TOPOLOGICAL OPTIMIZATION AND DEADLOCK-FREE ROUTING IN INTERCONNECTION NETWORKS BASED ON HYPERCYCLES

by

SIVAKUMAR RADHAKRISHNAN
M.A.Sc., University of Victoria, 1991
B.Eng., University of Madras, 1989

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

We accept this dissertation as conforming
to the required standard

Dr. N. J. Dimopoulos, Supervisor, Dept. of Elect. & Comp. Eng.

Dr. K. F. Li, Member, Dept. of Elect. & Comp. Eng.

Dr. F. El-Guibaly, Member, Dept. of Elect. & Comp. Eng.

Dr. M. Serra, Outside Member, Dept. of Computer Science

Dr. H. M. Alnuweiri, External Examiner, Dept of Elect. Eng.,
University of British Columbia

© SIVAKUMAR RADHAKRISHNAN, 1995

University of Victoria

All rights reserved. This dissertation may not be reproduced in whole or in part by photocopy or other means, without the permission of the author.

Supervisor: Dr. N.J. Dimopoulos

ABSTRACT

In this dissertation, we address some of the architectural, algorithmic and communication issues involved in the design of parallel computer systems. The performance of large scale parallel computers depends mainly on interprocessor communication. As a result, considerable effort has been directed towards the design of efficient interconnection networks, routing algorithms and hardware routers. This work specifically focuses on hypercycle based interconnection networks which are a class of multidimensional, symmetric graphs that are generalizations of several popular topologies such as the ring, hypercubes and generalized hypercubes. Hypercycles are products of circulant graphs with simple routing schemes. Their ability, given a fixed degree, to form alternate graphs makes them attractive for space/mass limited applications, e.g. on-board computers in satellites, autonomous vehicles, avionics, robotic control and submersibles.

One of the issues that confronts the design of efficient interconnection networks is finding a topology with minimum average distance, diameter and the degree*diameter product. New results on the synthesis of cost optimal hypercycle network topologies minimizing the above metrics are presented.

Graph embeddings permit a parallel machine having a given interconnection topology to simulate different computational structures such as chains, cycles, grids and trees. An efficient algorithm for embedding complete binary trees in a hypercycle network with unit dilation is proposed. The dissertation also investigates the problem of embedding Hamiltonian cycles and a generalized m-ary gray code algorithm for hypercycles is evolved.

Routing algorithms for interconnection networks must be efficient and free from deadlocks. Two algorithms, namely the e-cube and prioritized mixed e-cube & backtracking schemes are proposed and shown to be deadlock preventing. The system architecture of a routing engine to implement deadlock-free routing algorithms is proposed. A VLSI chip that implements the e-cube algorithm has been designed in 1.2 μm CMOS technology for a 4-dimensional hypercycle network with a measured throughput in excess of 15 million decisions per second.

Finally, a framework for rapidly prototyping policy-specific controllers for the routing engine using a novel general purpose hardware description language called CoDeL (Controller Design Language) has been developed. Using CoDeL, a circuit switching based e-cube routing controller has been synthesized and mapped to a Xilinx 4010PC84 chip. Test results of the controller indicate that a link in a source-destination path can be established in less than 3 μs.

Examiners:

Dr. N. J. Dimopoulos, Supervisor, Dept. of Elect. & Comp. Eng.

Dr. K. F. Li, Member, Dept. of Elect. & Comp. Eng.

Dr. F. El-Guibaly, Member, Dept. of Elect. & Comp. Eng.

Dr. M. Serra, Outside Member, Dept. of Computer Science

Dr. H. M. Alnuweiri, External Examiner, Dept of Elect. Eng.,
University of British Columbia

Table of Contents

Abstract ii

Table of Contents iv

List of Figures viii

List of Tables x

Notation xi

Acknowledgement xiv

Dedication xv

1 Introduction 1

1.1 Interconnection networks . . . 3

1.1.1 Symmetric Graphs . . . 3

1.2 Motivation . . . 4

1.2.1 Architectural Issues . . . 5

1.2.2 Algorithmic Issues . . . 5

1.2.3 Communication Issues . . . 5

1.3 Dissertation Outline . . . 6

1.3.1 Theoretical Issues . . . 6

1.3.2 Implementation Issues . . . 7

1.4 Contributions . . . 7

1.5 Definitions . . . 8

1.5.1 Hypercycle graphs . . . 9

1.6 Conclusion . . . 14

2 Topological Optimization in Hypercycles 15

2.1 Introduction . . . 15

2.2 Formulation of Optimal Hypercycle Network Design 16
2.3 Optimization of the average distance and diameter 19
2.3.1 Bounds for average distance and diameter 19
2.3.2 Perturbation Analysis 22
2.3.3 Numerical Experiments 33
2.4 Other Optimization cases 35
2.4.1 Case 2 35
2.4.2 Case 3 36
2.4.3 Optimum Dimension 38
2.5 Minimization of Degree*Diameter product 39
2.5.1 Case 1: $p = 2$ and $\vec{\rho} = [2\ 2\ \ldots\ 2\ 2]$ 42
2.5.2 Other Optimization Cases: 48
2.6 Network Latency and Throughput 48
2.7 Conclusion 51

3 Embedding Complete Binary Trees in Hypercycles 52
3.1 Introduction 52
3.2 Bandwidth problem for trees 54
3.3 Embedding T_h in multi-dimensional hypercycles 57
3.3.1 Multidimensional embedding algorithm 61
3.4 Optimization 68
3.4.1 Performance Comparison 69
3.5 Conclusion 71

4 Hamiltonian cycles in Hypercycles 72
4.1 Introduction 72
4.2 Problem History and Motivation 72
4.3 Hamiltonian Cycle Algorithm 75
4.3.1 Uniform Hypercycle 76
4.3.2 Even Hypercycle 78
4.3.3 Odd Hypercycle 83
4.3.4 Time complexity 90
4.3.5 Extension 90
4.4 Ranking and Unranking of Hamiltonian cycles 91
4.5 Hamiltonian Decomposition 97

4.5.1 Extension 99
4.6 Conclusion 101

5 Deadlock Preventing Routing Algorithms for Hypercycles 102
5.1 Prioritized Dimension e-cube routing 105
5.1.1 E-cube Algorithm 110
5.2 Prioritized Mixed E-cube Backtracking Routing 111
5.2.1 PRIMER Algorithm 113
5.3 Conclusion 114

6 Hardware Implementation 115
6.1 Introduction 115
6.2 Hypercycle Router 117
6.2.1 Architecture 117
6.2.2 Implementation 118
6.2.3 Data Path 118
6.2.4 Operating Modes 121
6.2.5 VLSI Chip 122
6.2.6 Test results 122
6.3 Design of the Controller 125
6.3.1 Hardware Description Languages 127
6.3.2 CoDeL Syntax 129
6.3.3 CoDeL Compiler 137
6.4 A Routing Engine Controller Implementation 139
6.5 Conclusion 142

7 Conclusion 144
7.1 Synopsis 144
7.2 Suggestions for Further Research 145

Bibliography 148

Appendix A 155
A.1 Proof of Lemma 2.1 155

Appendix B **157**
B.1 Proof of Lemma 2.6 157
B.2 Proof of Lemma 2.7 158
B.3 Proof of Lemma 2.8 160

Appendix C **161**

Appendix D **164**
D.1 Proof of Lemma 4.1 164

Appendix E CoDeL program listing of e-cube controller **165**

List of Figures

Figure 1.1 Graph Multiplication in hypercycles 10
Figure 1.2 Examples of hypercycle graphs 11
Figure 1.3 Greedy routing in hypercycle G_{43}^{11} 14

Figure 2.1 Bounds of the average distance for $\rho = 2$ in 1-dimensional hypercycles. 20
Figure 2.2 Plot of normalized network latency in Hypercycles for $M = 256$ and $d = 8$. The maximum network utilization for each case is given in brackets 49

Figure 3.1 Examples of embedding T_h on a line with a given bandwidth 56
Figure 3.2 Placement structure used by the Tree_into_line algorithm for embedding T_h on a line 56
Figure 3.3 Embedding of T_1, T_2 and T_3 in hypercycles using Tree_into_line embedding algorithm. The numbers on the edges represent the difference in connectivity between adjacent nodes 58
Figure 3.4 Procedure Two_dim_embed for embedding a tree T_h in a 2-dimensional hypercycle $\mathcal{G}_{m_1 m_2}^{\rho_1 \rho_2}$ where $m_1 = 5$, $m_2 = 2^{h-1} - 1$, $\rho_1 = 2$ and $\rho_2 = \lceil \frac{2^{h-2}-1}{h-2} \rceil$ 59
Figure 3.5 Some examples of unit dilation embedding using Algorithm Two_dim_embed. Only the relevant edges are shown for clarity 60
Figure 3.6 Pictorial representation of Algorithm Multidim_Tree_embed 62
Figure 3.7 Procedure Multidim_Tree_embed for embedding T_h in hypercycles . . 63
Figure 3.8 Procedure Node_label for recursive labelling of T_h 64
Figure 3.9 An illustration of T_5 unit dilation embedding in G_{257}^{122} using Algorithm Multidim_Tree_embed 65
Figure 3.10 Plot of the cost of hypercube and hypercycle structures for unit dilation tree embeddings for (a) $h \leq 6$ and (b) $7 \leq h \leq 12$. 70

Figure 4.1 Cayley graph C=G(S,T) isomorphic to $\mathcal{G}_{25}^{11}$ 73
Figure 4.2 Procedure Uniform for computing a Hamiltonian cycle in $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$. Assume that variables x and $sign$ are globally defined 76

Figure 4.3 Procedure Path_Uniform for computing a Hamiltonian path in $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$. 76
Figure 4.4 An example of a Hamiltonian Cycle traced in hypercycle $\mathcal{G}_{44}^{11}$ for the Uniform category with node (0,0) as the starting point 79
Figure 4.5 Procedure Even for computing a Hamiltonian cycle for an even hypercycle 80
Figure 4.6 Procedure Path_Even for computing a Hamiltonian path in $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ . . . 80
Figure 4.7 An example of a Hamiltonian Cycle traced in hypercycle $\mathcal{G}_{45}^{11}$ for the Even category with node (0,0) as the starting point 81
Figure 4.8 Procedure Path_Odd for computing a Hamiltonian path in $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ . . . 83
Figure 4.9 Node mapping in $\mathcal{G}_{35}^{11}$ 87
Figure 4.10 Procedure Odd for computing a Hamiltonian Cycle in an odd hypercycle 88
Figure 4.11 Example of a Hamiltonian Cycle in $\mathcal{G}_{75}^{11}$ for the Odd category with node (0,0) as the starting point 89

Figure 5.1 Channel Dependency graph 104
Figure 5.2 Odd-Even preference routing scheme in $\mathcal{G}_4^1$ 106
Figure 5.3 Dependency graph for $\mathcal{G}_4^1$ 107

Figure 6.1 Configuration of a hypercycle routing engine in a network node . . . 116
Figure 6.2 Block diagram of the internal components of e-cube router 119
Figure 6.3 Block diagram of deadlock-free router 120
Figure 6.4 Layout of the *e*-cube router chip designed in 1.2 μm CMOS technology 123
Figure 6.5 Floorplan of the *e*-cube router chip 124
Figure 6.6 Timing Diagram of the e-cube router chip 126
Figure 6.7 Basic Structure of a CoDeL program 129
Figure 6.8 A typical frame for a message header. Both destination and source addresses are expressed in mixed radix number formats 130
Figure 6.9 An example of a frame and its associated bit structures 131
Figure 6.10 A full handshaking protocol 132
Figure 6.11 Structure of an I/O protocol engine in CoDeL 133
Figure 6.12 Examples of port and protocol definitions 133
Figure 6.13 Structure of a circuit switching routing node in a hypercycle network 138
Figure 6.14 Pseudo code of the *e*-cube controller algorithm 140

List of Tables

Table 2.1 Average Distance Minimization for 3-D hypercycle - Case 1 33
Table 2.2 Violations in average distance minimization for 3-D hypercycles - Case 1 33
Table 2.3 Violations in diameter for 3-D hypercycles - Case 1 34
Table 2.4 Statistics of violations for Average distance 35
Table 2.5 Statistics of violations for Diameter 35
Table 2.6 Neighbourhood of balanced point for violation cases for Average Distance 36
Table 2.7 Average Distance Minimization for a 3-D hypercycle - Case 2 37
Table 2.8 Average Distance Minimization for a 3-D hypercycle - Case 3 37
Table 2.9 Case-study 2: Optimal Hypercycle Topologies for $M = 256$ with $d = 8$ 50
Table 2.10 Case-study 2: Optimal Hypercycle Topologies for $M = 400$ with $d = 24$ 51

Table 3.1 Bandwidth of some complete binary trees 55
Table 3.2 Optimal topologies for binary tree embedding in hypercycles 67

Table 6.1 Register Designation 120
Table 6.2 Results of single stuck-at fault testing 124
Table 6.3 List of CoDeL operators 135
Table 6.4 Statistics of e-cube controller implementation 142

Notation

List of Principal Symbols

δ	–	Average distance between two nodes in the hypercycle graph
r	–	Dimension of a hypercycle (integer)
$\vec{m}$	–	Node radix vector
$\vec{\rho}$	–	Connectivity vector
m_i	–	Node radix of a hypercycle in dimension i (integer)
ρ_i	–	Connectivity in dimension i (integer)
M	–	Total number of nodes in a hypercycle
$\mathcal{G}_{\vec{m}}^{\vec{\rho}}$	–	An r-dimensional hypercycle graph
$\mathcal{V}_{\vec{m}}^{\vec{\rho}}$	–	Set of nodes in the hypercycle graph
$\mathcal{E}_{\vec{m}}^{\vec{\rho}}$	–	Set of edges in the hypercycle graph
Ω	–	Solution space of dimension r

Design Functions

k	–	Diameter of a hypercycle
k_b	–	Diameter of balanced hypercycle
k'	–	Diameter of an unbalanced hypercycle
d	–	Degree of a hypercycle
δ	–	Average distance of a hypercycle
δ_b	–	Average distance of an balanced hypercycle
δ'	–	Average distance of an unbalanced hypercycle
χ	–	Product of hypercycle degree and diameter

Other Variables

S_j	–	Sum of distances of all nodes in the j^{th} dimension from the reference node
D_l	–	Shortest distance of node l from reference node
$\hat{\delta}$	–	Upper bound on the average distance of a hypercycle
$\check{\delta}$	–	Lower bound on the average distance of a hypercycle
$\hat{r}$	–	Upper bound on the dimension of a hypercycle
$\check{r}$	–	Lower bound on the dimension of a hypercycle

Performance Metrics

T	–	Network Latency
T'	–	Normalized Network Latency
T_d	–	Traffic Density
C_p	–	Network Capacity
T_p	–	Network Throughput
T_p'	–	Normalized Network Throughput
σ	–	Communication overhead
$b(\mathcal{G})$	–	Bandwidth of a graph $\mathcal{G}$
T_h	–	A complete binary tree having $2^{h+1}-1$ nodes

List of Abbreviations

AHPL	–	A Hardware Description Language
ALU	–	Arithmetic & Logic Unit
ASIC	–	Application Specific Integrated Circuit
ATE	–	Automatic Test Equipment
BTOR	–	Back-Track-to-the-Origin-and-Retry-Routing
CAD	–	Computer Aided Design
CATPAG	–	Computerized Automatic Test Pattern Generator
CLA	–	Carry Look-ahead Adder
CLB	–	Combinational Logic Block
CMC	–	Canadian Micro-electronics Corporation
CMOS	–	Complementary Metal Oxide Semiconductor
CMOS4S	–	Northern Telecom Electronics 1.2-micron double-polysilicon, double level metal, twin-well CMOS process
CoDeL	–	Controller Description Language
CPU	–	Central Processing Unit
CSMA/CD	–	Carrier Sense Multiple Access with Collision Detection
DDL	–	Digital Systems Language
DFT	–	Design For Testability
DSM	–	Design Scale Micron
GHC	–	Generalized HyperCubes
ISP	–	Instruction Set Processor

LSSD	–	Level Sensitive Scan Design
MDPS	–	Million Decisions Per Second
MIMD	–	Multiple Instruction Multiple Data
MRN	–	Mixed Radix Number
NPG	–	Next Port Generator
PE	–	Processing Element
PGA	–	Pin Grid Array
PLCC	–	Plastic Leaded Chip Carrier
PRIMER	–	Prioritized Mixed Ecube-backtracking Routing
RISC	–	Reduced Instruction Set Computer
ROM	–	Read-Only-Memory
SISD	–	Single Instruction Single Data
VHDL	–	Very high speed integrated circuits Hardware Description Language
VLSI	–	Very Large Scale Integration

Unix is a trade mark of AT & T laboratories. **Cadence** and **Edge** are trademarks of Cadence Design Inc. **Logic Master** is a trademark of Integrated Measurement Systems Inc. **HSPICE** is a trademark of Metasoftware Inc. **XACT** is the trademark of Xilinx Inc. **Design Architect, Quicksim, Autologic,** and **Mentor's system-1076 VHDL** are trademarks of Mentor Graphics Corporation. **Compass** is a trademark of Compass Design Automation.

Acknowledgement

I wish to record my gratitude to my supervisor, Dr. N. J. Dimopoulos for his encouragement, support, and insightful advice during the course of my graduate studies at UVic. I thank Drs. F. El-Guibaly and K. F. Li for for their technical guidance related to my research and my duties as a teaching assistant. I acknowledge the cooperation of my colleagues, Mr. V. V. Dimakopoulos and Mr. M. H. Chowdhury for their feedback in this research project.

I am grateful to the VLSI research group including the graduate students, and the faculty of the Department of Computer Science for their help on the testability aspects of VLSI. In particular, I am indebted to Dr. R. W. Byrne for his help in fault testing, to Prof. Joseph Peters of the Simon Fraser University, Dr. Dominique Sotteau of the Université de Paris-Sud, and Drs. J. Ellis & W. Myrvold of the University of Victoria for their intuitive and thought provoking comments on graph embedding.

I express my gratitude with sincerity to the department technicians - Mr. K. Jones, Mr. S. Campbell and the system analysts, Mr. W. A. Keddy, Mr. R. W. Kelly and Mr. W. Kastellic for their untiring efforts in helping me in many critical periods including software setup, maintenance and for providing IC testing/computing facilities in the VLSI/microelectronics laboratories, to the department secretaries for their excellent administrative duties and paperwork.

I thank the Canadian Microelectronics Corporation for its assistance in fabricating a VLSI chip for this project. Generous financial assistance received from my alma mater, the University of Victoria in the form of Fellowships/Awards and research grants provided by Dr. N. J. Dimopoulos (through NSERC) is gratefully acknowledged.

A word of gratitude to my parents, relatives, friends and well-wishers for their moral support, motivation and encouragement during my graduate studies at UVic. Finally, my thanks to PMC-Sierra Inc. for providing facilities to complete my dissertation.

Dedication

The research work concerning the hypercycle interconnection networks presented here is fully dedicated to ***His Holiness, the Late Sri Chandarasekharendra Saraswathi Sankaracharya Swamigal of the Kanchi Kamakoti Peetam, Kancheepuram, Tamilnadu, India****,* ***the Revered Sri Mukkur Lakshmi Nrisimhacharyar, Madras, India, and to my parents.***

Asathoma Sadgamaya
Thamasoma Jyotirgamaya
Mrityoma Amrutamgamaya

- from the Brighadharanya Upanishad

Knowledge is Supreme and the
Knowledge of the Infinite leads one
from impermanence to Permanence;
from darkness onto Light;
and from mortality to Immortality.

Chapter 1

Introduction

Every science and every inquiry,
and similarly every activity
and pursuit, is thought
to aim at some good
- Aristotle

Parallel computing is one of the most exciting fields to achieve prominence since the invention of electronic computers in the 1940s. Digital computers spanning from the *ENIAC* to the main/mini computers of the 70s to the present day scalable Reduced Instruction Set Computer (RISC) microprocessors have witnessed a major revolution in terms of processing speed and cost. As the world heads towards the 21st century, powerful scientific/socio-economic factors are driving the industry towards new computing architectures. Applications such as molecular modeling, computational fluid dynamics, econometric forecasting, visualization and imaging [1] require massively parallel computing and storage and the quest for higher speed, lower cost and more reliable computers remains a major challenge for scientists and engineers.

Notwithstanding the architectural advances within the Central Processing Unit (CPU) and memory of the *von Neumann* or Single Instruction Single Data (SISD) machines, sequential processing has been a major limitation in applications such as seismic processing for petroleum exploration, structural analysis for the aerospace, automotive, chemical and semiconductor industries, atmospheric modeling for weather prediction, computational electromagnetics, combustion modeling for engine design, medical imaging and multimedia applications. To circumvent the von Neumann bottleneck, innovations have been brought about in the form of *parallelism* and this has given way to the new field of parallel computing. The important approaches to achieve parallelism include pipelining, multiplicity of functional units, array processing and multiprocessing [2].

Of contemporary interest is the scalable parallel architecture which is seen as a viable platform to meet the grand challenges of number crunching problems [3]. Such systems traditionally known as multicomputers but more recently called *massively parallel computers* comprise of a multitude of processor-memory nodes interconnected via specialized high speed point-to-point networks and interprocessor communication is achieved through the message passing paradigm.

The earliest approach in using multiple Central Processing Units (CPUs) dates back to the PILOT computer system in the late 1950s [4]. Later developments include the ILLIAC IV [5] array processor which consists of 64 processing elements (PE) each containing an Arithmetic and Logic Unit (ALU) and local memory and interconnected via a two dimensional mesh structure. The C.mmp machine [6] which originated at the Carnegie Mellon University consists of 16 processors with 16 memory modules connected over a cross-bar switch. The mid 1970s saw the advent of mainframes such as the IBM System 370 and PDP-11. The Cray X/Y/Z-MP supercomputers are some examples of shared memory, tightly coupled Multiple Instruction Multiple Data (MIMD) machines of the 1980s which incorporate dedicated/bus oriented interconnection [7] but having a limited number of processing elements.

