits that are the same in u and v and using c-edge, the justification is also the same.

Example: In $FHS(6, 3)$, $u = 010101$, $v = 110010$, $dist(u, v) = 3$. The 4 node-disjoint paths of length 3 are:

$$010101(4) \rightarrow 110001(5) \rightarrow 010011(6) \rightarrow 110010$$

$$010101(6) \rightarrow 110100(5) \rightarrow 010110(4) \rightarrow 110010$$

$$010101(2) \rightarrow 100101(3) \rightarrow 001101(c) \rightarrow 110010$$

$$010101(c) \rightarrow 101010(2) \rightarrow 011010(3) \rightarrow 110010$$

□

Chapter 6

Conclusions

In this thesis, we have studied the hyper-star graph, especially its regular form $HS(2n, n)$. We proved some interesting properties of $HS(2n, n)$, as well as designed some communication algorithms for it. In addition, we also studied briefly a variation of the hyper-star graph, the folded hyper-star graph.

Specifically, we have discussed:

- the relation between odd graphs and hyper-star graphs. This relation is useful in tracing the Hamiltonian cycles in hyper-star graphs.
- the relation between even graphs and hyper-star graphs. This study leads us to relating the study of the Hamiltonicity problem of the hyper-star graphs to the middle-cubes.
- the surface area of the regular hyper-star graph.
- the Hamiltonicity problem of the hyper-star graphs. We proved the regular hyper-star graphs are isomorphic to the middle-cubes, and made the conjecture

that all regular hyper-star graphs are Hamiltonian. We also presented one way of finding the Hamiltonian cycles in regular hyper-star graphs.

- the neibourhood broadcasting algorithm on single-port model, our algorithm is optimal in view of the lower bound imposed by the communication model.
- the broadcasting algorithm on all-port model. We used a greedy approach to send the information level by level, the algorithm is optimal time-wise and creates no redundancy.
- the fault tolerance of the folded hyper-star graph. We proved that the folded hyper-star graphs are maximally fault-tolerant by constructing all the parallel paths between any two nodes.

The hyper-star graph is constructed as a hybrid of the hypercube and the star graph. It is thus hoped to possess the merits of both the hypercube and the star graph. Despite all the nice properties of the hyper-star graph, a recursive decomposition of the hyper-star graphs has yet to be found. As introduced in the thesis, a regular hyper-star graph can be easily decomposed to two identical subgraphs. But these two subgraphs are irregular hyper-stars, so if we carry the decompositon one more step, the subgraphs we end up with will no longer be identical. For this reason, we consider the regular hyper-star graphs can not be recursively decomposed using the method introduced in this thesis. On the other hand, both the hypercubes and the star graphs can be recursively decomposed to several identical subgraphs with a smaller scale, and this property makes the design of such algorithms as broadcasting and sorting easy and neat.

The above argument gives rise to two questions:

- Does the hyper-star graphs admit recursive characterization? In other words, can it be decomposed to several identical copies recursively using some other decomposition methods? If so, many algorithms can be designed in an easy and intuitive way.

- Is being recursively decomposable a necessary condition for the design of algorithms such as prefix sums, sorting? The answer is probably no. But the reason we focus on this property of a graph is that it makes the design and the implementation of many essential algorithms rather easy.

With all this being said, finding an alternative way of decomposing the hyper-star graph and designing algorithm (prefix sums, sorting, ...) in a non-recursive way would be interesting.

Bibliography

[1] S. B. Akers, D. Harel, and B. Krishnamurthy. The star graph: An attractive alternative to the n-cube. *Proc. International Conference on Parallel Processing*, pages 393–400, 1987.

[2] S. B. Akers and B. Krishnamurthy. The fault tolerance of star graphs. *2nd International Conference on Supercompputing*, III:270–276, 1987.

[3] S. B. Akers and B. Krishnamurthy. A group theoretic model for symmetric interconnection networks. *IEEE Transactions on Computers*, 38(4):555–566, 1989.

[4] S. G. Akl. *The Design and Analysis of Parallel Algorithms.* Prentice Hall, Englewood Cliffs, 1997.

[5] S. G. Akl. *Parallel Computation: Models And Methods.* Prentice Hall, Upper Saddle River, 1997.

[6] S. G. Akl and K. Qiu. Fundamental algorithms for the star and pancake interconnection networks with applications to computational geometry. *Networks*, 23:215–225, 1993.

[7] S. G. Akl and T. Wolff. Efficient sorting on the star graph interconnection network. *Technical Report*, (97):407–414, 1998.

[8] A. Bagchi, S. L. Hakimi, and E. F. Schmeichel. Gossiping in a distributed network. *IEEE Transactions on Computers*, 42(2):253–256, 1993.