The tremendous growth in micro-electronics technology coupled with advances in computer architecture has brought a new wave of powerful 32/64-bit RISC based microprocessors such as the Pentium, the PowerPC and the DEC Alpha AXP series computers. These microprocessors form the corner stone of new scalable multicomputer architectures that provide super performance at lower costs compared to the supercomputers by combining powerful concepts like superscalar processing, vector processing and massively parallel processing. Some examples of these implementations include the IBM Scalable Power (SP) parallel systems, Cray's T3D, HP 9000 MPS10, the DEC MPP. The Digital's Cluster Parallel System 4000 uses 32 Alpha 3000/600 64-bit microprocessors to form a network commonly called as a *workstation farm* or UNIX cluster [8]. These can be used as high performance servers for computationally intensive jobs even as an alternative to mainframes and supercomputers. These systems are easily scalable to incorporate advances in technology and computing power. The design of such high performance machines involves several challenging problems in computer architecture, network topology selection, routing algorithms, communication hardware design, fault-tolerance, operating systems, parallel languages, compilers and software.

1.1 Interconnection networks

The study of interconnection networks has been an active area of research and the choice of a specific interconnection network greatly affects the performance/cost tradeoffs of a parallel computer. The advantages of using regular and structured interconnection networks have been exemplified in many practical implementations. The NCUBE/10 [9], Intel's iPSc [10], MARK II & III [11]-[13], the CM-5 [14], J-Machine [15], Maspar MP-1 [16], DECmpp 12000 [17], Convex C3880, VAX 9000-420 VP and Intel's Paragon XP/S [18] are a few illustrative examples. Over the years, a plethora of interconnection topologies, including *bus, ring, chordal ring, complete connection, mesh, torus, omega, lens, X-tree, fat trees, hypercube, generalized hypercubes, express cubes and cube-connected cycles* have been proposed in the open literature [19]-[21]. Interconnection networks can be static or dynamic. A static network is one whose nodes are connected by fixed point to point links pertaining to a given topology, e.g. rings, meshes and hypercubes. In a dynamic network, the connection pattern is determined on the fly, e.g. crossbar and shuffle-exchange networks [22].

The topology of an interconnection network determines many architectural features which play a major role in the performance of a machine. An ideal network design should normally possess low degree, small diameter, minimum average distance, regularity, modularity, expandability, ease of implementation, simple routing schemes, fault-tolerance and reconfiguration capability. In the case of bus oriented networks [23] such as the Symmetry and Multimax parallel computers, system performance is dependent on several factors such as the number of processor nodes, bus arbitration schemes, synchronization and cache coherence schemes. Notwithstanding the simplicity of implementation, bus-based parallel systems suffer from limited scalability, excessive delay and single point failures. Meshes, on the other hand, are scalable and have a constant degree but their diameter grows linearly with the number of nodes. This tends to degrade their performance especially for communication intensive jobs. Again a fully connected topology is not viable for system expansion beyond a certain point in view of the quadratic increase in the number of interconnections.

1.1.1 Symmetric Graphs

The hypercube [7] topology is a well known interconnection topology for concurrent computer systems. The hypercube or binary n-cube consists of a pair of nodes in each dimension connected to each other such that there are a total of 2^n nodes in the system. This regular and well structured interconnection network has been studied and used extensively

in many real time applications, e.g. signal/image processing, quantum physics and they provide significant speedups and throughput [12]. However, for embedded volume/mass critical applications in submersibles/space stations [24], e.g. payload controllers, smart sensors/ robotics and engineering subsystems for telemetry, hypercube networks do not permit *incremental expansion*. The network always expands as a power of 2 and the number of communication links for each node increases linearly with increasing dimensionality.

In a concurrent computer system, the interconnection topology and the allocation of processing elements to meet certain tasks is dictated by the application requirements. In some cases, an increase in performance may be the objective. In airborne computers, for instance, fault tolerance and space-weight limitations are major issues. Another important parameter of multicomputer systems is system expansibility and specifically the smallest increment by which the system can be expanded in a useful way. In all these instances the underlying interconnection network should be modular, flexible and lend itself to changes. Bhuyan and Agarwal [25, 26] introduced the concept of generalized hypercubes (GHC) and hyperbus based structures which are basically extensions of the hypercube structures. Similarly, symmetric interconnection networks based on group-theoretic models have been introduced in [27]. Some of these include the *star graph, pancake graph, Shuffle-Exchange graphs* and the *de Bruijn graphs* which belong to the class of *Cayley graphs* known for its structural properties. For the star graph, the degree and diameter grow at a much slower rate compared to hypercubes. The binary de Bruijn graph has a constant degree and does not grow with the size of the network.

Hypercycles proposed in [28] are *Cayley graphs* as we shall prove later. They are products of circulant graphs that allow a richer set of component sub-graphs as compared to hypercubes ranging in complexity from simple rings to a fully connected network. Hypercycles are therefore generalizations of several popular interconnection networks such as the rings, toruses, k-ary n-cube, hypercubes and generalized hypercubes. The hypercycles being regular graphs retain the advantages of simple routing and structural generality similar to the hypercubes. Since hypercycles represent a class of graphs rather than isolated graphs, they offer the flexibility of tailoring them to suit applications that demand computation, communication, fault-tolerance and reliability.

1.2 Motivation

This dissertation will address some of the architectural, algorithmic and communication issues outlined below and provide an engineering solution for the design of a cost-effective,

high performance parallel computer system that follows a hypercycle topology.

1.2.1 Architectural Issues

A major step in designing a highly efficient and reliable interconnection network for parallel computer systems is finding a topology with minimum diameter and average distance for a given set of constraints such as the number of nodes, degree and dimension of the network. Interconnection networks are evaluated in terms of *network latency* which is the average time taken for a message to reach its destination and in terms of the *network throughput*, which is the number of messages successfully routed per unit time in the network [29]. The diameter and average distance are two static metrics that are used to judge the performance of interconnection networks in a technology independent way. The diameter signifies the worst case distance for a message to travel between two nodes while the average distance denotes the average of the lengths of all paths taken by messages in the network. Although the diameter is important, there are other considerations which govern the design of an interconnection network, for instance, the degree, degree*diameter product, ease of routing, fault-tolerance and the switching methodology. It is therefore of paramount importance to obtain insight on optimal topologies that minimize some of the above mentioned metrics.

1.2.2 Algorithmic Issues

The ability of an interconnection network to simulate another network topology efficiently is often of considerable interest when designing parallel algorithms for a multi-computer. Also the communication patterns in parallel algorithms inherently favor certain topologies. For instance, matrix operations often inherently map to a mesh topology while divide-and-conquer algorithms take up a binary tree structure. Simulating one topology on another permits algorithms designed for the former to be mapped easily on the latter and this constitutes the problem of graph embedding. It will be of practical and theoretical interests to investigate and develop algorithms for embedding simple communication structures like cycles, Hamiltonian cycles and binary trees in hypercycles.

1.2.3 Communication Issues

The basic function of an interconnection network is to efficiently transfer information among the communicating nodes. In a broad sense, routing refers to the communicating methods and algorithms used to implement this function. The end-to-end communication delay or network latency is a critical factor in interconnection networks. The interconnection network

plays a key role in determining how fast applications can be executed in the system. Many real-time programs running on a multicomputer are I/O bound [29] and their execution on a multi-computer is often limited by the network's communication capability. The total execution time of an application in a massively parallel machine is composed of the computation time and communication delay or network latency as dictated by *Amdahl's law* [30] which is stated as follows:

Amdahl's Law: Given f_n, the portion of code that can run in parallel over n nodes, then the speed-up, s_n, achievable is given by

$$s_n = \frac{1}{(1 - f_n) + \frac{f_n}{n} + \sigma} \tag{1.1}$$

where σ is a monotonically increasing function of n that corresponds to the communication overhead.

In a massively parallel system, the communication overhead should be minimal and comparable to the computation time, and the system throughput should be sufficiently high to permit efficient message passing. Therefore, the network interconnecting the underlying processors and the routing algorithm dictate the performance of the parallel system.

An important issue in the design of routing algorithms is to prevent or avoid the possibility of deadlocks. Deadlocks occur as a result of cyclic waits for resources held by two or more communicating processes and hence need to be eliminated. The design and implementation of deadlock-free algorithms is an area of intense activity in interconnection networks design and our attention will be focussed on this aspect with respect to the hypercycle network.

1.3 Dissertation Outline

The dissertation is organized into the following parts:

1.3.1 Theoretical Issues

Problems in the topological design and analysis of interconnection networks lend themselves to elegant formulations in combinatorics, graph theory and group theory.

Chapter 2 deals with topological optimization on hypercycles. We have formulated the hypercycle network design as an integer programming problem and the optimization is conducted with respect to three important design parameters, namely, the *diameter*, *average distance* and the degree*diameter product. Analytical solutions are derived, wherever possible and these are compared with numerical results of the optimization.

Chapter 3 is concerned with an analysis on the methodology and cost/performance tradeoffs in mapping a complete binary tree onto hypercycles. An efficient algorithm for embedding complete binary trees in a hypercycle network with unit dilation, and unit load factor is presented. The optimal hypercycle is identified using numerical techniques and is shown to be cost-effective with an asymptotic expansion of 1.25 compared to unit dilation embeddings in hypercubes.

Chapter 4 is devoted to the problem of embedding Hamiltonian cycles in hypercycles. Hypercycles are proved to be Cayley graphs and a generalized m-ary gray code algorithm is discussed. Expressions for the ranking/unranking of Hamiltonian cycles are derived.

Chapter 5 addresses the problem of finding deadlock-free routing schemes. It begins with a survey of various routing schemes used in interconnection networks. Two routing schemes namely, the e-cube routing and mixed e-cube and backtracking algorithms are proposed for the hypercycles and shown to be deadlock preventing.

1.3.2 Implementation Issues

Chapter 6 deals with the implementation aspects of the e-cube algorithm. An architecture of a routing engine for circuit switching is proposed. This engine consists of a specialized router that implements a particular routing algorithm, a crossbar for switching input-output connections and a controller module that deliberates the function of the engine. The specialized router has been designed for the e-cube algorithm for a 4-dimensional hypercycle network and implemented in 1.2 μm CMOS technology. A novel framework for designing and prototyping routing controllers through a general purpose hardware description language called CoDeL is also introduced.

Chapter 7 concludes the dissertation with a summary of results and suggestions for further research.

1.4 Contributions

The contributions of this dissertation are briefly summarized below:

- An in-depth study of the optimization problem in hypercycles with respect to the average distance, degree and diameter and the discovery of some interesting anomalies in the topological structure.
- Combinatorial analysis of a generalized m-ary gray code algorithm for hypercycles.

- A novel unit dilation embedding scheme of a complete binary tree on a hypercycle with an asymptotic expansion of 1.25.
- Design and adaptation of deadlock-free routing algorithms for hypercycles.
- Hardware implementation of an e-cube router component.
- Development and software implementation of a language for describing and synthesizing controllers at the algorithmic/register transfer level.

Before delving into the details, we review some of the basic concepts of graph theory and introduce the mathematical definitions and terminology associated with hypercycles.

1.5 Definitions

A simple, finite, undirected graph [31] is represented by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}(\mathcal{G})$ is the *node set* while $\mathcal{E}(\mathcal{G})$ denotes the *edge set*. The *cartesian product* $\mathcal{G}$ of two graphs G_1 and G_2 is given by $G_1 \times G_2$ where $V(G) = V(G_1) \times V(G_2)$ and two nodes (u_1, u_2) and (v_1, v_2) of $\mathcal{G}$ are adjacent if and only if either $u_1 = v_1$ and $(u_2, v_2) \in E(G_2)$ or $u_2 = v_2$ and $(u_1, v_1) \in E(G_1)$. The *connectivity*[†] is defined as the minimum number of edges of $\mathcal{G}$ whose removal disconnects $\mathcal{G}$. Two nodes v_i and v_j are *connected* if there is a path from v_i to v_j. The *degree* d of a node v in a graph $\mathcal{G}$ is the number of edges of $\mathcal{G}$ incident to v. A graph $\mathcal{G}$ is *regular* if all nodes have the same degree. A *matching* of a graph $\mathcal{G}$ is any subset of edges of $\mathcal{E}$ such that no two elements are adjacent. A maximum-cardinality matching is a matching which contains maximum number of edges. A *perfect matching* is a matching in which every node of the graph is an end-point of some element of the matching.

A *path* in a graph $\mathcal{G}$ is a sequence of distinct nodes $v_1, v_2, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E(G)$ for $i = 1 \ldots k-1$. If $(v_k, v_1) \in E(G)$, then the path is called a *cycle*. A *Hamiltonian cycle* of $\mathcal{G}$ is a cycle in which each node is visited exactly once.

Two graphs G and H are *isomorphic* if there is a bijection $\phi : V(G) \rightarrow V(H)$ such that $(x, y) \in E(G)$ iff $(\phi(x), \phi(y)) \in E(H)$. An *automorphism* of G is an isomorphism ϕ from G to itself. A graph is said to be *node transitive* or *node symmetric*, if for any two nodes $x, y \in V(G)$, there is an automorphism ϕ in G such that $\phi(x) = y$.

Let $(S, *)$ be a finite abelian *group* where '*' denotes the binary operation in S. Let T be a subset of S such that the identity element *id* of $S \notin T$ and $\forall \omega \in T$, $\omega^{-1} \in T$. A *Cayley graph* C= G(S,T) is defined as a graph whose node set is given by S and the edge set of C consists of pairs $(g, \omega * g)$ where $g \in S$ and $\omega \in T$. In other words if $(a, b) \in E(C)$

[†]Unless otherwise mentioned, connectivity refers to the edge connectivity of $\mathcal{G}$.

then $a^{-1} * b \in T$. The absence of id in T guarantees that there are no self loops in C. Furthermore the existence of inverse in S makes the graph undirected.

1.5.1 Hypercycle graphs

The Mixed Radix Number (MRN) system [26] is used in defining the topology of a hypercycle graph. Given a decimal number M which can be expressed as a product of r factors such that $M = \prod_{j=1}^{r} m_j$, then any number $0 \leq X \leq M-1$ can be expressed as a vector given by

$$\text{MRN}(X)_{m_1\ m_2\ \ldots\ m_r} = (x_1\ x_2\ \ldots\ x_r), \quad 0 \leq x_i \leq m_i - 1, \quad i = 1, \ldots r$$

Each x_i is chosen such that

$$X = \sum_{i=1}^{r} x_i w_i \quad \text{where } w_i = \frac{M}{\prod_{j=1}^{i} m_j}, \quad i = 1, \ldots, r \tag{1.2}$$

An r-dimensional hypercycle [32] is defined by $\mathcal{G}_{\vec{m}}^{\vec{\rho}} = \{\mathcal{V}_{\vec{m}}^{\vec{\rho}}, \mathcal{E}_{\vec{m}}^{\vec{\rho}}\}$ where

1. $\mathcal{V}_{\vec{m}}^{\vec{\rho}}$ is the set of nodes $\{0, 1, 2, \ldots M-1\}$ where $M = |\mathcal{G}_{\vec{m}}^{\vec{\rho}}|$.
2. $\mathcal{E}_{\vec{m}}^{\vec{\rho}}$ are the edges in the graph;
3. $\vec{m} = [m_1\ m_2\ \ldots\ m_r]$ is a basis vector such that $M = \prod_{i=1}^{r} m_i$ and $m_i \geq 2$ are integers;
4. $\vec{\rho} = [\rho_1\ \rho_2\ \ldots\ \rho_r]$ is the connectivity vector where $1 \leq \rho_i \leq \lfloor \frac{m_i}{2} \rfloor$.

If $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ has vertices (α, β) such that

$$(\alpha)_{m_1 m_2 \ldots m_r} = \alpha_1\ \alpha_2 \ldots \alpha_i \ldots \alpha_r, \quad 0 \leq \alpha_i \leq m_i - 1$$

$$(\beta)_{m_1 m_2 \ldots m_r} = \beta_1\ \beta_2 \ldots \beta_i \ldots \beta_r, \quad 0 \leq \beta_i \leq m_i - 1$$

then $(\alpha, \beta) \in \mathcal{E}_{m}^{\vec{\rho}}$ iff there exists some j for $1 \leq j \leq r$ such that

$$\beta_j = (\alpha_j \pm \xi_j) \bmod m_j, \quad 1 \leq \xi_j \leq \rho_j \tag{1.3}$$

$$\alpha_i = \beta_i \quad \text{for } i \neq j, \forall i \in \{1, \ldots, r\} \tag{1.4}$$

The degree d [33] of a hypercycle is given by

$$d = \sum_{i=1}^{r} d_i \tag{1.5}$$

where d_i is the degree in the i^{th} dimension such that

$$d_i = \begin{cases} 2\rho_i & \text{if } 2\rho_i < m_i \\ 2\rho_i - 1 & \text{if } 2\rho_i = m_i \end{cases}$$

The diameter k of a hypercycle is given by

$$k = \sum_{i=1}^{r} k_i \tag{1.6}$$

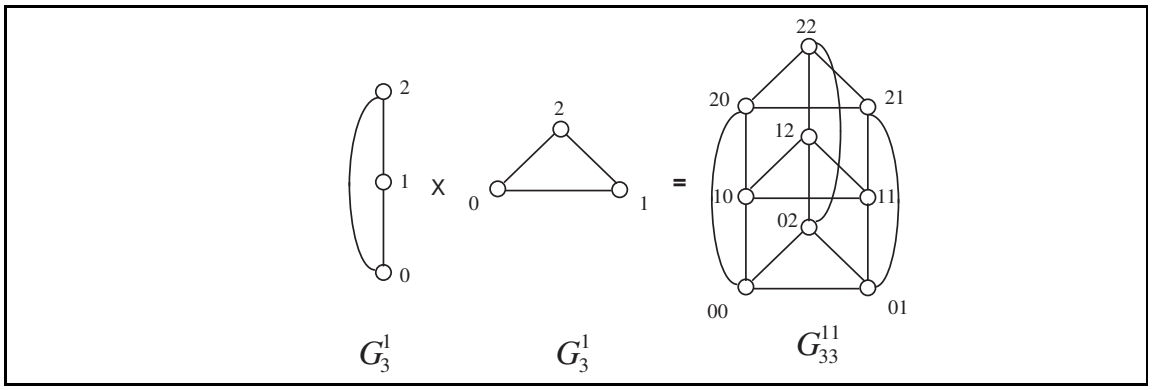


Figure 1.1 *Graph Multiplication in hypercycles*

where k_i is the diameter in the i^{th} dimension such that

$$k_i = \left\lceil \frac{\lfloor \frac{m_i}{2} \rfloor}{\rho_i} \right\rceil \tag{1.7}$$

The number of edges in $\mathcal{E}_{\vec{m}}^{\vec{\rho}}$ is given by

$$|\mathcal{E}_{\vec{m}}^{\vec{\rho}}| = \frac{Md}{2}$$

Hypercycles are cartesian products [34] of circulant graphs in each dimension and can be mathematically represented as:

$$\mathcal{G}_m^{\rho} = (\mathcal{G}_1)_{m_1}^{\rho_1} \times (\mathcal{G}_2)_{m_2}^{\rho_2} \times \ldots \times (\mathcal{G}_r)_{m_r}^{\rho_r} \tag{1.8}$$

The formation of a hypercycle through graph multiplication is shown in Fig. 1.1. Some typical examples of hypercycle graphs are illustrated in Fig. 1.2. By varying the connectivity in each dimension i, one can obtain different hypercycles that range from a simple cycle ($\rho_i = 1$) to a complete graph ($\rho_i = \lfloor \frac{m_i}{2} \rfloor$).

1.5.1.1 Routing in hypercycles

Hypercycles have routing properties [33, 35] that are similar to those of the binary n-cube. Each successive node address in a source destination path can be computed analytically as a function of the addresses of the current and destination nodes. Given two nodes

$$(\alpha)_{m_1 m_2 \ldots m_r} = \alpha_1 \alpha_2 \ldots \alpha_i \ldots \alpha_r$$

and

$$(\alpha^*)_{m_1 m_2 \ldots m_r} = \alpha_1 \alpha_2 \ldots \xi \ldots \alpha_r$$

whose addresses differ in the i^{th} position, it can be easily seen that the two nodes are separated by a distance of at most $\left\lceil \frac{\lfloor \frac{m_i}{2} \rfloor}{\rho_i} \right\rceil$ in each dimension i.

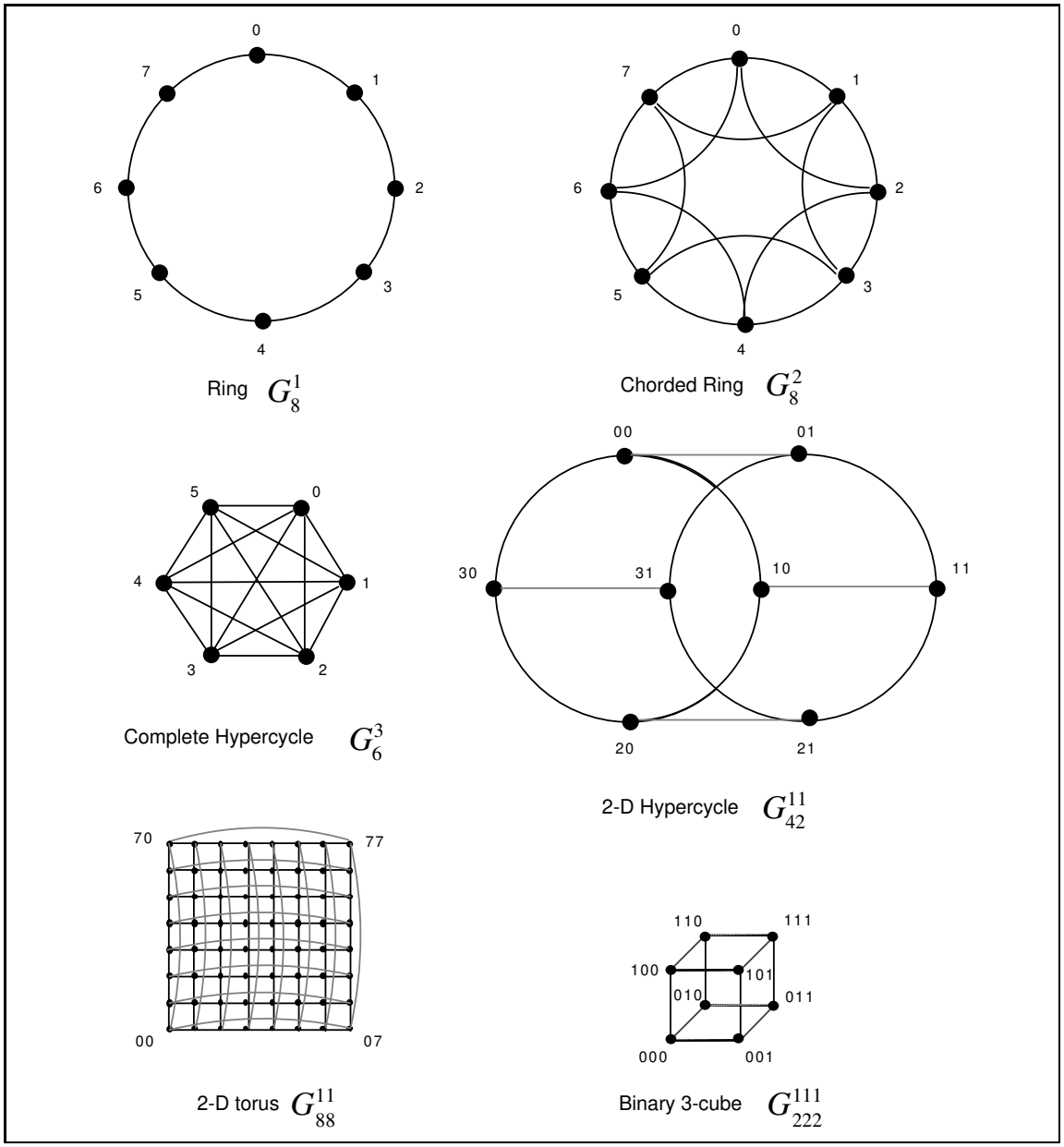


Figure 1.2 *Examples of hypercycle graphs*

A walk from node α to α^* can be constructed as follows:

$$\begin{aligned}
&\alpha_1\alpha_2 \ldots \alpha_i \ldots \alpha_r;\\
&\alpha_1\alpha_2 \ldots \xi_1 \ldots \alpha_r;\\
&\alpha_1\alpha_2 \ldots \xi_2 \ldots \alpha_r;\\
&\qquad\vdots\\
&\alpha_1\alpha_2 \ldots \xi \ldots \alpha_r;
\end{aligned}$$

such that[‡]

$$\xi_{j_i+1} = \begin{cases}
(\xi_{j_i} + \rho_i) \bmod m_i & \text{if } \left[(\xi - \xi_{j_i}) \bmod m_i = |\xi_{j_i}, \xi|\right] > \rho_i \quad \text{(a)}\\
(\xi_{j_i} + |\xi_{j_i}, \xi| \bmod \rho_i) \bmod m_i & \text{if } \left[(\xi - \xi_{j_i}) \bmod m_i = |\xi_{j_i}, \xi|\right] > \rho_i \quad \text{(b)}\\
& \text{and } |\xi_{j_i}, \xi| \bmod \rho_i \neq 0\\
(\xi_{j_i} - \rho_i) \bmod m_i & \text{if } \left[(\xi_{j_i} - \xi) \bmod m_i = |\xi_{j_i}, \xi|\right] > \rho_i \quad \text{(c)}\\
(\xi_{j_i} - |\xi_{j_i}, \xi| \bmod \rho_i) \bmod m_i & \text{if } \left[(\xi_{j_i} - \xi) \bmod m_i = |\xi_{j_i}, \xi|\right] > \rho_i \quad \text{(d)}\\
& \text{and } |\xi_{j_i}, \xi| \bmod \rho_i \neq 0\\
\xi & \text{if } |\xi_{j_i}, \xi| \leq \rho_i \quad \text{(e)}
\end{cases} \tag{1.9}$$

where ξ_{j_i} denotes the intermediate node in dimension i with $\xi_0 = \alpha_i$, $\xi_{max} = \xi$.

Equations (1.9 a-e) provide the analytical expressions for computing the minimal length paths to be taken from the source to the destination nodes in any dimension. Parts (a) and (c) of Eq. (1.9) constitute the *greedy strategy* since a maximum leap of ρ_i is taken towards the destination. On the other hand, parts (b) and (d) of Eq. (1.9) constitute a *liberal strategy* since it allows one shorter leap (less than ρ_i) towards the destination before the greedy strategy can be applied. We can prove this statement as follows:

Assume that the smaller leap is taken first from the current node ξ_{j_i} in dimension i by applying the liberal strategy (Eq. (1.9 b,d)). Then the next node address in the i^{th} dimension is given by

$$\xi_{j_i+1} = (\xi_{j_i} \pm |\xi_{j_i}, \xi| \bmod \rho_i) \bmod m_i \tag{1.10}$$

The remaining steps to reach the destination will be maximal (Eq. (1.9 a,c) since otherwise

$$|\xi_{j_i+1}, \xi| \bmod \rho_i \neq 0 \tag{1.11}$$

due to the liberal strategy. Expanding the L.H.S of Eq. (1.11), we have $|(\xi_{j_i} \pm |\xi_{j_i}, \xi| \bmod \rho_i) \bmod m_i, \xi| \bmod \rho_i$ and the following cases arise:

[‡] $|\mu, \nu| = \min\{(\mu - \nu) \bmod m, (\nu - \mu) \bmod m)\}$, the distance between two integers modulo m.

Case 1: $|\xi_{j_i}, \xi| = \xi - \xi_{j_i}$. Equation (1.9 b) applies and we obtain

$$|((\xi_{j_i} + \xi - \xi_{j_i}) \bmod \rho_i) \bmod m_i, \xi| \bmod \rho_i = 0$$

Case 2: $|\xi_{j_i}, \xi| = \xi_{j_i} - \xi$. Using of Eq. (1.9 d), we have

$$|((\xi_{j_i} - \xi_{j_i} + \xi) \bmod \rho_i) \bmod m_i, \xi| \bmod \rho_i = 0$$

which is a contradiction. This guarantees that the remaining leaps to the destination are maximal. As a consequence, it can be observed that the liberal strategy can be used only once in dimension i during minimal path routing towards the destination.

Given a source node $(\alpha)_{m_1 m_2 \ldots m_r} = \alpha_1 \alpha_2 \ldots \alpha_r$ and a destination node $(\beta)_{m_1 m_2 \ldots m_r} = \beta_1 \beta_2 \ldots \beta_r$, if q_i denotes the distance along dimension i, then the total distance $\mathrm{dis}(\alpha, \beta)$ is given by

$$\mathrm{dis}(\alpha, \beta) = q = \sum_{i=1}^{r} q_i$$

As explained in [33], there are a total of

$$\begin{aligned} I &= \begin{pmatrix} q \\ q_1, q_2, \ldots, q_r \end{pmatrix} \\ &= \frac{q!}{q_1! q_2! \ldots q_r!} \end{aligned}$$

possible[§] combinations of distinct walks of length q between α and β. These paths can be constructed by modifying the source address sequentially, each time substituting a source digit by an arbitrary walk digit chosen from the eligible dimension until the final destination is reached. The whole sequence is determined by the minimal length routing given by Eq. (1.9). As an illustration, the following walk connects the source α to the destination node β by successive modification of the address digits.

$$\begin{aligned} \textbf{source} = \; & \alpha_1 \alpha_2 \ldots \alpha_i \ldots \alpha_j \ldots \alpha_r; \\ & \alpha_1 \alpha_2 \ldots \xi_1 \ldots \alpha_j \ldots \alpha_r; \\ & \alpha_1 \alpha_2 \ldots \xi_1 \ldots \psi_1 \ldots \alpha_r; \\ & \alpha_1 \alpha_2 \ldots \xi_2 \ldots \psi_1 \ldots \alpha_r; \\ & \alpha_1 \alpha_2 \ldots \xi_2 \ldots \psi_2 \ldots \alpha_r; \\ & \qquad \vdots \\ & \alpha_1 \alpha_2 \ldots \beta_i \ldots \psi_2 \ldots \alpha_r; \\ & \qquad \vdots \\ & \beta_1 \beta_2 \ldots \beta_i \ldots \beta_j \ldots \beta_r \quad = \textbf{destination} \end{aligned}$$

[§]For the definition of a multinomial number, see [36, pp. 15].

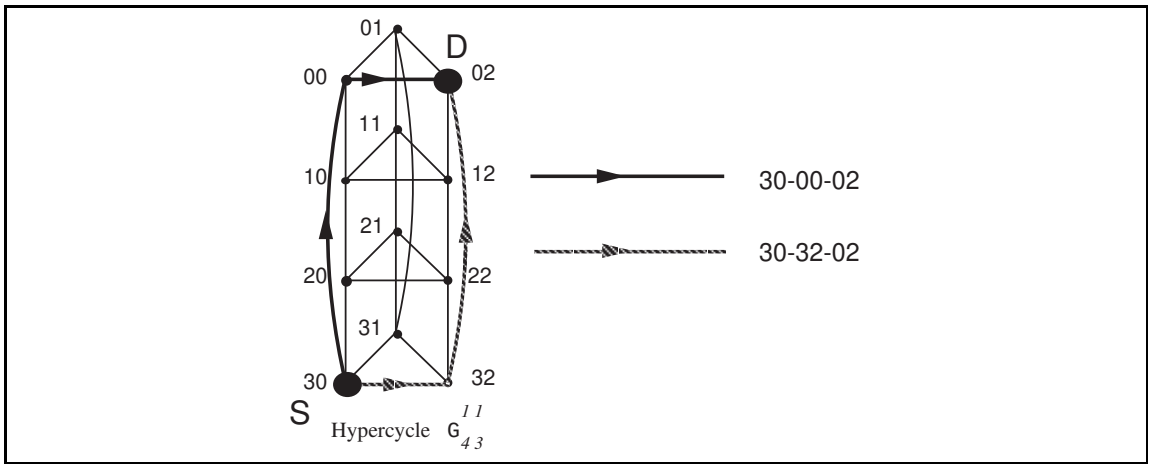


Figure 1.3 *Greedy routing in hypercycle* G_{43}^{11}

If the greedy strategy is followed, then the so formed paths are called greedy paths. Fig. 1.3 gives an example of a hypercycle $\mathcal{G}_{43}^{11}$ which shows two distinct greedy paths of equal minimum length that connect a source to a destination in a hypercycle. If the source and destination addresses are 30 and 02 respectively, then the possible routes determined by Eq. (1.9) are:

1. 30-32-02
2. 30-00-02.

1.6 Conclusion

In this Chapter, a brief introduction to parallel computers and the motivation for developing efficient interconnection networks was established. The scope and contributions of this dissertation was outlined which will be discussed in detail in the subsequent Chapters.

Chapter 2

Topological Optimization in Hypercycles

2.1 Introduction

A major factor in realizing a highly efficient and reliable network is finding a graph with minimum diameter and average distance and maximal connectivity for a given set of nodes and degree [37]. The diameter and average distance are related to the network latency while connectivity is related to the degree and fault-tolerance of the system. In a massively parallel system, the communication overhead should be minimal and comparable to the computation time, and the system throughput should be sufficiently high to permit data sharing. Technological constraints limit the number of I/O ports in each node of a parallel computer and this places an upper bound on the communication bandwidth. From the stand point of logical and physical realizability, it is advantageous to have minimum degree but at the cost of increasing the network latency. Therefore, a pragmatic and viable approach to design a network system with high performance and minimum cost, should consider both connectivity and diameter. This often necessitates the introduction of optimization techniques in network design [38]-[40].

In [25], Bhuyan and Agarwal have made an analysis in realizing a cost-optimal network based on the degree and diameter of the Generalized Hypercube (GHC) and Hyperbus structures. Dally [29] considered the optimization of the network latency and throughput for k-ary n-cubes incorporating worm-hole routing.

In this Chapter, we present an analysis that establishes the optimality of balanced hypercycles (those which have equal node population and degree in each dimension) with respect to the average distance, diameter and product of degree and diameter. However, it is shown that under certain conditions, the optimum corresponds to hypercycles which are not

balanced. Section 2.2 deals with the formulation of the average distance as an optimization problem in hypercycle networks. In Section 2.3, the optimization of the average distance and diameter is considered. Analytical solutions are presented and compared with numerical results. Section 2.4 considers special cases of the formulated optimization problem while the product of degree and diameter is analyzed in Section 2.5. Section 2.6 briefly discusses network latency and throughput in hypercycles. Finally, the Chapter concludes with a short discussion of the obtained results and their significance.

Definition 2.1 *An r-dimensional hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is said to be* **balanced** *if the node radix and connectivity vectors are such that $m_1 = m_2 = \ldots = m_r$ and $\rho_1 = \rho_2 = \ldots = \rho_r$.*

Definition 2.2 *The connectivity-sum, C, of a hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is defined as the sum of its connectivity in each dimension such that*

$$C = \sum_{i=1}^{r} \rho_i \geq \frac{d}{2}$$

Definition 2.3 *The solution space Ω of the node radix vector $\vec{m}$ for a given M is the set of all possible integer values of $m_1, m_2, \ldots, m_r$ such that $M = \prod_{i=1}^{r} m_i$.*

Definition 2.4 *The neighbourhood of a point, $\vec{m}^* = [m_1\ m_2\ \ldots\ m_r]$ in the solution space Ω of the node radix vector for a given $\vec{\epsilon}$ is a region defined by*

$$\mathcal{R}_{\vec{m}^*}^{\vec{\epsilon}} = \left\{ \vec{m} : \frac{|m_i - m_i^*|}{m_i^*} \leq \epsilon_i, \vec{\epsilon} = [\epsilon_1\ \epsilon_2\ \ldots\ \epsilon_i\ \ldots\ \epsilon_r], i = 1, \ldots, r \right\}$$

where m_i is an integer.

2.2 Formulation of Optimal Hypercycle Network Design

The hypercycle network, as a massively parallel system, requires high throughput and low network latency to achieve maximum performance. The average distance is one of the factors that affect the network latency and throughput [25, 29] and therefore minimizing the average distance to arrive at an optimal network design is of considerable interest. Since the node radix vector, $\vec{m}$, and connectivity vector $\vec{\rho}$ of an r-dimensional hypercycle can assume only discrete values, the optimization will involve integer design variables. In addition, physical realizability of a network system imposes constraints on the design parameters. The formulation of a nonlinear integer programming problem for average distance is presented subsequently.

Average Distance: For a hypercycle, given a source node $(\alpha)_{m_1 m_2 \ldots m_r} = \alpha_1 \alpha_2 \ldots \alpha_r$ and a destination node $(\beta)_{m_1 m_2 \ldots m_r} = \beta_1 \beta_2 \ldots \beta_r$, if $\Psi(\beta_i, \alpha_i)$ denotes the shortest distance

separating α and β in dimension i, then the Hamming distance between α and β is given by $H_D(\alpha,\beta) = \sum_{i=1}^{r} \Psi(\beta_i, \alpha_i)$.

The average distance, δ, of a hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is defined as the normalized sum of the Hamming distances of all nodes in the graph with respect to a reference node. This can be computed by summing the distance of each node of $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ with respect to a reference node in each dimension i, for $i = 1, \ldots, r$. Since hypercycles are symmetrical graphs by definition, it can be assumed without loss of generality that node $\alpha = \underbrace{(0,0,0\ldots,0)}_{r}$ is the reference node. Furthermore, let $\beta = (\beta_1, \beta_2, \ldots, \beta_r)$ be any arbitrary node. If we consider the distance along dimension j between α and nodes $\beta' = \underbrace{(X, X, \ldots, \beta_j, X, X, \ldots X)}_{r}^{\dagger}$, then there are $\prod_{i=1, i\neq j}^{r} m_i$ nodes having address β_j in the j^{th} dimension. Therefore the total distance along dimension j for all these β' nodes is given by $\Psi(\beta_j, 0).\prod_{i=1,i\neq j}^{r} m_i$. Since $0 \leq \beta_j \leq m_j - 1$, the total distance in the j^{th} dimension between α and all nodes in $\mathcal{G}_m^{\rho}$ is given by $\sum_{\beta_j=0}^{m_j-1} \Psi(\beta_j, 0) \prod_{i=1,i\neq j}^{r} m_i$. This is summed over r dimensions and the total distance is given by $\sum_{j=1}^{r}\sum_{\beta_j=0}^{m_j-1} \Psi(\beta_j, 0) \prod_{i=1,i\neq j}^{r} m_i$. Since the total distance from the reference node is calculated for $M-1$ remaining nodes, we normalize it with respect to $M-1$ to obtain the required average distance.

The expression for average distance is given as follows:

$$\delta = \frac{\sum_{j=1}^{r} S_j}{M-1} \tag{2.1}$$

where

$$S_j = f(m_j, \rho_j) \prod_{i=1, i\neq j}^{r} m_i \tag{2.2}$$

$$f(m_j, \rho_j) = \sum_{\beta_j=0}^{m_j-1} \Psi(\beta_j, 0) \tag{2.3}$$

$$\Psi(\beta_j, 0) = \begin{cases} \left\lceil \frac{\beta_j}{\rho_j} \right\rceil & \text{if } \beta_j \leq \lfloor \frac{m_j}{2} \rfloor \\ \left\lceil \frac{m_j - \beta_j}{\rho_j} \right\rceil & \text{otherwise} \end{cases} \tag{2.4}$$

The variable S_j denotes the distance of all nodes in the j^{th} dimension from the reference node. The function $\Psi(\beta_j, 0)$, defined by Eq. (2.4), gives the shortest distance in dimension j for node β with respect to the reference node (0) and is computed using the *greedy strategy* outlined in [35]. The term $f(m_j, \rho_j)$ denotes the sum of the distances of the 1-dimensional hypercycle $\mathcal{G}_{m_j}^{\rho_j}$. The problem of determining optimal hypercycle network configurations

[†] X denotes a don't care address digit.

which minimize the average distance can now be formulated as follows:

Case 1: Given the total number of nodes M, the number of dimensions r and the connectivity sum C (which loosely corresponds to the degree of the network), find the node radix vector $\vec{m} = [m_1\ m_2\ \ldots\ m_r]$ and the connectivity vector $\vec{\rho} = [\rho_1\ \rho_2\ \ldots\ \rho_r]$ that define the graph with the minimum average distance.

$$\min_{w.r.t\ m_j,\ \rho_j} \delta(m_j,\ \rho_j) = \frac{1}{M-1}\sum_{j=1}^{r} S_j \quad \text{subject to}$$

$$\begin{aligned} h_1 &= M - \prod_{j=1}^{r} m_j = 0 \\ h_2 &= C - \sum_{j=1}^{r} \rho_j = 0 \\ g_i &= 1 - \rho_i \leq 0 \quad (i = 1, \ldots, r) \\ g_{i+r} &= 2 - m_i \leq 0 \end{aligned}$$

The above general problem can also be specialized as the following two problems.

Case 2: Given the connectivity vector $\vec{\rho}$, the number of dimensions r and the total number of nodes M, find the node radix vector $\vec{m}$, that will minimize the average distance.

$$\min_{w.r.t\ m_j} \delta(m_j) = \frac{1}{M-1}\sum_{j=1}^{r} S_j \quad \text{subject to}$$

$$\begin{aligned} h_1 &= M - \prod_{j=1}^{r} m_j = 0 \\ h_{2+i} &= \rho_i = c_i \quad (i = 1, \ldots, r) \\ g_i &= 2 - m_i \leq 0 \end{aligned}$$

where c_i is some integer constant bounded from above by $\left\lfloor \frac{M}{2^r} \right\rfloor$.

Case 3: Given the node radix vector $\vec{m}$, the number of dimensions r and connectivity-sum C, determine the connectivity vector $\vec{\rho}$ that minimizes the average distance.

$$\min_{w.r.t\ \rho_j} \delta(\rho_j) = \frac{1}{M-1}\sum_{j=1}^{r} S_j \quad \text{subject to}$$

$$\begin{aligned} h_i &= m_i = c_i \quad (i = 1, \ldots, r) \\ h_{r+i} &= C - \prod_{j=1}^{r} \rho_j = 0 \end{aligned}$$

2.3 Optimization of the average distance and diameter

In this section, we examine the optimization problem as presented in Case 1 earlier and establish the structure of hypercycles which have minimum average distance (diameter) given a fixed number of nodes, connectivity sum, and assuming that the number of dimensions is a given constant. Since the objective functions are discrete, we compute the optimum in two stages:

- Firstly, we establish continuous upper and lower bounds and determine the optimum point with regard to those bounds.
- Secondly, small perturbations are applied to the optimal point derived for the bounds and we prove that these perturbations lead to non-optimal points for the original discrete function.

2.3.1 Bounds for average distance and diameter

The assumptions of Case 1, i.e, $\prod_{j=1}^{r} m_j = M$ and $\sum_{j=1}^{r} \rho_j = C$ hold good for Lemmas 2.1-2.5 which are given subsequently.