[9] L. N. Bhuyan and D. P. Agrawal. Generalized hypercube and hyperbus structures for a computer network. *IEEE Transactions on Computers*, 33(4):323–333, 1984.

[10] M. Cosnard and A. Ferreira. On the real power of loosely coupled parallel architectures. *Parallel Processing Letters*, 1(2):103–111, 1991.

[11] R. K. Das, K. Mukhopadhyaya, and B. P. Sinha. A new family of bridged and twisted hypercubes. *IEEE*, 43(10):1240–1247, 1994.

[12] K. Day and A. Tripathi. A comparative study of topological properties of hypercubes and star graphs. *IEEE Transactions on Parallel and Distributed Systems*, 5(1):31–38, 1994.

[13] K. Efe. A variation on the hypercubes with lower diameter. *IEEE*, 40(11):1312–1316, 1991.

[14] A. EI-Amawy and S. Latifi. Properties and performance of folded hypercubes. *IEEE Transactions on Parallel and Distributed Systems*, 2(1):31–42, 1991.

[15] A.-H. Esfahanian, L. M. Ni, and B. E. Sagan. The twisted n-cube with application to multiprocessing. *IEEE Transactions on Computers*, 40(1):88–93, 1991.

[16] M. Flynn. Some computer organizations and their effectives. *IEEE Trans. Comput.*, C(21):948–960, 1972.

[17] A. Ghafoor. A class of fault-tolerant multiprocessor networks. *IEEE Transactions on Reliability*, 38(1):5–15, 1989.

[18] A. Ghafoor and T. R. Bashkow. A study of odd graghs as fault-tolerant interconnection newtorks. *IEEE*, pages 225–232, 1991.

[19] J. P. Huang, S. Lakshmivarahan, and S. K. Dhall. Analysis of interconnection networks based on cayley graphs of strong generating sets. *IEEE*, pages 42–45, 1994.

[20] N. Imani, H. Sarbazi-Azad, and S. G. Akl. Some topological properties of star graphs: The surface area and volume. *Discrete Mathematics*, 2008.

[21] E. O. Jong-Seok Kim, H.-O. Lee, and Y.-N. Heo. Topological and communication aspects of hyper-star graphs. *Proceedings of the 18th International Symposium on Computer and Information Science*, 2003.

[22] B. H. H. Juurlink, J. F. Sibeyn, and P. S. Rao. Gossiping on meshes and tori. *IEEE Transactions on Parallel and Distributed Systems*, 9(6):513–525, 1998.

[23] J.-S. Kim, E. Cheng, L. Liptak, and H.-O. Lee. Embedding hypercubes, rings, and odd graphs into hyper-stars. *International Journal of Computer Mathematics*, pages 1–8, 2008.

[24] S. Lakshmivarahan, J.-S. Jwo, and S. K. Dhall. Symmetry in interconnection networks based on cayley graphs of permutation groups : A survey. *Parallel Computing*, 19:361–382, 1993.

[25] S. Latifi and P. K. srimani. Transposition networks as a class of fault- tolerant robust networks. *IEEE Transactions on Computers*, 45(2):230–238, 1996.

[26] H.-O. Lee, J.-S. Kim, E. Oh, and H.-S. Lim. Hyper-star graph: A new interconnection newtork improving the network cost of the hypercube. *M. H. Shafazand and A. M. Tjoa (Eds): EurAsia-ICT*, pages 858–865, 2002.

[27] S. Madabhushi, S. Lakshmivarahan, and S. Dhall. Analysis of the modified even networks. *Technical Report*, University of Oklahoma, 1990.

[28] F. Portier and T. Vaughan. Whitney numbers of the second kind for the star poset. *Europ. J. Combinatorics*, 11:277–288, 1990.

[29] F. P. Preparata and J. Vuillemin. The cube-connected cycles: A versatile network for parallel computation. *Computer Architecture and Systems*, 24(5):300–309, 1981.

[30] K. Qiu. Broadcasting on the star and pancake interconnection networks. *Proceedings of the 9th International Parallel Processing Symposium*, pages 660–665, 1995.

[31] K. Qiu and S. G. Akl. On some properties of the star graph. *VLSI Design*, 2(4):389–396, 1995.

[32] Y. Saad and M. H. Schultz. Topological properties of hypercubes. *IEEE*, pages 867–872, 1988.

[33] M. R. Samatham and D. K. Pradhan. The de bruijin multiprocessor network: A versatile parallel processing and sorting network for vlsi. *IEEE Transactions on Computers*, 38(4):567–581, 1989.

[34] H. S. Stone. Parallel processing with the perfect shuffle. *IEEE Transactions on Computers*, 20(2):153–160, 1971.