Lemma 2.1 *The average distance function δ given by Eq. (2.1) is bounded from above and below by*

$$\hat{\delta} = \frac{M}{(M-1)} \left(\frac{m_1}{4\rho_1} + \sum_{j=2}^{r} \frac{m_j}{4\rho_j} + r \right) \quad \textit{and} \tag{2.5}$$

$$\check{\delta} = \frac{M}{(M-1)} \left(\frac{m_1}{4\rho_1} + \sum_{j=2}^{r} \frac{m_j}{4\rho_j} - \frac{r}{4} \right) \tag{2.6}$$

respectively

Proof. Refer to Section A.1 in Appendix A. Fig 2.1 gives a plot of the average distance along with its bounds defined by the above equations for a 1-dimensional hypercycle with $\rho = 2$. ■

Lemma 2.2 *The cost optimal point for the upper and lower bounds of the average distance function, namely, $\hat{\delta}$ and $\check{\delta}$ as specified by Case 1 is $m_i = \sqrt[r]{M}$, $\rho_i = \frac{C}{r}$, $i = 1, \ldots, r$ where r, M and C are given constants.*

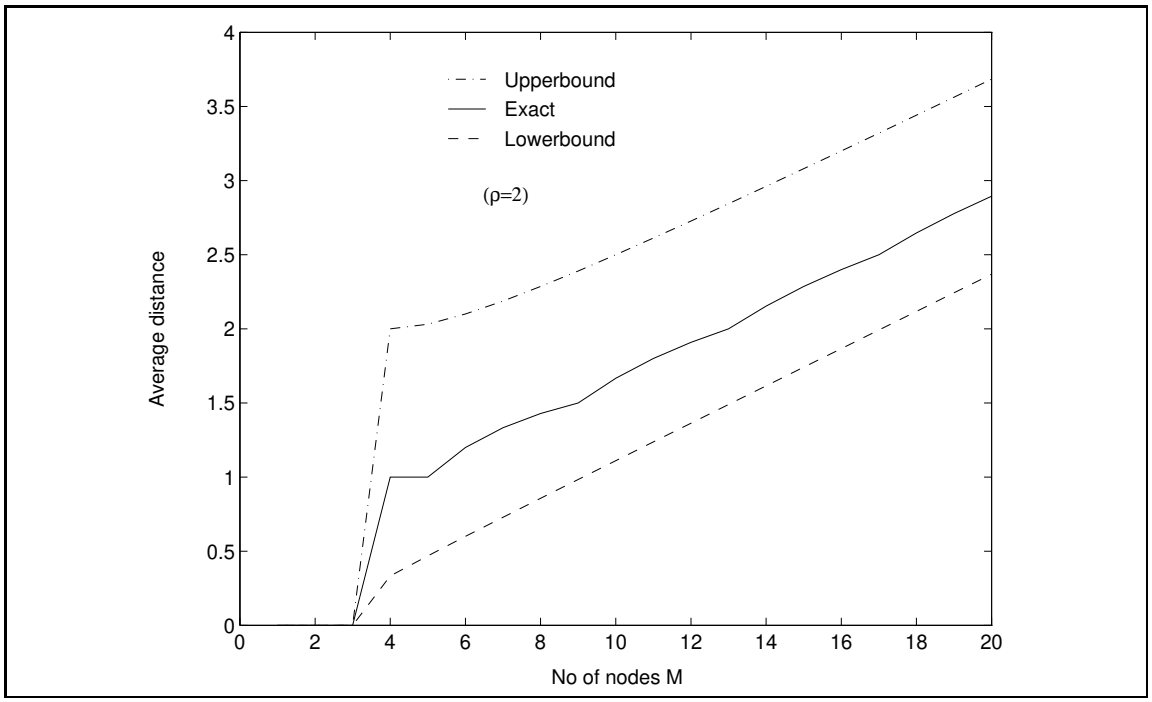


Figure 2.1 *Bounds of the average distance for $\rho = 2$ in 1-dimensional hypercycles.*

Proof. Substituting $m_1 = M/(m_t \prod_{j=2,j\neq t}^{r} m_j)$ in Eq. (2.5) where $2 \leq t \leq r$, we obtain

$$\hat{\delta} = \frac{M}{M-1}\left(\frac{M}{4\rho_1 m_t \prod_{j=2,j\neq t}^{r} m_j} + \frac{m_t}{4\rho_t} + \sum_{j=2,j\neq t}^{r} \frac{m_j}{4\rho_j} + r\right) \tag{2.7}$$

To find the minimum of $\hat{\delta}$, the first order necessary condition given by $\bigtriangledown\hat{\delta}(m^*, \rho^*) = 0$ has to be satisfied by the optimal point. Computing the derivative of $\hat{\delta}$ with respect to m_t, we have

$$\frac{\partial\hat{\delta}}{\partial m_t} = \frac{M}{4(M-1)}\left(\frac{-M}{m_t^2\rho_1 \prod_{j=2,j\neq t}^{r} m_j} + \frac{1}{\rho_t}\right) = 0$$

i.e.,

$$\frac{m_1}{m_t} = \frac{\rho_1}{\rho_t} \tag{2.8}$$

and

$$\frac{\partial^2\hat{\delta}}{\partial m_t^2} = \frac{M}{4(M-1)}\left(\frac{2m_1}{\rho_1 m_t^2}\right) > 0, \quad \forall m_t \geq 2$$

Substituting $\rho_1 = C - \rho_t - \sum_{j=2,j\neq t}^{r} \rho_j$ in Eq. (2.5), we obtain

$$\hat{\delta} = \frac{M}{M-1}\left(\frac{m_1}{4(C - \rho_t - \sum_{j=2,j\neq t}^{r} \rho_j)} + \frac{m_t}{4\rho_t} + \sum_{j=2,j\neq t}^{r} \frac{m_j}{4\rho_j} + r\right) \tag{2.9}$$

Taking the derivative of Eq. (2.9) with respect to ρ_t, $2 \leq t \leq r$, yields

$$\frac{\partial \hat{\delta}}{\partial \rho_t} = \frac{m_1}{4\rho_1^2} - \frac{m_t}{4\rho_t^2} = 0$$

i.e.,

$$\frac{m_1}{m_t} = \frac{\rho_1^2}{\rho_t^2} \tag{2.10}$$

and

$$\frac{\partial^2 \hat{\delta}}{\partial \rho_t^2} = \frac{M}{(M-1)} \left(\frac{m_t}{2\rho_t^3} \right) > 0, \quad \forall \rho_t \geq 1$$

From Eq. (2.8) and (2.10), we obtain

$$\frac{\rho_1}{\rho_t} = \frac{\rho_1^2}{\rho_t^2} \quad \text{and}$$

$$\frac{\rho_1}{\rho_t} \left(\frac{\rho_1}{\rho_t} - 1 \right) = 0$$

which leads to

$$\rho_1 = \rho_t$$

$$m_1 = m_t$$

It therefore follows that $\hat{\delta}$ is a minimum when $\rho_1 = \rho_2 = \ldots = \rho_r = \frac{C}{r}$ and $m_1 = m_2 = \ldots = m_i = \ldots = m_r = \sqrt[r]{M}$, $i = 1, \ldots, r$. In a similar manner, it can be shown that the minimum of $\check{\delta}$ given by Eq. (2.6) is identical to that of $\hat{\delta}$. ■

Lemma 2.3 *The diameter of the hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is bounded from above by $\hat{k} = \sum_{i=1}^{r} \frac{m_i}{2\rho_i} + r$.*

Proof. From Eq. (1.6), $k = \sum_{i=1}^{r} \left\lceil \frac{\lfloor \frac{m_i}{2} \rfloor}{\rho_i} \right\rceil$. Hence $k \leq \sum_{i=1}^{r} \left\lceil \frac{\frac{m_i}{2}}{\rho_i} \right\rceil \leq \sum_{i=1}^{r} \left(\frac{m_i}{2\rho_i} + 1 \right) = \sum_{i=1}^{r} \frac{m_i}{2\rho_i} + r$. Therefore $\hat{k} = \sum_{i=1}^{r} \frac{m_i}{2\rho_i} + r$. ■

Lemma 2.4 *The diameter of the hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is bounded from below by $\check{k} = \sum_{i=1}^{r} \frac{m_i}{2\rho_i} - r$.*

Proof. We know that $k = \sum_{i=1}^{r} \left\lceil \frac{\lfloor \frac{m_i}{2} \rfloor}{\rho_i} \right\rceil$. Hence $k \geq \sum_{i=1}^{r} \left\lceil \frac{\frac{m_i}{2} - 1}{\rho_i} \right\rceil \geq \sum_{i=1}^{r} \left(\frac{m_i}{2\rho_i} - \frac{1}{\rho_i} \right) \geq \sum_{i=1}^{r} \frac{m_i}{2\rho_i} - r$. Therefore $\check{k} = \sum_{i=1}^{r} \frac{m_i}{2\rho_i} - r$. ■

Lemma 2.5 *The cost optimal point for the upper and lower bounds of the diameter, $\hat{k}$ and $\check{k}$ is given by $m_i = \sqrt[r]{M}$, $\rho_i = \frac{C}{r}$, $i = 1, \ldots, r$ where r, M and C are given constants.*

Proof. From Lemma 2.3, we know that $\hat{k} = \sum_{i=1}^{r} \frac{m_i}{2\rho_i} + r = \frac{m_1}{2\rho_1} + \sum_{j=2}^{r} \frac{m_j}{2\rho_j} + r$. Substituting, $m_1 = M/(m_t \prod_{j=2, j\neq t}^{r} m_j)$, $2 \leq t \leq r$, we get

$$\hat{k} = \frac{M}{2\rho_1 m_t \prod_{j=2, j\neq t}^{r} m_j} + \frac{m_t}{2\rho_t} + \sum_{j=2, j\neq t}^{r} \frac{m_j}{2\rho_j} + r$$

Evaluating the first order derivative of $\hat{k}$ w.r.t m_t, we obtain

$$\frac{\partial \hat{k}}{\partial m_t} = \frac{-M}{2m_t^2 \rho_1 \prod_{j=2, j\neq t}^{r} m_j} + \frac{1}{2\rho_t} = 0 \quad \text{and hence}$$

$$\frac{m_1}{m_t} = \frac{\rho_1}{\rho_t} \tag{2.11}$$

With $\rho_1 = C - \rho_t - \sum_{j=2, j\neq t}^{r} \rho_j$, $2 \leq t \leq r$, we have

$\hat{\delta} = \left(\frac{m_1}{2\left(C - \rho_t - \sum_{j=2, j\neq t}^{r} \rho_j\right)} + \frac{m_t}{2\rho_t} + \sum_{j=2, j\neq t}^{r} \frac{m_j}{2\rho_j} + r \right)$ and evaluating the derivative of $\hat{k}$ with respect to ρ_t yields

$$\frac{\partial \hat{\delta}}{\partial \rho_t} = \frac{m_1}{2\rho_1^2} - \frac{m_t}{2\rho_t^2} = 0 \tag{2.12}$$

From Eqs. (2.11) and (2.12), we obtain $m_1 = m_t$, $\rho_1 = \rho_t$ and hence $k_1 = k_t$, for $t = 2, \ldots, r$. Likewise an identical result can be obtained for $\check{k}$ as well. ■

2.3.2 Perturbation Analysis

In this section, we examine whether the optimal point obtained for the bounds in Section 2.3.1 remain optimal for the original discrete functions, namely the average distance and diameter. We establish the optimality by calculating the value of the discrete function in the neighbourhood of the optimal point and showing that this value is always larger than the value at the optimal point. While the global optimality of the balanced point can be established only by considering perturbations in r-dimensions, this approach becomes extremely difficult and complex to tackle analytically. The perturbation analysis dealt with in this section will be restricted to two dimensions and this will provide confidence to some degree on the optimality of the point at least locally. Before proceeding further, we state Lemmas 2.6, 2.7 and 2.8 which establish some intermediate computation results and which will be used in establishing our main result. The proofs of these Lemmas are given in Sections B.1 and B.2 of Appendix B.

Lemma 2.6

$$f(m,\rho) = \begin{cases} 2\sum_{i=1}^{\frac{m-1}{2}} \left\lceil \frac{i}{\rho} \right\rceil & \text{if } m \text{ is odd} \\ 2\sum_{i=1}^{\frac{m}{2}} \left\lceil \frac{i}{\rho} \right\rceil - \left\lceil \frac{m}{2\rho} \right\rceil & \text{if } m \text{ is even} \end{cases} \tag{2.13}$$

Lemma 2.7

$$\sum_{i=1}^{n} \left\lceil \frac{i}{\rho} \right\rceil = \frac{n}{2} + \frac{n^2}{2\rho} + \frac{X(\rho - X)}{2\rho} \quad \text{where } X = n \bmod \rho \tag{2.14}$$

Lemma 2.8 *For* $1 \leq \rho \leq \lfloor \frac{m}{2} \rfloor$ *and* $\lfloor \frac{m}{2} \rfloor \bmod \rho = 0$, *then* $f(m,\rho)$ *can be expressed as*

$$f(m,\rho) = \begin{cases} \frac{m-1}{2} + \frac{(m-1)^2}{4\rho} & \text{if } m \text{ is odd} \\ \frac{m}{2} + \frac{m(m-2)}{4\rho} & \text{if } m \text{ is even} \end{cases} \tag{2.15}$$

Theorem 2.1 *Given a hypercycle* $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ *and assuming that* $\prod_{j=1}^{r} m_j = M$ *and* $\sum_{j=1}^{r} \rho_j = C$ *where* m_i *and* ρ_i *are integers, then the average distance is cost-optimal for* $m_i = \sqrt[r]{M}$, $\rho_i = \frac{C}{r}$, *and* $\lfloor \frac{m_i}{2} \rfloor \bmod \rho_i = 0$, *for* $i = 1, \ldots, r$ *when the dimension* r, M *and* C *are given integer constants and assuming that* $\sqrt[r]{M}$ *and* $\frac{C}{r}$ *are integers.*

Proof. For the balanced hypercycle, let $m_j = m$ and $\rho_j = \rho$, $(j = 1, \ldots, r)$. If δ_b denotes its corresponding average distance, then we can express

$$\begin{aligned} \delta_b &= \frac{\sum_{j=1}^{r} S_j}{M-1} \\ &= \frac{rS}{M-1} \end{aligned} \tag{2.16}$$

where

$$\begin{aligned} S &= f(m,\rho) \prod_{k=1, k\neq j}^{r} m_k \\ &= m^{r-1} f(m,\rho) \end{aligned} \tag{2.17}$$

Substituting Eq. (2.17) in Eq. (2.16), the average distance for the balanced hypercycle is given by

$$\delta_b = \frac{r m^{r-1} f(m,\rho)}{M-1} \tag{2.18}$$

Suppose we move away from $m_1 = m_2 = \ldots = m_r = m$ to a point m^* given by $\underbrace{m, \ldots, m+\upsilon, \ldots, m-\lambda, \ldots, m}_{r}$ where λ and υ are integer variables such that $m^r = (m+\upsilon)(m-\lambda)m^{r-2} \Rightarrow m^2 = (m+\upsilon)(m-\lambda)$ then $m\upsilon = \lambda(m+\upsilon)$ which implies

$$\lambda = \frac{\upsilon}{1 + \frac{\upsilon}{m}} \tag{2.19}$$

$$\upsilon - \lambda = \frac{\upsilon\lambda}{m} > 0 \quad \text{(assuming that both } \upsilon, \lambda \neq 0) \tag{2.20}$$

Observe that $0 \leq \lambda \leq m-2$ and $0 \leq \upsilon \leq \frac{m}{2}(m-2)$ when $m \geq 2$. It is worth noting that $\upsilon \neq 1$ since otherwise $\lambda = \frac{m}{m+1} < 1$. Thus there exists no positive integer λ which would satisfy Eq. (2.19) when $\upsilon = 1$. By a similar token, $\lambda \neq 1$.

The average distance calculated at the perturbed point m^* is given by

$$\begin{aligned} \delta' &= \frac{1}{M-1}\Big[(r-2)m^{r-3}m^2 f(m,\rho) + m^{r-2}(m+\upsilon)f(m-\lambda,\rho) \\ &\qquad + m^{r-2}(m-\lambda)f(m+\upsilon,\rho)\Big] \end{aligned} \tag{2.21}$$

The difference between δ' and δ_b is given by

$$\begin{aligned} \delta' - \delta_b &= \frac{1}{M-1}\Big[(r-2).m^{r-3}(m+\upsilon)(m-\lambda).f(m,\rho) + m^{r-2}(m+\upsilon)f(m-\lambda,\rho) \\ &\qquad + m^{r-2}(m-\lambda).f(m+\upsilon,\rho) - r.m^{r-1}f(m,\rho)\Big] \\ &= \frac{m^{r-2}}{M-1}\left[(m+\upsilon)f(m-\lambda,\rho) + (m-\lambda)f(m+\upsilon,\rho) - 2mf(m,\rho)\right] \end{aligned} \tag{2.22}$$

If δ_b is indeed a minimal point, then any one step perturbation in a 2-dimensional plane will result in a value of δ' being larger than δ_b, i.e., $\delta' - \delta_b \geq 0$ for a given ρ, $\forall \upsilon, \lambda \geq 0$. Note that when the diameter k_i is an integral multiple of the connectivity ρ_i in each dimension i, $1 \leq i \leq r$, i.e., $\lfloor \frac{m_i}{2} \rfloor \bmod \rho_i = 0$, then the total distance $f(m,\rho)$ is precisely defined by Lemma 2.8 (See Equations (B.12) and (B.14) in Section B.3 of Appendix B). For the perturbation analysis of the balanced hypercycle, we shall first analyze Eq. (2.22) when $\lfloor \frac{m}{2} \rfloor \bmod \rho = 0$. Consider the following cases of m.

1. m **- Odd**

When m is odd, the factors of m^2, namely $(m-\lambda)$ and $(m+\upsilon)$ are also odd. Hence, substituting Eq. (B.12) for $f(m-\lambda,\rho)$ and $f(m+\upsilon,\rho)$ in Eq. (2.22), we obtain

$$\begin{aligned} (M-1)(\delta' - \delta_b) &\geq m^{r-2}\left[(m+\upsilon)\left[\frac{(m-\lambda-1)}{2} + \frac{(m-\lambda-1)^2}{4\rho}\right]\right. \\ &\qquad \left. + (m-\lambda)\left[\frac{(m+\upsilon-1)}{2} + \frac{(m+\upsilon-1)^2}{4\rho}\right] - 2mf(m,\rho)\right] \\ &= m^{r-2}\left(\frac{(m^2-m-\upsilon)}{2} + \frac{(m-\lambda-1)(m^2-m-\upsilon)}{4\rho}\right. \\ &\qquad \left. + \frac{(m^2-m+\lambda)}{2} + \frac{(m+\upsilon-1)(m^2-m+\lambda)}{4\rho} - 2mf(m,\rho)\right) \\ &= m^{r-2}\left[m^2 - m + \left(\frac{(m-1)^2}{4\rho} - \frac{1}{2}\right)(\upsilon-\lambda) + \frac{2\upsilon\lambda}{4\rho} + \frac{m^3}{2\rho} - \frac{m^2}{\rho}\right. \\ &\qquad \left. + \frac{m}{2\rho} - 2mf(m,\rho)\right] \end{aligned}$$

Similarly, substituting Eq. (2.15) for $f(m,\rho)$ in the above, we obtain

$$\begin{aligned}(M-1)(\delta'-\delta_b) &\geq m^{r-2}\left[m^2-m+\left(\frac{(m-1)^2}{4\rho}-\frac{1}{2}\right)(\upsilon-\lambda)+\frac{2\upsilon\lambda}{4\rho}+\frac{m^3}{2\rho}-\frac{m^2}{\rho}\right.\\ &\quad\left.+\frac{m}{2\rho}-2m\left(\frac{m-1}{2}+\frac{(m-1)^2}{4\rho}\right)\right]\\ &= m^{r-2}\left[m^2-m+\left(\frac{(m-1)^2}{4\rho}-\frac{1}{2}\right)(\upsilon-\lambda)+\frac{2\upsilon\lambda}{4\rho}+\frac{m^3}{2\rho}-\frac{m^2}{\rho}\right.\\ &\quad\left.+\frac{m}{2\rho}-m(m-1)-\frac{m(m-1)^2}{2\rho}\right]\\ &= m^{r-2}\left[m^2-m+\left(\frac{(m-1)^2}{4\rho}-\frac{1}{2}\right)(\upsilon-\lambda)+\frac{2\upsilon\lambda}{4\rho}+\frac{m^3}{2\rho}-\frac{m^2}{\rho}\right.\\ &\quad\left.+\frac{m}{2\rho}-m^2+m-\frac{m^3}{2\rho}+\frac{m^2}{\rho}-\frac{m}{2\rho}\right]\\ &= m^{r-2}\left[\left(\frac{(m-1)^2}{4\rho}-\frac{1}{2}\right)(\upsilon-\lambda)+\frac{2\upsilon\lambda}{4\rho}\right]\geq 0 \qquad (2.23)\end{aligned}$$

since $(\upsilon-\lambda)=\frac{\upsilon\lambda}{m}=\frac{\upsilon^2}{m+\upsilon}\geq 0$ and $\frac{(m-1)^2}{4\rho}-\frac{1}{2}\geq\frac{m-2}{2}>0$ for $m\geq 3$ and $1\leq\rho\leq\frac{m-1}{2}$.

2. m **- Even**

When m is even, the factors of m^2, namely, $m-\lambda$ and $m+\upsilon$ can assume 3 possible combinations depending on whether λ and υ are even or odd.

(a) υ and λ - Even

In this case, $m-\lambda$ and $m+\upsilon$ are even. Substituting Eq. (B.14) for $f(m-\lambda,\rho)$ and $f(m+\upsilon,\rho)$ in Eq. (2.22), we obtain

$$\begin{aligned}(M-1)(\delta'-\delta_b) &\geq m^{r-2}\left[(m+\upsilon)\left(\frac{(m-\lambda)}{2}+\frac{(m-\lambda)(m-\lambda-2)}{4\rho}\right)\right.\\ &\quad+(m-\lambda)\left(\frac{(m+\upsilon)}{2}+\frac{(m+\upsilon)(m+\upsilon-2)}{4\rho}\right)\\ &\quad\left.-2mf(m,\rho)\right]\\ &= m^{r-2}\left[\frac{m^2}{2}+\frac{m^2(m-\lambda-2)}{4\rho}+\frac{m^2}{2}+\frac{m^2(m+\upsilon-2)}{4\rho}\right.\\ &\quad\left.-2mf(m,\rho)\right]\\ &= m^{r-2}\left[m^2+\frac{m^3}{2\rho}-\frac{m^2}{\rho}+\frac{m^2}{4\rho}(\upsilon-\lambda)-2mf(m,\rho)\right]\end{aligned}$$

Substituting for $f(m,\rho)$ (m being even) from Eq. (2.15) in the above, we obtain

$$\begin{aligned}\delta'-\delta_b &\geq m^{r-2}\left[m^2+\frac{m^3}{2\rho}-\frac{m^2}{\rho}+\frac{m^2(\upsilon-\lambda)}{4\rho}-2m\left(\frac{m}{2}+\frac{m(m-2)}{4\rho}\right)\right]\\ &\geq \left(\frac{m^r}{4\rho}\right)(\upsilon-\lambda)\geq 0 \qquad \text{since } \upsilon>\lambda \qquad (2.24)\end{aligned}$$

(b) υ - odd and λ - even

In this case, substituting Eqs. (B.12) and (B.14) for $(m+\upsilon,\rho)$ and $f(m-\lambda,\rho)$ respectively, we obtain

$$\begin{aligned}(M-1)(\delta'-\delta_b) &\geq m^{r-2}\left[(m+\upsilon)\left[\frac{(m-\lambda)}{2}+\frac{(m-\lambda)(m-\lambda-2)}{4\rho}\right]\right.\\ &\quad\left.+(m-\lambda)\left[\frac{(m+\upsilon-1)}{2}+\frac{(m+\upsilon-1)^2}{4\rho}\right]-2mf(m,\rho)\right]\\ &= m^{r-2}\left[\frac{m^2}{2}+\frac{m^2(m-\lambda-2)}{4\rho}+\frac{(m^2-m+\lambda)}{2}\right.\\ &\quad\left.+\frac{(m+\upsilon-1)(m^2-m+\lambda)}{4\rho}-2mf(m,\rho)\right]\\ &= m^{r-2}\left[m^2+\frac{m^3}{2\rho}+\frac{\upsilon\lambda}{4\rho}+\frac{m-\lambda}{4\rho}+\frac{\lambda}{2}+\frac{(\upsilon-\lambda)m(m-1)}{4\rho}\right.\\ &\quad\left.-\frac{m}{2}-\frac{m^2}{\rho}-2mf(m,\rho)\right]\end{aligned}$$

Substituting $f(m,\rho)$ given by Eq. (B.14) in the above, we obtain

$$\begin{aligned}(M-1)(\delta'-\delta_b) &\geq m^{r-2}\left[m^2+\frac{m^3}{2\rho}+\frac{\upsilon\lambda}{4\rho}+\frac{m-\lambda}{4\rho}+\frac{\lambda}{2}+\frac{(\upsilon-\lambda)m(m-1)}{4\rho}\right.\\ &\quad\left.-\frac{m}{2}-\frac{m^2}{\rho}-2m\left(\frac{m}{2}+\frac{m(m-2)}{4\rho}\right)\right]\\ &= m^{r-2}\left[\frac{(\upsilon-\lambda)m(m-1)}{4\rho}+\frac{\upsilon\lambda}{4\rho}+\frac{m-\lambda}{4\rho}+\frac{\lambda}{2}-\frac{m}{2}\right]\\ &= m^{r-2}\left[\frac{m^2(\upsilon-\lambda)}{4\rho}+\frac{m-\lambda}{4\rho}+\frac{\lambda}{2}-\frac{m}{2}\right]\geq 0\quad\forall m,\rho \qquad (2.25)\end{aligned}$$

Observe that $\frac{m^2(\upsilon-\lambda)}{4\rho}-\frac{m}{2}=\frac{m}{2}\left(\frac{m(\upsilon-\lambda)}{2\rho}-1\right)$. Since $\upsilon-\lambda\geq 1$ and $m\geq 2\rho$, it implies that $m(\upsilon-\lambda)\geq 2\rho$ and $\frac{m(\upsilon-\lambda)}{2\rho}-1\geq 0$.

(c) υ - even and λ - odd

In a similar manner as above, substituting Eqs. (B.12) and (B.14) for $f(m-\lambda,\rho)$ and $(m+\upsilon,\rho)$ respectively, we obtain

$$\begin{aligned}(M-1)(\delta'-\delta_b) &\geq m^{r-2}\left[(m+\upsilon)\left(\frac{(m-\lambda-1)}{2}+\frac{(m-\lambda-1)^2}{4\rho}\right)+\right.\\ &\quad\left.(m-\lambda)\left(\frac{(m+\upsilon)}{2}+\frac{(m+\upsilon)(m+\upsilon-2)}{4\rho}\right)-2mf(m,\rho)\right]\\ &= m^{r-2}\left[\frac{(m^2-m-\upsilon)}{2}+\frac{(m-\lambda-1)(m^2-m-\upsilon)}{4\rho}+\frac{m^2}{2}\right.\\ &\quad\left.+\frac{m^2(m+\upsilon-2)}{4\rho}-2mf(m,\rho)\right]\end{aligned}$$

$$= m^{r-2}\left[m^2 + \frac{m^3}{2\rho} + \frac{(\upsilon-\lambda)m(m-1)}{4\rho} - \frac{m^2}{\rho} + \frac{m+\upsilon}{\rho} - \frac{m+\upsilon}{2} + \frac{\upsilon\lambda}{4\rho} - 2mf(m,\rho)\right]$$

Substituting $f(m,\rho)$ given by Eq. (B.14) in the above, we obtain

$$\begin{aligned}(M-1)(\delta'-\delta_b) &\geq m^{r-2}\left[m^2 + \frac{m^3}{2\rho} + \frac{(\upsilon-\lambda)m(m-1)}{4\rho} - \frac{m^2}{\rho} + \frac{m+\upsilon}{\rho} - \frac{m+\upsilon}{2} + \frac{\upsilon\lambda}{4\rho} - 2m\left(\frac{m}{2} + \frac{m(m-2)}{4\rho}\right)\right] \\ &= m^{r-2}\left[\frac{(\upsilon-\lambda)m(m-1)}{4\rho} + \frac{m+\upsilon}{\rho} - \frac{m+\upsilon}{2} + \frac{\upsilon\lambda}{4\rho}\right] \\ &= m^{r-2}\left[\frac{m^2(\upsilon-\lambda)}{4\rho} + \frac{m+\upsilon}{\rho} - \frac{m+\upsilon}{2}\right] \qquad (2.26)\end{aligned}$$

since $\upsilon - \lambda = \frac{\upsilon\lambda}{m}$.

To prove $\delta' - \delta_b \geq 0$ as given Eq. (2.26) for $1 \leq \rho \leq \frac{m-1}{2}$, we consider the following cases:

1. $1 \leq \rho \leq \frac{m-\lambda-1}{2}$

   Substituting the upper bound of ρ, namely, $\rho = \frac{m-\lambda-1}{2}$ in Eq. (2.26), we obtain

   $$\begin{aligned}(M-1)(\delta'-\delta_b) &\geq m^{r-2}\left[\frac{m^2(\upsilon-\lambda)}{2(m-\lambda-1)} + \frac{2(m+\upsilon)}{m-\lambda-1} - \frac{m+\upsilon}{2}\right] \\ &= m^{r-2}\left[\frac{m^2(\upsilon-\lambda) + (m+\upsilon) - m^2 + (m+\upsilon)}{2(m-\lambda-1)}\right] \\ &= m^{r-2}\left[\frac{m^2(\upsilon-\lambda-1) + 2(m+\upsilon)}{2(m-\lambda-1)}\right] \geq 0\end{aligned}$$

   since $\upsilon - \lambda \geq 1$ when $\upsilon, \lambda \neq 0$.

2. $\frac{m-\lambda+1}{2} \leq \rho \leq \frac{m}{2}$

   Since $\rho \geq \frac{m-\lambda+1}{2} \geq \lfloor \frac{m-\lambda}{2} \rfloor$, it implies that $f(m-\lambda,\rho) = m-\lambda-1$ since it is fully connected. Hence

   $$\begin{aligned}(M-1)(\delta'-\delta_b) &\geq m^{r-2}\left[(m+\upsilon)(m-\lambda-1) + (m-\lambda)\frac{(m+\upsilon-1)}{2} + (m-\lambda)\frac{(m+\upsilon-1)^2}{4\rho} - 2mf(m,\rho)\right] \\ &= m^{r-2}\left[m^2 - m - \upsilon + \frac{m^2}{2} + \frac{m^2(m+\upsilon-2)}{4\rho} - 2mf(m,\rho)\right] \\ &= m^{r-2}\left[m^2 - m - \upsilon + \frac{m^2}{2} + \frac{m^2(m+\upsilon-2)}{4\rho} - 2m\left(\frac{m}{2} + \frac{m(m-2)}{4\rho}\right)\right]\end{aligned}$$

$$= m^{r-2}\left[m^2\left(\frac{\upsilon+2}{4\rho}\right) - (m+\upsilon) + \frac{m^2}{2} - \frac{m^3}{4\rho}\right]$$

Substituting the lower and upper bounds of ρ in the above equation, we obtain,

$$\begin{aligned}(M-1)(\delta'-\delta_b) &\geq m^{r-2}\left[m^2\left(\frac{\upsilon+2}{2m}\right) - (m+\upsilon) + \frac{m^2}{2} - \frac{m^3}{2(m-\lambda+1)}\right] \\ &= m^{r-2}\left[\frac{m(\upsilon+2)}{2} - (m+\upsilon) + \frac{m^2}{2}\left(1 - \frac{1}{m-\lambda+1}\right)\right] \\ &= m^{r-2}\left[\upsilon\left(\frac{m}{2}-1\right) + \frac{m^2}{2}\left(1 - \frac{1}{m-\lambda+1}\right)\right] \geq 0\end{aligned}$$

since $m \geq 2$.

■

Theorem 2.2 *The balanced hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ has a cost optimal diameter when $m_i = \sqrt[r]{M}$ and $\rho_i = \frac{C}{r}$ such that $\lfloor \frac{m_i}{2} \rfloor \bmod \rho_i = 0$, $\forall i \in \{1, 2, \ldots, r\}$ where C is a given constant, and assuming that $\sqrt[r]{M}$ and $\frac{C}{r}$ are integers.*

Proof. For the balanced hypercycle, the diameter k_b is given by

$$k_b = r\left\lceil \frac{\lfloor \frac{m}{2} \rfloor}{\rho} \right\rceil \tag{2.27}$$

where $m = \sqrt[r]{M}$ and $\rho = \frac{C}{r}$.

In an analysis similar to the average distance, we consider a 2-dimensional perturbation from the balanced point m when any two dimension bases change to $(m+\upsilon)$ and $(m-\lambda)$ such that $m^2 = (m+\upsilon)(m-\lambda)$. Then, the new diameter k' is given by

$$\begin{aligned}k' &= (r-2)\left\lceil \frac{\lfloor \frac{m}{2} \rfloor}{\rho} \right\rceil + \left\lceil \frac{\lfloor \frac{m+\upsilon}{2} \rfloor}{\rho} \right\rceil + \left\lceil \frac{\left\lfloor \frac{m-\lambda}{2} \right\rfloor}{\rho} \right\rceil \\ k'-k_b &= \left\lceil \frac{\lfloor \frac{m+\upsilon}{2} \rfloor}{\rho} \right\rceil + \left\lceil \frac{\left\lfloor \frac{m-\lambda}{2} \right\rfloor}{\rho} \right\rceil - 2\left\lceil \frac{\lfloor \frac{m}{2} \rfloor}{\rho} \right\rceil \end{aligned} \tag{2.28}$$

Since $\lfloor \frac{m}{2} \rfloor \bmod \rho = 0$, we have

$$\begin{aligned}k'-k_b &= \left\lceil \frac{\lfloor \frac{m+\upsilon}{2} \rfloor}{\rho} \right\rceil + \left\lceil \frac{\left\lfloor \frac{m-\lambda}{2} \right\rfloor}{\rho} \right\rceil - \frac{2\lfloor \frac{m}{2} \rfloor}{\rho} \\ &\geq \frac{\lfloor \frac{m+\upsilon}{2} \rfloor}{\rho} + \frac{\left\lfloor \frac{m-\lambda}{2} \right\rfloor}{\rho} - \frac{2\lfloor \frac{m}{2} \rfloor}{\rho}\end{aligned} \tag{2.29}$$

Let us analyze Eq. (2.29) by considering the following cases.

Case 1: m **- even**

1. Both $m+\upsilon$ and $m-\lambda$ are even

$$\begin{aligned} k'-k_b &\geq \frac{1}{\rho}\left(\frac{m+\upsilon}{2}+\frac{m-\lambda}{2}-m\right) \\ &\geq \frac{\upsilon-\lambda}{2\rho} \geq 0 \end{aligned}$$

2. $m+\upsilon$ is even, $m-\lambda$ is odd and vice versa

$$\begin{aligned} k'-k_b &\geq \frac{1}{\rho}\left(\frac{m+\upsilon}{2}+\frac{m-\lambda}{2}-\frac{1}{2}-m\right) \\ &\geq \frac{\upsilon-\lambda-1}{2\rho} \geq 0 \quad \text{(from Eq. (2.20))} \end{aligned}$$

Case 2: m **- odd**

It implies that both $m+\upsilon$ and $m-\lambda$ are odd.

$$\begin{aligned} k'-k_b &\geq \frac{1}{\rho}\left(\frac{m+\upsilon-1}{2}+\frac{m-\lambda-1}{2}-(m-1)\right) \\ &= \frac{\upsilon-\lambda}{2\rho} > 0 \end{aligned}$$

■

When $\lfloor \frac{m}{2} \rfloor \bmod \rho \neq 0$, the perturbation analysis is exceedingly complex and intractable. However, it is possible that 2-dimensional perturbations (from the balanced case) may lead to graphs that have a smaller average distance and diameter as compared to a balanced hypercycle. This point will be elaborated by the following theorem.

Theorem 2.3 *When the number of nodes M, dimension r and connectivity-sum C are given constants, balanced hypercycles do not necessarily exhibit minimum average distance (diameter) when $m_i = m = \sqrt[r]{M}$, $\rho_i = \rho = \frac{C}{r}$ and $\lfloor \frac{m}{2} \rfloor \bmod \rho \neq 0$ such that $\sqrt[r]{M}$ and $\frac{C}{r}$ are integers.*

Proof. This theorem can be proved through a counter example. Let $M = 144$, $r = 2$ and $C = 8$. Applying the result of Theorem 2.1, we obtain the balanced hypercycle $\mathcal{G}_{12}^{4\ 4}{}_{12}$ whose average distance is 2.349650. Observe that in this case $\lfloor \frac{m}{2} \rfloor \bmod \rho \neq 0$ while the hypercycle $\mathcal{G}_{9\ 16}^{4\ 4}$ has an average distance of 2.279720. Similarly the diameter of $\mathcal{G}_{12\ 12}^{4\ 4}$ is 4 while that of $\mathcal{G}_{9\ 16}^{4\ 4}$ is 3. ■

In fact one can find families of hypercycles that present smaller average distance (diameter) compared to the balanced hypercycle when $\lfloor \frac{m}{2} \rfloor \bmod \rho \neq 0$ and this will be discussed shortly in Theorem 2.4. Before proceeding further, we present Lemmas 2.9 to 2.15 whose proofs are given in Appendix C.

Lemma 2.9 *If* $m = \rho(\rho-1)$, *then* $f(m,\rho) = \frac{\rho}{4}\left(\rho^2-2\right)$ *where* ρ *is even.*

Lemma 2.10 *If* $m+\upsilon = \rho^2$, *then* $f(m+\upsilon,\rho) = \frac{\rho}{4}\left(\rho^2+2\rho-2\right)$ *where* υ *is a positive integer and* ρ *being even.*

Lemma 2.11 *If* $m-\lambda = (\rho-1)^2$, *then* $f(m-\lambda,\rho) = \frac{\rho^2}{4}\left(\rho-2\right)$ *where* λ *is a positive integer and* ρ *being even.*

Lemma 2.12 *If* $m = (\rho-1)(\rho-2)$, *then* $f(m,\rho) = \frac{(\rho-2)\left(\rho^2-2\rho+2\right)}{4}$ *where* ρ *is even.*

Lemma 2.13 *If* $m-\lambda = (\rho-2)^2$, *then* $f(m-\lambda,\rho) = \frac{(\rho-2)(\rho^2-4\rho+6)}{4}$ *where* λ *is a positive integer and* ρ *being even.*

Lemma 2.14 *If* $m = \rho(\rho+1)$, *then* $f(m,\rho) = \frac{(\rho+2)\left(\rho^2+2\rho-2\right)}{4}$ *where* ρ *is even.*

Lemma 2.15 *If* $m+\upsilon = (\rho+1)^2$, *then* $f(m+\upsilon,\rho) = \frac{\rho(\rho+2)(\rho+4)}{4}$ *where* υ *is a positive integer and* ρ *being even.*

Theorem 2.4 *The following families of non-balanced 2-dimensional hypercycles* $\mathcal{G}^{\rho\ \rho}_{m_1\ m_2}$ *have smaller average distance compared to the balanced hypercycle* $\mathcal{G}^{\rho\ \rho}_{m\ m}$ *when* $\lfloor \frac{m}{2} \rfloor \bmod \rho \neq 0$.

Family 1*: ρ is even and $\rho \geq 4$*

$$\begin{aligned} m &= \rho(\rho-1) \\ m_1 &= (\rho-1)^2 \\ m_2 &= \rho^2 \end{aligned}$$

Family 2*: ρ is even and $\rho \geq 6$*

$$\begin{aligned} m &= (\rho-2)(\rho-1) \\ m_1 &= (\rho-2)^2 \\ m_2 &= (\rho-1)^2 \end{aligned}$$

Family 3*: ρ is even and $\rho \geq 4$*

$$\begin{aligned} m &= \rho(\rho+1) \\ m_1 &= \rho^2 \\ m_2 &= (\rho+1)^2 \end{aligned}$$

Proof.

Family 1: Observe that $M = m^2 = m_1 m_2 = (m-\lambda)(m+\upsilon)$ and hence $m-\lambda = (\rho-1)^2$ and $m+\upsilon = \rho^2$. The connectivity constraint is satisfied for $m-\lambda$ since $2\rho < \rho^2 - 2\rho + 1$ which implies that $\rho^2 - 4\rho + 1 > 0$ for $\rho \geq 4$. Substituting $r = 2$ and Eqs (C.1), (C.2) and (C.3) in Eq. (2.22), the difference $\delta' - \delta_b$ between the non-balanced and balanced hypercycles can be computed as

$$\begin{aligned}(M-1)(\delta'-\delta_b) &= \rho^2\left(\frac{\rho^2(\rho-2)}{4}\right) + (\rho-1)^2\left(\frac{\rho}{4}\left(\rho^2+2\rho-2\right)\right) \\ &\quad -2\rho(\rho-1)\left(\frac{\rho^2(\rho-2)}{4}\right) \\ &= \frac{\rho}{4}\left[-\rho^2+2\rho-2\right] < 0 \quad \text{for } \rho \geq 4 \end{aligned} \tag{2.30}$$

since the maximum value of $(-\rho^2 + 2\rho - 2)$ is -1 for $\rho = 1$.

Family 2: In this case, $m+\upsilon = (\rho-1)^2$ and $m-\lambda = (\rho-2)^2$. To check the connectivity constraint of $m-\lambda$, we have $2\rho \leq \rho^2 - 4\rho + 4$ which implies that $\rho^2 - 6\rho + 4 \geq 0$ for $\rho \geq 6$. Note that $f(m+\upsilon, \rho)$ is precisely defined by Eq. (C.3).

Substituting Eqs. (C.3), (C.4) and (C.5) in Eq. (2.22), we obtain

$$\begin{aligned}(M-1)(\delta'-\delta_b) &= (\rho-1)^2\left(\frac{(\rho-2)(\rho^2-4\rho+6)}{4}\right) + (\rho-2)^2\left(\frac{\rho^2(\rho-2)}{4}\right) \\ &\quad -2(\rho-1)(\rho-2)^2\left(\frac{\rho^2-2\rho+2}{4}\right) \\ &= \frac{\rho-2}{4}\Big[(\rho^2-2\rho+1)(\rho^2-4\rho+6) + \rho^2(\rho^2-4\rho+4) \\ &\quad -2(\rho^2-3\rho+2)(\rho^2-2\rho+2)\Big] \\ &= \frac{\rho-2}{4}\left[-\rho^2+4\rho-2\right] < 0 \quad \text{for } \rho \geq 6 \end{aligned} \tag{2.31}$$

Family 3: We have $m+\upsilon = (\rho+1)^2$ and $m-\lambda = \rho^2$. Therefore, $f(m-\lambda)$ is given by Eq. (C.2). Substituting Eqs. (C.2), (C.6) and (C.7) in Eq. (2.22), we have

$$\begin{aligned}(M-1)(\delta'-\delta_b) &= (\rho+1)^2\left(\frac{\rho(\rho^2+2\rho-2)}{4}\right) + \rho^2\left(\frac{\rho(\rho+2)(\rho+4)}{4}\right) \\ &\quad -2\rho(\rho+1)(\rho+2)\left(\frac{\rho^2+2\rho-2}{4}\right) \\ &= \frac{\rho}{4}\Big[(\rho^2+2\rho+1)(\rho^2+2\rho-2) + \rho^2(\rho^2+6\rho+8) \\ &\quad -2(\rho^2+3\rho+2)(\rho^2+2\rho-2)\Big] \\ &= \frac{\rho}{4}\left[-\rho^2+2\rho+6\right] < 0 \quad \text{for } \rho \geq 4 \end{aligned} \tag{2.32}$$

■

Theorem 2.5 *Families 1 and 3 as defined in Theorem 2.4 have smaller diameter than the 2-dimensional balanced hypercycle* $\mathcal{G}^{\rho\,\rho}_{m\,m}$.

Proof.

Family 1: For $m = \rho(\rho-1)$, the diameter is given by

$$\left\lceil \frac{\left\lfloor \frac{m}{2} \right\rfloor}{\rho} \right\rceil = \left\lceil \frac{\left\lfloor \frac{\rho(\rho-1)}{2} \right\rfloor}{\rho} \right\rceil = \left\lceil \frac{\rho(\rho-1)}{2\rho} \right\rceil = \frac{\rho}{2} \tag{2.33}$$

For $m + v = \rho^2$, we have

$$\left\lceil \frac{\left\lfloor \frac{m+v}{2} \right\rfloor}{\rho} \right\rceil = \left\lceil \frac{\rho^2}{2\rho} \right\rceil = \frac{\rho}{2} \tag{2.34}$$

Similarly when $m - \lambda = (\rho-1)^2$

$$\left\lceil \frac{\left\lfloor \frac{m-\lambda}{2} \right\rfloor}{\rho} \right\rceil = \left\lceil \frac{\left\lfloor \frac{(\rho-1)^2}{2} \right\rfloor}{\rho} \right\rceil = \left\lceil \frac{(\rho-1)^2-1}{2\rho} \right\rceil = \frac{\rho-2}{2} \tag{2.35}$$

The difference in diameter between the non-balanced hypercycle and the balanced hypercycle can be obtained by substituting Eqs. (2.33), (2.34) and (2.35) in Eq. (2.28) as follows:

$$k' - k_b = \frac{\rho}{2} + \frac{\rho-2}{2} - 2\left(\frac{\rho}{2}\right) = -1 \tag{2.36}$$

Family 3: Substituting $m = \rho(\rho+1)$, $m + v = (\rho+1)^2$ and $m - \lambda = \rho^2$ in Eq. (2.27), we obtain

$$\begin{aligned} k' - k_b &= \left\lceil \frac{\left\lfloor \frac{(\rho+1)^2}{2} \right\rfloor}{\rho} \right\rceil + \left\lceil \frac{\left\lfloor \frac{\rho^2}{2} \right\rfloor}{\rho} \right\rceil - 2\left\lceil \frac{\left\lfloor \frac{\rho(\rho+1)}{2} \right\rfloor}{\rho} \right\rceil \\ &= \left\lceil \frac{(\rho+1)^2-1}{2\rho} \right\rceil + \left\lceil \frac{\rho^2}{2\rho} \right\rceil - 2\left\lceil \frac{\rho(\rho+1)}{2\rho} \right\rceil \\ &= \frac{\rho+2}{2} + \frac{\rho}{2} - 2\left(\frac{\rho+2}{2}\right) = -1 \end{aligned} \tag{2.37}$$

■

Remark: If the difference $k'-k_b$ is computed for Family 2 by substituting $m = (\rho-1)(\rho-2)$, $m - \lambda = (\rho-2)^2$ and $m + v = (\rho-1)^2$ in Eq. (2.27), then

$$\begin{aligned} k' - k_b &= \left\lceil \frac{\left\lfloor \frac{(\rho-1)^2}{2} \right\rfloor}{\rho} \right\rceil + \left\lceil \frac{\left\lfloor \frac{(\rho-2)^2}{2} \right\rfloor}{\rho} \right\rceil - 2\left\lceil \frac{\left\lfloor \frac{(\rho-1)(\rho-2)}{2} \right\rfloor}{\rho} \right\rceil \\ &= \left\lceil \frac{(\rho-1)^2-1}{2\rho} \right\rceil + \left\lceil \frac{(\rho-2)^2}{2\rho} \right\rceil - 2\left\lceil \frac{(\rho-1)(\rho-2)}{2\rho} \right\rceil \\ &= \frac{\rho-2}{2} + \frac{\rho-2}{2} - 2\left(\frac{\rho-2}{2}\right) \\ &= 0 \end{aligned} \tag{2.38}$$

Table 2.1 *Average Distance Minimization for 3-D hypercycle - Case 1*

Input		Theoretical Optimum						Numerical Optimum						
Nodes	C	Node Radices*			Connectivity*			Node Radices*			Connectivity*			Average
M		m_1	m_2	m_3	ρ_1	ρ_2	ρ_3	m_1	m_2	m_3	ρ_1	ρ_2	ρ_3	Distance
128	5	5.09	5.09	5.09	1.67	1.67	1.67	4	4	8	1	1	2	3.0236
625	6	8.55	8.55	8.55	2	2	2	25	5	5	4	1	1	4.3269
1000	10	10	10	10	3.33	3.33	3.33	10	10	10	3	3	4	3.4034
2744	9	14	14	14	3	3	3	14	14	14	3	3	3	4.5016
27000	30	30	30	30	10	10	10	30	30	30	10	10	10	3.8001
32768	21	32	32	32	7	7	7	32	32	32	7	7	7	4.7813

Table 2.2 *Violations in average distance minimization for 3-D hypercycles - Case 1*

Input		Theoretical Optimum							Numerical Optimum						
Nodes	C	Node Radices*			Connectivity*			Average	Node Radices*			Connectivity*			Average
M		m_1	m_2	m_3	ρ_1	ρ_2	ρ_3	Distance	m_1	m_2	m_3	ρ_1	ρ_2	ρ_3	Distance
1728	12	12	12	12	4	4	4	3.5020	9	12	16	4	4	4	3.4325
8000	21	20	20	20	7	7	7	3.6005	16	20	25	7	7	7	3.5605
27000	33	30	30	30	11	11	11	3.6001	25	24	45	11	11	11	3.5069
46656	42	36	36	36	14	14	14	3.5000	27	32	54	14	14	14	3.4699
343000	69	70	70	70	23	23	23	3.9429	50	70	98	23	23	23	3.9155

Hence the non-balanced hypercycle for Family 2 has the same diameter as the balanced hypercycle. The exceptions stipulated by Theorem 2.3 shall be further discussed in Section 2.3.3.

2.3.3 Numerical Experiments

In the previous section, the optimal hypercycle network design for minimum average distance has been formulated as a subclass of the more general mixed discrete constrained nonlinear programming (MDCNP) problem. To conduct perturbation analysis on the balanced point, a C program was written that exhaustively enumerated the various combinations in the search space Ω for computationally feasible values of r and M. The results of discrete optimization for the various cases are presented in Tables 2.1 to 2.8 corresponding to 3-dimensional hypercycles. The optimal point calculated theoretically is listed along with the obtained numerical value.

Remarks:

- The numerical solution, given in Table 1, agrees with the analytical solution for all cases $\lfloor \frac{m_i}{2} \rfloor \bmod \rho_i = 0$, $i = 1, \ldots, r$.

Table 2.3 *Violations in diameter for 3-D hypercycles - Case 1*

Input		Theoretical Optimum							Numerical Optimum						
Nodes	C	Node Radices*			Connectivity*			Diameter	Node Radices*			Connectivity*			Diameter
M		m_1	m_2	m_3	ρ_1	ρ_2	ρ_3	k	m_1	m_2	m_3	ρ_1	ρ_2	ρ_3	k
216	6	6	6	6	2	2	2	6	4	6	9	2	2	2	5
1728	12	12	12	12	4	4	4	6	8	9	24	4	4	4	5
8000	9	20	20	20	3	3	3	12	16	20	25	3	3	3	11
13824	27	24	24	24	9	9	9	6	18	24	32	9	9	9	5
27000	42	30	30	30	14	14	14	6	20	25	54	14	14	14	4
373248	24	72	72	72	8	8	8	15	64	72	81	8	8	8	14

- When the analytically computed points $\vec{m}$ and $\vec{\rho}$ are not integers, then a discrete optimization has to be carried in the neighbourhood of the theoretical optimum to determine the true integer values. The theoretical minimum could therefore provide a convenient starting point for the optimization.
- When $\lfloor \frac{m_i}{2} \rfloor \bmod \rho_i \neq 0$, $i = 1, \ldots, r$ and the theoretical minimum of $\vec{m}$ and $\vec{\rho}$ are integers, there exist hypercycles which are not balanced and which have smaller average distance and diameter as compared to those of the balanced hypercycle given by Theorems 2.1 and 2.2. We shall call these cases as *violation cases*. Some typical counter examples for average distance and diameter are listed in Tables 2.2 and 2.3. In order to quantify these violations, an exhaustive search for graphs from a given balanced point was conducted with r-dimensional perturbations where $r \in \{2, 3, 4, 5\}$. This search covered all valid graphs upto $k = 3.0518\text{E}10$, $d = 1245$ and $\delta = 1.5259\text{E}10$. The balanced point was chosen such that $M = m^r$, $1 \leq \rho \leq \lfloor \frac{m}{2} \rfloor$ where $m \in \{2, 3, \ldots, 250\}$ for each r as stated above. The results of the numerical experiments are summarized in Tables 2.4 and 2.5 where δ' (k') refers to the average distance (diameter) of the violation points respectively. As it can be seen a small percentage of balanced hypercycles are non-optimal and this occurs when $\lfloor \frac{m_i}{2} \rfloor \bmod \rho_i \neq 0$. In addition, the following empirical observations are noteworthy.
  - The percentage of violations decreases as r increases.
  - The neighbourhood of the balanced point which contains the true optimum for δ when $X \neq 0$ is listed in Table 2.6. Each constant ϵ_i, $1 \leq i \leq 5$, represents the maximum relative deviation from a given balanced point $\vec{m}^*$ obtained empirically.
  - A violation for average distance does not necessarily imply a violation in diameter and vice versa.

Table 2.4 *Statistics of violations for Average distance*

Dimension of hypercycle	No. of combinations	No. of violations	Relative No. of violations	Peak relative difference $\frac{\lvert\delta_b-\delta'\rvert}{\delta_b}$
2	114856	1793	1.5611 %	3.3766 %
3	2040895	3532	0.1731 %	3.1111 %
4	53449165	7782	0.0146 %	2.6251 %
5	292607420	10781	0.0037 %	2.1437 %

Table 2.5 *Statistics of violations for Diameter*

Dimension of hypercycle	No. of combinations	No. of violations	Relative No. of violations	Peak relative difference $\frac{\lvert k_b-k'\rvert}{k_b}$
2	114856	3272	2.8488 %	25.00 %
3	2040895	15455	0.7573 %	33.33 %
4	53449165	90977	0.1702 %	37.50 %
5	292607420	299461	0.1023 %	40.50 %

– For the diameter, the number of violations are larger compared to that of the average distance for any given r and this can be attributed to the fact that the bounds of k used in the analytical derivation are less rigid. However, no violations were observed when $\lfloor \frac{m_i}{2} \rfloor \bmod \rho_i = 0$, $i = 1, \ldots, r$.

2.4 Other Optimization cases

In this section, we present some analytical results for the average distance/diameter corresponding to Cases 2 and 3 and these are direct consequences of Lemmas 2.1 and 2.5.

2.4.1 Case 2

Theorem 2.6 *The hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is cost optimal for the upper and lower bounds of the average distance and diameter when $\rho_i = c_i$ is an integer such that*

$$m_i = \sqrt[r]{\frac{Mc_i^{r-1}}{\prod_{j=1, j\neq i}^{r} c_j}}, \qquad i = 1, \ldots, r \tag{2.39}$$

Table 2.6 *Neighbourhood of balanced point for violation cases for Average Distance*

Dimension	Search boundary from balanced point $\vec{m}^*$				
	$\vec{\epsilon} = [\epsilon_1 \quad \epsilon_2 \quad \epsilon_3 \quad \epsilon_4 \quad \epsilon_5]$				
r	ϵ_1	ϵ_2	ϵ_3	ϵ_4	ϵ_5
2	31.58 %	46.15 %			
3	31.58 %	30.00 %	77.78 %		
4	31.58 %	30.00 %	30.00 %	87.50 %	
5	31.25 %	30.00 %	30.00 %	30.00 %	100 %

where the dimension r, the total number of nodes M and the connectivities c_i are given integer constants.

Proof. From Eq. (2.8), we have

$$m_t = m_1 \left(\frac{\rho_t}{\rho_1}\right) = \left(\frac{m_1}{c_1}\right) c_t, \qquad 2 \le t \le r$$

Then

$$\prod_{t=2}^{r} m_t = \left(\frac{m_1}{c_1}\right)^{r-1} \prod_{t=2}^{r} c_t$$

$$\frac{M}{m_1} = \left(\frac{m_1}{c_1}\right)^{r-1} \prod_{t=2}^{r} c_t$$

$$\Rightarrow m_1 = \sqrt[r]{\frac{M c_1^{r-1}}{\prod_{t=2}^{r} c_t}}$$

$$\Rightarrow m_t = \sqrt[r]{\frac{M c_t^{r-1}}{\prod_{j=1, j\neq t}^{r} c_j}}$$

The results obtained from numerical optimization are tabulated in Table 2.7. ■

2.4.2 Case 3

Theorem 2.7 *The hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is cost optimal for the upper and lower bounds of the average distance and diameter when $m_i = c_i$ such that*

$$\rho_i = \frac{C\ \sqrt{c_i}}{\sqrt{c_1} + \sqrt{c_2} + \ldots + \sqrt{c_r}}, \qquad i = 1, \ldots, r \tag{2.40}$$

is an integer where r, C and c_i are given integer constants.

Proof. Rewriting Eq. (2.10), we obtain

$$\rho_t = \rho_1 \sqrt{\frac{m_t}{m_1}} = \rho_1 \sqrt{\frac{c_t}{c_1}}, \qquad 2 \le t \le r$$

Table 2.7 *Average Distance Minimization for a 3-D hypercycle - Case 2*

Input Parameters				Theoretical Optimum			Numerical Optimum			
Total Nodes	Connectivity			Node Radices*			Node Radices*			Average
M	ρ_1	ρ_2	ρ_3	m_1	m_2	m_3	m_1	m_2	m_3	Distance
81	1	2	1	3.43	6.87	3.43	3	9	3	2.7000
100	2	1	2	5.85	2.92	5.85	5	4	5	2.6263
256	4	4	2	8	8	4	8	8	4	2.5098
256	4	2	2	10.08	5.04	5.04	8	4	8	2.8862
512	4	2	1	16	8	4	16	8	4	3.6321
1024	1	1	16	4	4	16	4	4	16	3.4721
1728	3	2	1	19.81	13.21	6.60	18	16	6	5.5866
4000	1	5	3	6.44	32.18	19.31	8	25	20	5.6814
10000	8	5	4	31.75	19.84	15.87	25	20	20	4.3304
16000	4	8	3	22.01	44.03	16.51	25	40	16	5.3328

Table 2.8 *Average Distance Minimization for a 3-D hypercycle - Case 3*

Input Parameters					Theoretical Optimum			Numerical Optimum			
Nodes	Connect.	Node Radices			Connectivity*			Connectivity*			Average
M	Sum	m_1	m_2	m_3	ρ_1	ρ_2	ρ_3	ρ_1	ρ_2	ρ_3	Distance
16	4	2	2	4	1.17	1.17	1.66	1	1	2	1.8667
64	6	4	4	4	2	2	2	2	2	2	2.2857
256	7	8	8	4	2.59	2.59	1.82	3	2	2	3.0117
1000	8	10	20	5	2.56	3.62	1.81	2	4	2	3.9540
1000	7	10	10	10	2.33	2.33	2.33	3	2	2	4.2042
5000	15	25	20	10	5.94	5.31	3.75	6	5	4	3.8408
12000	40	50	20	12	18.85	11.92	9.23	24	10	6	2.8669
17576	14	24	24	24	4.67	4.67	4.66	5	5	4	5.1254
46656	27	36	36	36	9	9	9	9	9	9	4.3334
64000	30	40	40	40	10	10	10	10	10	10	4.3501

Then

$$\begin{aligned}
\sum_{t=2}^{r} \rho_t &= \rho_1 \left(\frac{\sqrt{c_2} + \ldots + \sqrt{c_r}}{\sqrt{c_1}} \right) \\
C - \rho_1 &= \rho_1 \left(\frac{\sqrt{c_2} + \ldots + \sqrt{c_r}}{\sqrt{c_1}} \right) \\
\Rightarrow \rho_1 &= \frac{C\sqrt{c_1}}{\sqrt{c_1} + \sqrt{c_2} + \ldots + \sqrt{c_r}} \\
\Rightarrow \rho_t &= \frac{C\sqrt{c_t}}{\sqrt{c_1} + \sqrt{c_2} + \ldots + \sqrt{c_r}}
\end{aligned} \tag{2.41}$$

Table 2.8 lists the results of numerical optimization for Case 3. ■

2.4.3 Optimum Dimension

The discussion in the previous section intuitively directs us to the next question, whether it is possible to determine the optimal dimension r that will minimize the average distance δ. However, this problem is mathematically intractable owing to the variation in the node radix and connectivity vectors with the dimension r. For specific cases though, it is possible to obtain an analytical solution. For instance, consider the case of a balanced hypercycle with $m_i = m = \sqrt[r]{M}$ and $\rho_i = 1$, $i = 1, \ldots, r$, where $M = p^a$, $m = p^b$ with a, b being integers greater than 0 and p being a given prime number. Then $r = \log_m M = a/b$[‡]. Consider the following cases.

Case 1. $p = 2$, $\rho = 1$

Then, from Eqs. (2.18) and (B.14) we obtain

$$\begin{aligned}
\delta &= \frac{r m^{r-1} f(m, \rho)}{M - 1} \\
&= \frac{rM(m + 2(\rho - 1))}{4\rho(M - 1)} \\
&= \left(\frac{a}{b} \right) \left(\frac{2^a 2^b}{4(2^a - 1)} \right) \\
&= c \left(\frac{2^b}{b} \right)
\end{aligned} \tag{2.42}$$

where $c = \frac{a2^a}{4(2^a - 1)}$ is a constant.

As can be seen, Eq (2.42) is monotonically increasing for $b \geq 2$ since

$$c \left(\frac{2^b}{b} - \frac{2^{b+1}}{b+1} \right) = c \left(\frac{2^b(1-b)}{b(b+1)} \right) < 0, \quad \forall b \geq 2 \tag{2.43}$$

[‡] a/b is an integer because we assumed a balanced hypercycle.

Hence the minimum[§] is $b^* = 1$ and consequently $r = \log_2 M$.

Case 2. $p > 2,\ \rho = 1$

In this case, p is odd and substituting Eq. (B.12) in Eq. (2.18), we have

$$\begin{aligned}\delta &= \frac{rM(m-1)(m-1+2\rho))}{4\rho m(M-1)} \\ &= \left(\frac{a}{b}\right)\left(\frac{p^a(p^{2b}-1)}{4p^b(p^a-1)}\right) \quad \text{since } \rho = 1 \\ &= c\left(\frac{p^{2b}-1}{bp^b}\right) \end{aligned} \tag{2.44}$$

where $c = \frac{ap^a}{4(p^a-1)}$ is a constant. Then

$$\frac{c(p^{2b}-1)}{bp^b} - \frac{c(p^{2(b+1)}-1)}{(b+1)p^{b+1}} = \frac{c\Big((b+1)p(p^{2b}-1) - b(p^{2(b+1)}-1)\Big)}{b(b+1)p^{b+1}} \tag{2.45}$$

Simplifying the numerator, we obtain

$$\begin{aligned} c\Big((b+1)p(p^{2b}-1) - b(p^{2(b+1)}-1)\Big) &= c\Big(p^{2b+1}(b+1-bp) + (b-(b+1)p)\Big) \\ &< 0 \quad \forall b \geq 1, p > 2 \end{aligned}$$

Thus the minimum of Eq (2.44) is achieved when $b^* = 1$ and $r = \log_p M$ gives the optimum dimension.

The above discussions can be summarized as follows:

Lemma 2.16 *When M is an integral power of p where p is prime and $\rho = 1$, a hypercycle $\mathcal{G}^{\vec{\rho}}_{\vec{m}}$ with r dimensions minimizes the average distance such that $r = \log_p M$.*

2.5 Minimization of Degree*Diameter product

A good design of an interconnection network topology should aim at increasing the fault-tolerance, reducing the message routing delay and minimizing the traffic density [41]. The degree and diameter of the network shape the communication bandwidth, network latency, traffic density, the number of alternate paths in the system [25, 42] and the cost of the system. There is a direct trade-off between the degree and diameter since a larger degree implies a smaller diameter and vice versa. Bhuyan and Agarwal [25] therefore introduced the "degree*diameter" product as one of the performance metrics.

The degree*diameter product for a hypercycle is given by

$$\chi = d \times k = \sum_{i=1}^{r} d_i \times \sum_{i=1}^{r} k_i \tag{2.46}$$

[§]In essence, the largest number of dimensions minimizes the average distance and since $\rho = 1$, this corresponds to the largest degree graph.

Due to the increased variability of the node radix and connectivity vectors, the optimization of χ is generally difficult.

As one of the special cases, let us determine the optimum dimension r that will minimize χ corresponding to a balanced hypercycle. Let $M = 2^a$ and $\rho = 2^l$ be given integer constants where $l \geq 0$ and $a \geq 1$. Furthermore, let $m = 2^b$ where b is an integer and $b \geq l + 1$ (since $m \geq 2\rho$). Obviously $\lfloor \frac{m}{2} \rfloor \bmod \rho = 2^{b-1} \bmod 2^l = 0$ and $r = a/b$ should be an integer. To compute the optimum value of χ and hence r, we consider the following cases with the above constraints.

Case 1: $2\rho < m$

Here $b > l+1$. This implies that the hypercycle is not fully connected in each dimension. Then

$$\chi = r \times 2\rho \times r \left\lceil \frac{m}{2\rho} \right\rceil = r^2 m = a^2 \left(\frac{2^b}{b^2} \right) \tag{2.47}$$

To minimize χ, we treat b as the design parameter. By setting the derivative of χ w.r.t. b to zero, we have $\frac{\partial \chi}{\partial b} = \frac{a^2 2^b}{b^3} (b \log 2 - 2) = 0$ which leads to the real solution $b' = 2/\log 2 = 2.8854$. Since b is an integer, the discrete optimum of b can be either 2 or 3. Substituting $b = 2, 3$ in Eq. 2.47 and comparing, we obtain $\frac{4a^2}{4} > \frac{8a^2}{9}$. Hence $b^* = 3$ is the integer minimum and consequently $m^* = 8$. Since $b^* > l + 1$, it implies that $l < b^* - 1$ and hence $l \in \{0, 1\}$. Therefore $\rho = \{1, 2\}$.

Note that the second order derivative of χ is given by

$$\frac{\partial^2 \chi}{\partial b^2} = \frac{a^2 [b 2^b \log 2 \, (b \log 2 - 4) + 6(2^b)]}{b^4}$$

which is positive for all integer values of $b \geq 1$. Thus, when $\rho = 1, 2$, $b^* = 3$ is indeed a minimum of χ and if $r = a/b$ is an integer, it implies that a should be a multiple of 3.

Case 2: $m = 2\rho$

The degree*diameter product can be expressed as

$$\chi = r(2\rho - 1) \times r \left(\frac{m}{2\rho} \right) = r^2 (m - 1) = \frac{a^2 (2^b - 1)}{b^2} \tag{2.48}$$

Setting the first order derivative of Eq. 2.48 to zero, we obtain $\frac{\partial \chi}{\partial b} = \frac{a^2}{b^3} \left(2^b b \log 2 - 2(2^b - 1) \right)$ $= 0$ which has the real solution $b' = 2.2991$. Since b is an integer, the discrete optimum b can be either 2 or 3. Substituting $b = 2, 3$ in Eq. 2.48 and comparing, we obtain $\frac{3a^2}{4} < \frac{7a^2}{9}$. Hence $b^* = 2$ is the integer minimum and consequently $m^* = 4$ and $\rho = 2$. Since $b^* = l + 1$, it implies $l = 1$ and hence $\rho = 2$.

The second order derivative of eqn. 2.48 is given by

$$\frac{\partial^2 \chi}{\partial b^2} = \frac{a^2 [b 2^b \log 2 (b \log 2 - 4) + 6(2^b - 1)]}{b^4} \tag{2.49}$$

which is also positive for all integer values of $b \geq 1$ and therefore $b^* = 2$ is the minimum.

Remark: For $\rho = 2$, we have obtained two optimum values of b^*, namely $b^* = 3$ and $b^* = 2$ corresponding to Cases 1 and 2 respectively. By substituting the respective values in Eqs. (2.47) and (2.48), the global minimum for $\rho = 2$ can be ascertained. Indeed, observe that $\frac{8a^2}{9} > \frac{3a^2}{4}$ and hence $b^* = 2$ is the optimum solution for $\rho = 2$. This implies that a should be a multiple of 2 if $r = a/b$ is an integer.

Discussion: In essence, we have shown for all balanced hypercycles when M, m and ρ are powers of 2,

1. A hypercycle that is not fully connected in each dimension has an optimum degree*diameter when $\rho = 1$, $m = 8$ provided $r = \log_8 M$ is an integer.
2. A hypercycle that is fully connected in each dimension has an optimum degree*diameter when $\rho = 2$, $m = 4$ provided $r = \log_4 M$ is an integer.

On a related note, it is worth mentioning the conjectures [25] raised by Bhuyan and Agrawal in their analysis of the degree*diameter product for the generalized hypercube structures. Note that when $\rho_i = 1$, $1 \leq i \leq r$, the hypercycle $\mathcal{G}^{\vec{\rho}}_{\vec{m}}$ degenerates to a nearest neighbour hypercube while $\rho_i = \lfloor \frac{m_i}{2} \rfloor$ corresponds to a fully connected generalized hypercube structure. Bhuyan and Agrawal's conjectures are summarized below.

Conjecture 2.1 *When M is a given power of 2, a nearest neighbour hypercube has an optimum degree*diameter when $r^* = \lceil \log_8 M \rceil$ dimensions and*

$$[m_1, \ldots, m_{r^*}] = \begin{cases} [8\ 8\ \ldots\ 8\ 8] & \text{if } \log_2 M \bmod 3 = 0 \\ [8\ 8\ \ldots\ 8\ 2] & \text{if } \log_2 M \bmod 3 = 1 \\ [8\ 8\ \ldots\ 8\ 4] & \text{if } \log_2 M \bmod 3 = 2 \end{cases}$$

Conjecture 2.2 *When M is a given power of 2, a generalized hypercube has an optimum degree*diameter when $r^* = \lfloor \log_4 M \rfloor$ dimensions and*

$$[m_1, \ldots, m_{r^*}] = \begin{cases} [4\ 4\ \ldots\ 4\ 4] & \text{if } \log_2 M \text{ is even} \\ [4\ 4\ \ldots\ 4\ 8] & \text{if } \log_2 M \text{ is odd} \end{cases}$$

Chen *et al.* [43] conducted a mathematical study of the above mentioned conjectures and proved that Conjecture 2.1 is true while Conjecture 2.2 is not true when $M = 512$. The above two conjectures are valid only for generalized/nearest neighbour hypercubes. We can extend the above analysis of the degree*diameter product for some special cases of hypercycles (circulants) where the value of ρ is fixed (other than the GHC structures $\rho_i = 1$ or $\rho_i = \lfloor \frac{m_i}{2} \rfloor$, $1 \leq i \leq r$). In the next section, we shall derive some optimal hypercycle structures minimizing the degree*diameter product for $\rho = 2$ assuming that the number of nodes M, and the node radix m_i, are powers of a prime number, p.

2.5.1 Case 1: $p = 2$ and $\vec{\rho} = [2\ 2\ \ldots\ 2\ 2]$

Let $\rho_i = 2$, $1 \leq i \leq r$ for some arbitrary r. i.e. the connectivity in each dimension is fixed as 2 and furthermore let M, m_i be powers of 2.

Theorem 2.8 *When M is an even power of 2 and $\rho_i = 2$, $1 \leq i \leq r$ for some r, a hypercycle $\mathcal{G}^{\vec{\rho}}_{\vec{m}}$ where $\vec{m} = [4\ 4 \ldots 4\ 4]$ and $\vec{\rho} = [2\ 2 \ldots 2\ 2]$ with r^* dimensions has a cost optimal degree*diameter product such that $r^* = \log_4 M$.*

Proof. The approach in this proof is to compute χ for all possible configurations satisfying the above constraints and showing that it is not smaller than the optimal structure. Let the node radices of the optimal hypercycle be represented as $m_t = 4$, $\forall t$ such that $1 \leq t \leq r^*$ and $\rho_t = 2$ and furthermore let $M = 2^a$. Then $r^* = a/2$ and from Eq. (2.48), we have

$$\chi^* = \frac{3a^2}{4} \tag{2.50}$$

The optimality of Eq. (2.50) can be established by comparing χ^* for all valid hypercycle topologies satisfying the constraints $\rho_t = 2$, $m_t \geq 2\rho_t$, $\forall t\ 1 \leq t \leq r$, such that $\chi \geq \chi^*$. i.e, to prove that $\Delta = \chi - \chi^* \geq 0$. Note that r is bounded from above by r^* since otherwise if $r > r^*$, the connectivity constraint will be violated as $\exists m_i$ such that $m_i \leq 2\rho_i$, $1 \leq i \leq r$. However, before proceeding to the actual proof, we state the following Lemmas which are important for subsequent discussion.

Lemma 2.17 *For given integers x and y such that $x, y \geq 3$,*

$$2^x + 2^y - \frac{3}{2}xy \geq \frac{5}{2} \tag{2.51}$$

Proof. Equating the first order derivative of $f = 2^x + 2^y - \frac{3}{2}xy$ to zero, we obtain

$$\begin{aligned} \frac{\partial f}{\partial x} &= 2^x \log 2 - \frac{3y}{2} = 0 \\ \frac{\partial f}{\partial y} &= 2^y \log 2 - \frac{3x}{2} = 0 \end{aligned}$$

which can be numerically solved. The minimum of $f = 2^x + 2^y - \frac{3}{2}xy$ is 2.3406 when $x^* = y^* = 1.9127$ $\forall x, y \geq 0$ and the Hessian matrix is positive definite at the minimum point. The lower bound of Eq. (2.51) can then be obtained by substituting the constraints $x = y = 3$. ■

Lemma 2.18 *For given integers n, b_i where $1 \leq i \leq n$ and $b_i \geq 3$,*

$$(n-1)\sum_{i=1}^{n} 2^{b_i} - \frac{3}{2}\sum_{t=1}^{n-1}\sum_{j=t+1}^{n} b_t b_j \geq \frac{5n(n-1)}{4} \tag{2.52}$$

Proof. We can express Eq. (2.52) as

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(2^{b_i}+2^{b_j}-\frac{3}{2}b_ib_j\right)\geq\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{5}{2}=\frac{5n(n-1)}{4} \tag{2.53}$$

from Lemma 2.17). ■

Lemma 2.19 *For given integers x and y such that $x\geq 2$ and $y\geq 3$,*

$$2^{x+1}+\frac{5}{8}2^y-\frac{3}{2}xy\geq 4 \tag{2.54}$$

Proof. As in Lemma 2.17. ■

Lemma 2.20 *For given integers n, $x\geq 3$ and $y\geq 2$,*

$$2^{x-2}-x \geq -1 \tag{2.55}$$

$$2^x-x^2 \geq -1 \tag{2.56}$$

$$\frac{7}{8}2^x-\frac{3}{4}x^2 \geq \frac{1}{4} \tag{2.57}$$

$$2^x-\frac{3}{4}x^2 \geq \frac{5}{4} \tag{2.58}$$

$$\frac{1}{8}\sum_{i=1}^{n}2^x \geq n \tag{2.59}$$

$$\frac{1}{4}\sum_{i=1}^{n}2^x \geq 2n \tag{2.60}$$

Proof. By elementary mathematical analysis similar to Lemma 2.17. ■

We can now evaluate the cost function Δ for all valid topologies such that $M=\prod_{t=1}^{r}m_t$ where $m_t\geq 4$ and $\rho_t=2$. Let s be an integer such that $0\leq s\leq r$ such that $m_j=4$, $\forall j$ such that $1\leq j\leq s$ and $m_i=2^{b_i}$ where $b_i\geq 3$, $\forall i$ such that $(s+1)\leq i\leq r$. It follows immediately that $a=2s+\sum_{i=s+1}^{r}b_i$. The degree of these hypercycles is given by $d=3s+4(r-s)$ from Eq. (1.5) while the diameter can be calculated from Eq. (1.6) to obtain $k=s+\frac{1}{4}\sum_{i=s+1}^{r}2^{b_i}$. Therefore

$$\begin{aligned}\chi &= (3s+4(r-s))\times\left(s+\frac{1}{4}\sum_{i=s+1}^{r}2^{b_i}\right)\\ &= (4r-s)\left(s+\sum_{i=s+1}^{r}2^{b_i-2}\right)\end{aligned} \tag{2.61}$$

Substituting $a = 2s + \sum_{i=s+1}^{r} b_i$ in Eq. (2.50), we obtain

$$\begin{aligned}\chi^* &= \frac{3}{4}\left(4s^2 + 4s\sum_{i=s+1}^{r} b_i + \left(\sum_{i=s+1}^{r} b_i\right)^2\right) \\ &= 3s^2 + 3s\sum_{i=s+1}^{r} b_i + \frac{3}{4}\left(\sum_{i=s+1}^{r} b_i\right)^2 \end{aligned} \tag{2.62}$$

Subtracting Eq. (2.62) from Eq. (2.61), we have

$$\begin{aligned}\Delta &= 4rs - (4r-s)\sum_{i=s+1}^{r} 2^{b_i-2} - 4s^2 - 3s\sum_{i=s+1}^{r} b_i - \frac{3}{4}\left(\sum_{i=s+1}^{r} b_i\right)^2 \\ &= 4rs - 4s^2 + 3s\sum_{i=s+1}^{r}(2^{b_i-2} - b_i) + 4(r-s)\sum_{i=s+1}^{r} 2^{b_i-2} - \frac{3}{4}\left(\sum_{i=s+1}^{r} b_i\right)^2\end{aligned}$$

Substituting Eq. (2.55) in the above, we have

$$\begin{aligned}\Delta &\geq 4rs - 4s^2 - 3s(r-s) + 4(r-s)\sum_{i=s+1}^{r} 2^{b_i-2} - \frac{3}{4}\left(\sum_{i=s+1}^{r} b_i\right)^2 \\ &= s(r-s) + (r-s)\sum_{i=s+1}^{r} 2^{b_i} - \frac{3}{4}\left(\sum_{i=s+1}^{r} b_i\right)^2 \end{aligned} \tag{2.63}$$

Subcase 1: $s = r$

In this case, $\Delta = 0$ and the case of a balanced hypercycle is obtained with $m_t = 4$, for $t = 1, \ldots, r$ and $\rho_t = 2$.

Subcase 2: $s = r - 1$

Simplifying Eq. (2.63), we obtain

$$\begin{aligned}\Delta &\geq s(r-s) + (r-s)2^{b_r} - \frac{3}{4}b_r^2 \\ &= (r-1) + 2^{b_r} - \frac{3}{4}b_r^2 \qquad \text{(Substituting } s = r-1\text{)} \\ &\geq (r-1) + \frac{5}{4} \qquad \text{(From Eq. (2.58))} \\ &= r + \frac{1}{4} > 0\end{aligned} \tag{2.64}$$

Subcase 3: $s \leq r - 2$

Expanding Eq. (2.63), we obtain

$$\begin{aligned}\Delta &\geq s(r-s) + (r-s)\sum_{i=s+1}^{r} 2^{b_i} - \frac{3}{4}\sum_{i=s+1}^{r} b_i^2 - \frac{3}{2}\sum_{t=s+1}^{r-1}\sum_{j=t+1}^{r} b_t b_j \\ &= s(r-s) + \frac{3}{4}\sum_{i=s+1}^{r}(2^{b_i} - b_i^2) + \left(r - s - \frac{3}{4}\right)\sum_{i=s+1}^{r} 2^{b_i} - \frac{3}{2}\sum_{t=s+1}^{r-1}\sum_{j=t+1}^{r} b_t b_j\end{aligned}$$

$$
\begin{aligned}
&= s(r-s) + \frac{3}{4}\sum_{i=s+1}^{r}(2^{b_i} - b_i^2) + \frac{1}{4}\sum_{i=s+1}^{r} 2^{b_i} + (r-s-1)\sum_{i=s+1}^{r} 2^{b_i} \\
&\qquad -\frac{3}{2}\sum_{t=s+1}^{r-1}\sum_{j=t+1}^{r} b_t b_j \\
&\geq s(r-s) - \frac{3}{4}(r-s) + 2(r-s) \quad \text{(From Eqs. (2.56) and (2.60))} \\
&\qquad + (r-s-1)\sum_{i=s+1}^{r} 2^{b_i} - \frac{3}{2}\sum_{t=s+1}^{r-1}\sum_{j=t+1}^{r} b_t b_j \\
&\geq s(r-s) - \frac{3}{4}(r-s) + 2(r-s) + \frac{5(r-s)(r-s-1)}{4} \quad \text{(From Lemma 2.18)} \\
&= (r-s)\left(s + \frac{5}{4} + \frac{5(r-s-1)}{4}\right) \\
&\geq 2(s+\frac{5}{2}) > 0 \quad \text{(Substituting the lower bound } r-s=2 \text{ in the above eqn.)}
\end{aligned}
$$

■

The preceding discussion naturally leads us to the question what is the optimal configuration when M is an odd power of two given that $\rho = 2$. Under this condition, it can be observed that $r = a/2$ is not an integer since $M = 2^a$ is odd. Intuitively, we can consider two possible solutions.

1. $r = \lceil \log_4 M \rceil$. Observe that there exists an m_i such that $m_i = 2$, $1 \leq i \leq r$ which would violate the connectivity constraint since $m_i < 2\rho$ for $\rho = 2$ and therefore we can preclude this solution.
2. $r = \lfloor \log_4 M \rfloor$. In this case, we have $m_t \geq 4$ for $t = 1, \ldots, r$ and it is obvious that the connectivity constraints are satisfied in all dimensions for $\rho = 2$.

We shall now state and prove the optimality of the second assumption as follows.

Theorem 2.9 *A hypercycle $\mathcal{G}^{\vec{\rho}}_{\vec{m}}$ with r^* dimensions has a cost optimal degree*diameter product such that $r^* = \lfloor \log_4 M \rfloor$, $\vec{m} = [4\ 4 \ldots 4\ 8\ 4 \ldots 4\ 4]$ and $\vec{\rho} = [2\ 2 \ldots\ 2\ 2\ 2 \ldots 2\ 2]$ if M is an odd power of 2 and $\rho_i = 2$, $1 \leq i \leq r$ for some r.*

Proof. Since $M = 2^a$, we have $r^* = \lfloor \frac{a}{2} \rfloor = \frac{a-1}{2}$. The node radix of the optimal hypercycle can be represented as $m_t = 4$, $1 \leq t \leq r^*$, $t \neq i$, $1 \leq i \leq r^*$ and $m_i = M/4^{r^*-1} = 2^{a-2(r^*-1)} = 8$ with $\rho = 2$. The degree of the network is then given by $3(r^*-1)+4$ while its diameter is $r^* - 1 + 2 = r^* + 1$. Hence the degree*diameter product is given by

$$
\begin{aligned}
\chi^* &= (3(r^*-1)+4)(r^*+1) = (3r^*+1)(r+1) \\
&= \frac{3a^2 + 2a - 1}{4}
\end{aligned} \tag{2.65}
$$

As before, the optimality of Eq. (2.65) can be established by comparing χ^* with all valid hypercycle topologies satisfying the constraints $\rho_t = 2$, $m_t \geq 2\rho_t$, $\forall t$, $1 \leq t \leq r$ such that $\Delta = \chi - \chi^* \geq 0$ where $r \leq r^*$.

Cost of Hypercycle topologies such that $M = \prod_{t=1}^{r} m_t$ where $m_t \geq 4$ and $\rho_t = 2$.

Let s be an integer such that $0 \leq s \leq r-1$ and $m_j = 4$, $1 \leq j \leq s$ with $m_i \geq 2^{b_i}$ where $b_i \geq 3$ and $(s+1) \leq i \leq r$. Observe that if $s = r$, then $\prod_{j=1}^{r} m_j = 4^r < M$ since $r \leq r^*$ and $4^{r^*} = M/2$. The degree of these topologies is $3s + 4(r-s)$ while the diameter is equal to $s + \frac{1}{4}\sum_{i=s+1}^{r} 2^{b_i}$ as obtained previously. Hence χ is given by Eq. (2.61).

Substituting the value of $a = 2s + \sum_{i=s+1}^{r} b_i$ in Eq. (2.65), we obtain

$$\begin{aligned}
\chi^* &= \frac{3}{4}\left(4s^2 + 4s\sum_{i=s+1}^{r} b_i + \left(\sum_{i=s+1}^{r} b_i\right)^2\right) + \frac{1}{4}\left(4s + 2\sum_{i=s+1}^{r} b_i\right) - \frac{1}{4} \\
&= 3s^2 + s + \left(3s + \frac{1}{2}\right)\sum_{i=s+1}^{r} b_i + \frac{3}{4}\left(\sum_{i=s+1}^{r} b_i\right)^2 - \frac{1}{4} \qquad (2.66)
\end{aligned}$$

Subtracting Eq. (2.66) from Eq. (2.61), we have

$$\begin{aligned}
\Delta &= 4rs - 4s^2 - s + (4r-s)\sum_{i=s+1}^{r} 2^{b_i-2} - \left(3s + \frac{1}{2}\right)\sum_{i=s+1}^{r} b_i - \frac{3}{4}\left(\sum_{i=s+1}^{r} b_i\right)^2 + \frac{1}{4} \\
&= 4rs - 4s^2 - s + \left(3s + \frac{1}{2}\right)\sum_{i=s+1}^{r}(2^{b_i-2} - b_i) + \left(4(r-s) - \frac{1}{2}\right)\sum_{i=s+1}^{r} 2^{b_i-2} \\
&\qquad - \frac{3}{4}\left(\sum_{i=s+1}^{r} b_i\right)^2 + \frac{1}{4}
\end{aligned}$$

Substituting Eq. (2.55) in the above, we have

$$\begin{aligned}
\Delta &\geq 4rs - 4s^2 - s - \left(3s + \frac{1}{2}\right)(r-s) + \left(4(r-s) - \frac{1}{2}\right)\sum_{i=s+1}^{r} 2^{b_i-2} \\
&\qquad - \frac{3}{4}\left(\sum_{i=s+1}^{r} b_i\right)^2 + \frac{1}{4} \\
&= s\left(r - s - \frac{1}{2}\right) - \frac{r}{2} + \left(r - s - \frac{1}{8}\right)\sum_{i=s+1}^{r} 2^{b_i} - \frac{3}{4}\left(\sum_{i=s+1}^{r} b_i\right)^2 + \frac{1}{4} \qquad (2.67)
\end{aligned}$$

Subcase 1: $s = r - 1$

Substituting $s = r - 1$ in Eq. (2.67), we obtain

$$\Delta \geq \frac{r-1}{2} - \frac{r}{2} + \frac{7}{8}2^{b_r} - \frac{3}{4}b_r^2 + \frac{1}{4} \qquad (2.68)$$

$$\begin{aligned} &\geq -\frac{1}{2}+\frac{1}{4}+\frac{1}{4} \qquad \text{(From Eq. (2.57))} \\ &= 0 \end{aligned}$$

Subcase 2: $s \leq r-2$

Expanding Eq. (2.67), we obtain

$$\begin{aligned} \Delta &\geq s\left(r-s-\frac{1}{2}\right)-\frac{r}{2}+\left(r-s-\frac{1}{8}\right)\sum_{i=s+1}^{r}2^{b_i}-\frac{3}{4}\sum_{i=s+1}^{r}b_i^2 \\ &\qquad -\frac{3}{2}\sum_{t=s+1}^{r-1}\sum_{j=t+1}^{r}b_t b_j+\frac{1}{4} \\ &= s\left(r-s-\frac{1}{2}\right)-\frac{r}{2}+\frac{3}{4}\sum_{i=s+1}^{r}(2^{b_i}-b_i^2)+\left(r-s-\frac{7}{8}\right)\sum_{i=s+1}^{r}2^{b_i} \\ &\qquad -\frac{3}{2}\sum_{t=s+1}^{r-1}\sum_{j=t+1}^{r}b_t b_j+\frac{1}{4} \\ &= s\left(r-s-\frac{1}{2}\right)-\frac{r}{2}+\frac{3}{4}\sum_{i=s+1}^{r}(2^{b_i}-b_i^2)+\frac{1}{8}\sum_{i=s+1}^{r}2^{b_i} \\ &\qquad +(r-s-1)\sum_{i=s+1}^{r}2^{b_i}-\frac{3}{2}\sum_{t=s+1}^{r-1}\sum_{j=t+1}^{r}b_t b_j+\frac{1}{4} \\ &\geq s\left(r-s-\frac{1}{2}\right)-\frac{r}{2}-\frac{3}{4}(r-s)+(r-s) \quad \text{(From Eqs. (2.56) and (2.59))} \\ &\qquad +(r-s-1)\sum_{i=s+1}^{r}2^{b_i}-\frac{3}{2}\sum_{t=s+1}^{r-1}\sum_{j=t+1}^{r}b_t b_j+\frac{1}{4} \\ &\geq s\left(r-s-\frac{1}{2}\right)-\frac{r}{2}+\frac{r-s}{4}+\frac{5(r-s)(r-s-1)}{4}+\frac{1}{4} \\ &\qquad \text{(From Lemma 2.18)} \\ &= (r-s)\left(s+\frac{1}{4}\right)-\frac{r+s}{2}+\frac{5(r-s)(r-s-1)}{4}+\frac{1}{4} \\ &= \frac{(r-s)(5r-4-s)}{4}-\frac{r+s}{2}+\frac{1}{4} \\ &= (r-s)(r-1)+\frac{(r-s)^2}{4}-\frac{r+s}{2}+\frac{1}{4} \\ &= r\left(r-s-\frac{3}{2}\right)+\frac{s}{2}+\frac{(r-s)^2}{4}+\frac{1}{4} \\ &\geq \frac{r+s}{2}+\frac{5}{4}>0 \quad \text{(Substituting the lower bound } r-s=2 \text{ in the above)} \end{aligned}$$

■

2.5.2 Other Optimization Cases:

Let $M = p^a$, $m = p^b$ where p is a prime number greater than 2 and let $\lfloor \frac{m}{2} \rfloor \bmod \rho = 0$. Then χ can be rewritten as

$$\chi = r2\rho * r \left\lceil \frac{\lfloor \frac{m}{2} \rfloor}{\rho} \right\rceil = r^2(m-1) = \frac{a^2(p^b-1)}{b^2} \tag{2.69}$$

Setting the first order derivative of Eq. (2.69) to zero, we obtain
$\frac{\partial \chi}{\partial b} = \frac{a^2}{b^3}\left(bp^b \log p - 2(p^b-1)\right) = 0$.

1. $p = 3$

   The analytical minimum is given by $b' = 1.4506$ and there exists two local integer optimums namely $b^* = 1$ and 2. The second order derivative given by

   $$\frac{\partial^2 \chi}{\partial b^2} = \frac{a^2[bp^b \log p\,(b \log p - 4) + 6(p^b-1)]}{b^4} \tag{2.70}$$

   is positive for all integer values of $b \geq 1$. Hence the following Lemma.

   Lemma 2.21 *A hypercycle with r^* dimensions has an optimal degree*diameter product such that*

   $$r^* = \begin{cases} \log_3 M & \text{when } M \text{ is a power of 3 and } \rho = 1 \\ \log_9 M & \text{when } M \text{ is an even power of 3 and } \rho \in \{2,4\} \end{cases}$$

2. $p = 5$

   The analytical solution for b is given by $b' = 0.9902$ and the integer minimum is given by $b^* = 1$.

   Lemma 2.22 *When M is a power of 5, a hypercycle with r dimensions has an optimal degree*diameter product such that $r = \log_5 M$ if $\rho = c$ where $c \in \{1,2\}$.*

2.6 Network Latency and Throughput

Hypercycles can be modeled as an M/M/1 communication network [25] in which the expression for the network latency T is given as

$$\begin{aligned} T &= \frac{\delta |\mathcal{E}_m^\rho|}{\mu C_p (1-\delta\gamma)} \\ &= \frac{\delta M d}{2\mu C_p (1-\delta\gamma)} \end{aligned} \tag{2.71}$$

where C_p is the capacity of the network, μ is the average service rate, δ is the average distance and γ being the utilization factor. The capacity of the network is the maximum bandwidth of the network while the utilization factor γ is the number of messages handled by the network

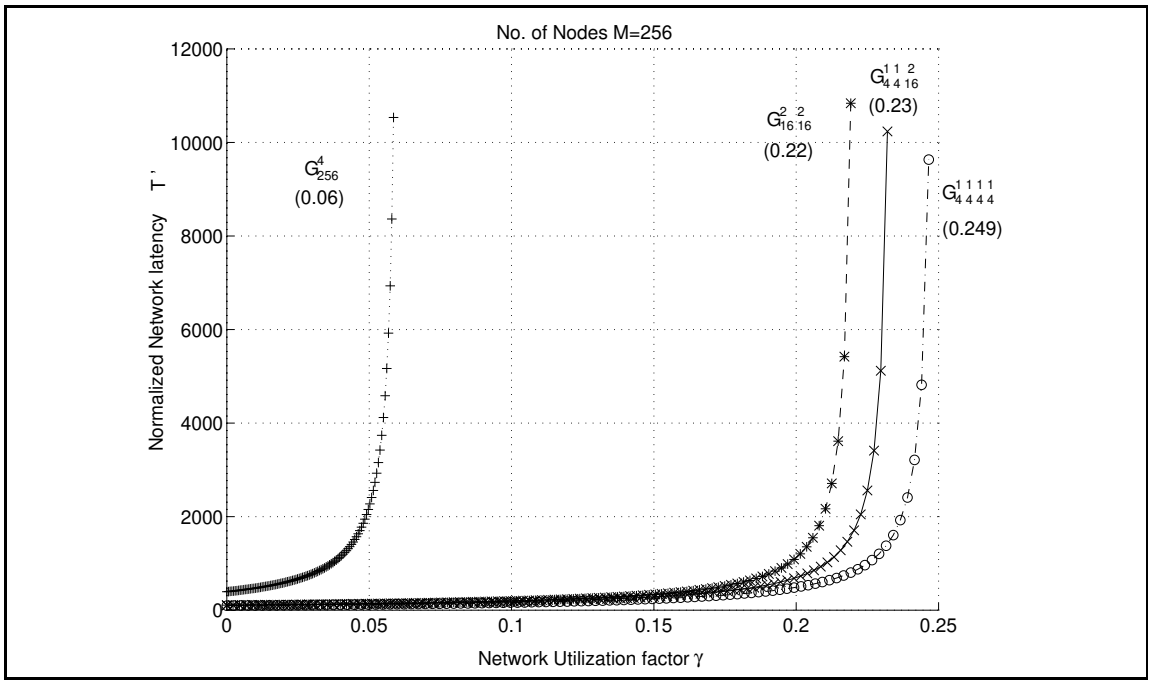


Figure 2.2 *Plot of normalized network latency in Hypercycles for $M = 256$ and $d = 8$. The maximum network utilization for each case is given in brackets*

and is normalized to the capacity C_p. In the above model, it is assumed that messages are independent and uniformly distributed. Furthermore, messages are routed using the shortest path or greedy routing [35] assuming packet switching with infinite queueing.

Taking μ, C_p and M as constants and normalizing the network latency, we obtain

$$T' = \frac{\delta d}{(1 - \delta\gamma)} \tag{2.72}$$

The network delay increases exponentially as the utilization factor increases, and the delay asymptotically saturates when $\gamma = 1/\delta$. Similarly, the average traffic density, T_d, is given by

$$\begin{aligned} T_d &= \frac{\text{Average distance} \times \text{Total number of nodes}}{\text{Total number of edges}} \\ &= \frac{\delta.M}{\left(\frac{Md}{2}\right)} = \frac{2\delta}{d} \end{aligned} \tag{2.73}$$

Hence the smaller the average distance, the greater is the utilization factor and the smaller is the network delay and traffic density for a given load. Let us analyze the network latency and utilization factors of two sets of hypercycles of constant degree.

- **Case-study 1: M - a power of two**

A plot of the normalized network latency constant degree hypercycles is illustrated

Table 2.9 *Case-study 2: Optimal Hypercycle Topologies for* $M = 256$ *with* $d = 8$

Dimension r	Optimized Hypercycle	Average Distance	Max. Network utilization Factor
1	$\mathcal{G}_{256}^{4}$	16.4392	0.0608
2	$\mathcal{G}_{16\ 16}^{2\ 2}$	4.5176	0.2213
3	$\mathcal{G}_{4\ 4\ 16}^{1\ 1\ 2}$	4.2667	0.2344
4	$\mathcal{G}_{4\ 4\ 4\ 4}^{1\ 1\ 1\ 1}$	4.0157	0.2490
5	$\mathcal{G}_{2\ 2\ 4\ 4\ 4}^{1\ 1\ 1\ 1\ 1}$	4.0157	0.2490
6	$\mathcal{G}_{2\ 2\ 2\ 2\ 4\ 4}^{1\ 1\ 1\ 1\ 1\ 1}$	4.0157	0.2490
7	$\mathcal{G}_{2\ 2\ 2\ 2\ 2\ 2\ 4}^{1\ 1\ 1\ 1\ 1\ 1\ 1}$	4.0157	0.2490
8	$\mathcal{G}_{2\ 2\ 2\ 2\ 2\ 2\ 2\ 2}^{1\ 1\ 1\ 1\ 1\ 1\ 1\ 1}$	4.0157	0.2490

in Fig. 2.2 where $M = 256$ and $d = 8$. The average distance and utilization factors are listed in Table 2.9 for this example. It is interesting to see in this example that structures which are isomorphic to the 8-dimensional hypercube have maximum network utilization, e.g. the 4-dimensional nearest neighbour hypercube. This result essentially follows the discussion in Section 2.4.3 where the average distance can be decreased by having the maximum number of dimensions with $\rho = 1$ and M being a power of 2.

- **Case-study 2: M - not a power of two**

  As an example consider $M = 400$ and $d = 24$. From discrete optimization, optimal hypercycles for each dimension with maximum network utilization have been listed in Table. 2.10. Note that the 2-dimensional hypercycle $\mathcal{G}_{16\ 25}^{6\ 6}$ corresponding to the violation case (Family 2) presents the best performance among the enlisted graphs.

The *throughput* T_p of the network is defined as the number of messages handled by the network per unit time, and is inversely proportional to the traffic density. Therefore

$$\begin{aligned} T_p &= \frac{M}{T_d T} \\ &= \frac{\left(\frac{Md}{2\delta}\right)}{T} \\ &\propto \frac{(1-\delta\gamma)}{\delta^2} \end{aligned}$$

Thus the normalized throughput of the network is given as $T_p' = \frac{(1-\delta\gamma)}{\delta^2}$.

Table 2.10 *Case-study 2: Optimal Hypercycle Topologies for* $M = 400$ *with* $d = 24$

Dimension r	Optimized Hypercycle	Average Distance	Max. Network utilization Factor
1	$\mathcal{G}_{400}^{12}$	8.8195	0.1134
2	$\mathcal{G}_{20\ 20}^{6\ 6}$	2.6065	0.3837
2*	$\mathcal{G}_{16\ 25}^{6\ 6}$	2.5714	0.3889
3	$\mathcal{G}_{2\ 10\ 20}^{1\ 5\ 7}$	2.6065	0.3837
4	$\mathcal{G}_{2\ 2\ 5\ 20}^{1\ 1\ 2\ 9}$	2.8070	0.3563
5	$\mathcal{G}_{2\ 2\ 2\ 2\ 25}^{1\ 1\ 1\ 1\ 10}$	3.1278	0.3197

2.7 Conclusion

In this chapter, a rigorous mathematical analysis of the average distance, diameter and degree*diameter product has been attempted and it provides some insight into the optimal design of hypercycle topologies. Analytical solutions, wherever possible have been derived and this has not been addressed in detail in the open literature to the best of the author's understanding. In essence, we have shown in this work that a balanced hypercycle is an optimal network only when the diameter k_i is an integral multiple of the connectivity ρ_i in each dimension i, i.e., $\lfloor \frac{m_i}{2} \rfloor \bmod \rho_i = 0$, $i = 1, \ldots, r$. When this is not the case, the balanced hypercycle degenerates into a quasi-optimal[¶] structure and some interesting families of counter examples have been found.

It is a significant result that shows that a well balanced and symmetrical network is not necessarily the true optimum for arbitrary input parameters. The exposition of this result applies to all graphs encompassed by the hypercycle including the hypercubes, k-ary hypercubes, GHC, toruses and circulant graphs. For the r-dimensional torus, hypercubes and k-ary hypercubes, $\rho_i = 1$, while for the GHC, $\rho_i = \lfloor \frac{m_i}{2} \rfloor$, $i = 1, \ldots, r$. It immediately follows that $\lfloor \frac{m_i}{2} \rfloor \bmod \rho_i = 0$ in all these cases and hence the balanced network provides an optimal structure. For cartesian products of circulant graphs (similar to hypercycles) whose connectivity $\rho_i \geq 2$, the abnormality presented here applies. The hypercycle networks have been developed primarily for message passing concurrent computers and the optimization study conducted in this Chapter will help in choosing a network configuration for a given application.

[¶]A quasi-optimal point is a point which lies in a neighbourhood of the true optimum.

Chapter 3

Embedding Complete Binary Trees in Hypercycles

3.1 Introduction

The issue of graph embeddings [44, 45] has been addressed for a variety of computationally intensive problems such as those arising in the generation of efficient VLSI layouts and algorithms for parallel architectures. In a parallel computer, a given algorithm is modeled by some communication structure that permits the exchange of messages that occur between processors [46]. Such a structure can be represented as a precedence graph wherein nodes correspond to processors while edges represent the exchange of information between them. Typical examples of communication structures include chains, cycles, Hamiltonian cycles, meshes, k-dimensional toruses and binary trees.

A majority of the papers in the open literature concentrate on embedding schemes for the n-dimensional hypercube (represented as Q_n) since this is one of the popular and efficient interconnection networks [46]. A survey on embedding graphs in interconnection networks such as grids, meshes, binary trees, pyramids and hypercubes can be found in [47].

The embedding of chains and cycles into hypercubes was considered by Chan and Saad [48] while the embedding of rectangular grids into hypercubes was discussed in [49, 50]. Some of the important cost functions that measure the quality of an embedding scheme of a problem graph $H = (V, E)$ onto a target graph $G = (V, E)$ include the following:

1. *Expansion* - The ratio of the size (number of nodes) in the target graph G to the size of the problem graph H, namely $X = \frac{|V(G)|}{|V(H)|}$.
2. *Dilation* - The length of the longest path in G which is mapped by any edge of H.
3. *Load Factor* - The load factor is the maximum number of nodes of H that are mapped to the same node in G.

The expansion, X, refers to the processor space complexity of G and $X-1$ denotes the relative number of idle processors. Observe that the embedding is cost optimal if $X=1$. Dilation measures the communication delay cost between nodes. Ideally, unit dilation gives the optimal time complexity for the given communication structure while the load factor reflects on the queuing delay of the messages.

Binary trees constitute an important class of data structures and computational structures which frequently arise in many algorithms and parallel architectures [51]. Typical examples include sorting and searching [52] which use divide-and-conquer strategies, arithmetic circuits such as adders, comparators [53], solution of banded and sparse systems in linear algebra [54], functional & logic programming and in the database and dictionary machines [55, 56]. Apart from graph theoretical interests, the ability to simulate a binary tree enhances the general purpose capability of the host machine.

Several authors address the topic of embedding binary trees in interconnection networks, particularly the hypercube, and Wagner's thesis [57] provides an extensive bibliography on the same. Havel *et. al.* [58] established upper and lower bounds on the dimensionality of the hypercube for embedding complete k-ary trees. It has been shown in [58]-[60] that a complete binary tree[†] T_h with unit dilation can be embedded in Q_{h+2} with an expansion of 2.00 but not in Q_{h+1}. No embedding permits both unit dilation and unit expansion in a hypercube. Nebesky [61] showed that the double rooted complete binary tree is a spanning tree of Q_{h+1}. Bhatt and Ipsen [54] discuss a recursive algorithm for embedding T_h in Q_{h+1} but with a single edge having dilation 2. For arbitrary binary trees, their algorithm gives a hypercube with a maximum size of $\mathcal{O}(N^{1.71})$ nodes and as small as $\mathcal{O}(N^{1.31})$ [57, pp. 80]. Wagner [62] describes another algorithm for embedding arbitrary binary trees in a hypercube with unit dilation/load factor with an expansion of $\mathcal{O}(\log N)$.

An embedding[‡] of T_h onto a hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is a one-to-one mapping $\phi : V(T) \Rightarrow \mathcal{G}_{\vec{m}}^{\vec{\rho}}$ for some $\vec{m}$ and $\vec{\rho}$ such that for each edge $(x,y) \in E(T)$, $(\phi(x), \phi(y)) \in E(G)$. This definition of embedding can be construed as relabelling the nodes of T_h with the nodes of $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ such that the edge adjacency of $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is preserved.

In this Chapter, the problem of embedding a complete binary tree on a hypercycle based parallel computer is investigated. Section 3.2 reviews the well known problem of bandwidth minimization [63] for embedding general graphs (or trees in our case) on a linear array. This method produces a unit dilation, unit expansion and unit load factor embedding of trees

[†]A complete binary tree of height h is denoted by T_h having $2^{h+1}-1$ nodes.

[‡]Unless otherwise mentioned, a tree or a binary tree implies a complete binary tree.

on a 1-dimensional hypercycle but the growth rate in the number of links is unviable for large trees. To circumvent this overhead, Section 3.3 presents a multi-dimension embedding algorithm which achieves unit dilation but with an expansion greater than 1. An optimal hypercycle that embeds T_h is derived in Section 3.4 through integer programming techniques and its performance/cost benefit is compared with the hypercubes. It will be shown that this network has an asymptotic expansion of 1.25 which is efficient compared to the best known unit dilation embeddings in hypercubes.

Definition 3.1 *Let p be the root of a complete binary tree T_h. Define T^0_{h-1} and T^1_{h-1} as the left and right subtrees of p of height $h-1$ where the superscript denotes the address of the respective subtree. Observe that there exists 2^i unique subtrees of height $h-i$ for some i such that $1 \leq i \leq h$. Then $T^{a_1,\ldots,a_i}_{h-i}$ is a subtree of height $h-i$ where $(a_{i-1},\ldots,a_0)$ is the binary representation of some a such that $0 \leq a \leq 2^i-1$ and $i = 1,\ldots,h$.*

3.2 Bandwidth problem for trees

Given an undirected graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$, a one-to-one mapping $\pi : \mathcal{V} \rightarrow \mathcal{S}$ where $S = \{1,2,\ldots,|\mathcal{V}(\mathcal{G})|\}$ constitutes a labelling of $\mathcal{G}$. For a given labelling π, we define

$$b_\pi(\mathcal{G}) = \max_{(u,v)\in\mathcal{E}(\mathcal{G})} |\pi(u) - \pi(v)| \tag{3.1}$$

The bandwidth of $\mathcal{G}$ denoted by $b(\mathcal{G})$ is the minimum value of $b_\pi(\mathcal{G})$ for all possible labellings of $\mathcal{G}$. This is the so called *bandwidth minimization problem* in graphs and matrices which has been rigorously studied in the past and the reader is referred to [64] for an excellent survey. It has been shown in [65] that the problem of determining the bandwidth of a graph is $\mathcal{NP}$-complete even for the case of trees of maximum degree 3. Upper and lower bounds have been derived for the bandwidth of several graphs. For the case of a complete binary tree T_h, it has been shown [66] that

$$b(T_h) \quad \geq \quad \left\lceil \frac{2^h-1}{h} \right\rceil \tag{3.2}$$

Table 3.1 summarizes the bandwidth of T_h while Fig. 3.1 gives some examples of embeddings of T_h on a linear array using this bandwidth. Observe that for the bandwidth labelling, there can exist several permutations that achieve the same bandwidth. Variations of the bandwidth problem include the min-cut linear arrangement [67], optimal linear arrangements of trees [68] and the tree embedding in VLSI [69]-[72] that have been studied. If an embedding of a tree on a line as defined by the bandwidth problem can be achieved, then im-

Table 3.1 *Bandwidth of some complete binary trees*

h	$\lvert T_h \rvert$	Lower bound of $b(T_h) = \left\lceil \frac{2^h - 1}{h} \right\rceil$
1	3	1
2	7	2
3	15	3
4	31	4
5	63	7
6	127	11
7	255	19
8	511	32
9	1023	57
10	2047	103

plicitly, we have obtained an embedding on a circulant graph[§] $C_{T_h}\langle\{1, 2, \ldots, b(T_h)\}\rangle$ whose maximum chord length is given by the bandwidth $b(T_h)$. Observe that a 1-dimensional hypercycle is isomorphic to the circulant graph defined above and hence an embedding of T_h on a 1-dimensional hypercycle can be realized.

While addressing the algorithmic aspects, it must be mentioned that due to the $\mathcal{NP}$-completeness of the problem, the question "Does the graph $\mathcal{G}$ have a numbering such that the bandwidth of $\mathcal{G} \leq b$ where b is a given integer" remains computationally intractable. For a constant b however, it has been shown by Saxe [64, 73] that the bandwidth problem always has a polynomial solution whose complexity is $\mathcal{O}(f(b)n^{b+1})$ where $f(b)$ is a function that depends on b. Approximate algorithms for obtaining the bandwidth labelling in general graphs have been proposed, such as the Cuthill-Mckee and the Gibbs-Poole-Stockmeyer algorithm [64]. These methods start with some initial assignment of node labels and try to optimize the numbering such that the effective bandwidth is closer to the optimum. The performance of these algorithms have been verified only experimentally [74] and they exhibit an average run-time complexity of $\mathcal{O}(n^{1.5})$ and $\mathcal{O}(n^{1.3})$ respectively for the two algorithms mentioned above with the latter achieving a better bandwidth than the former.

For a complete binary tree, Heckmann *et. al.* [75] give a polynomial time algorithm, Tree_into_line, that labels T_h on a line achieving the lower bound of Eq. 3.2. We shall briefly

[§] $C_{T_h}\langle\{1, 2, \ldots, b(T_h)\}\rangle$ is a graph with T_h nodes such that node j is adjacent to nodes $j \pm 1, j \pm 2, \ldots, j \pm b(T_h) \pmod{T_h}$ for $j \in V(T_h)$.

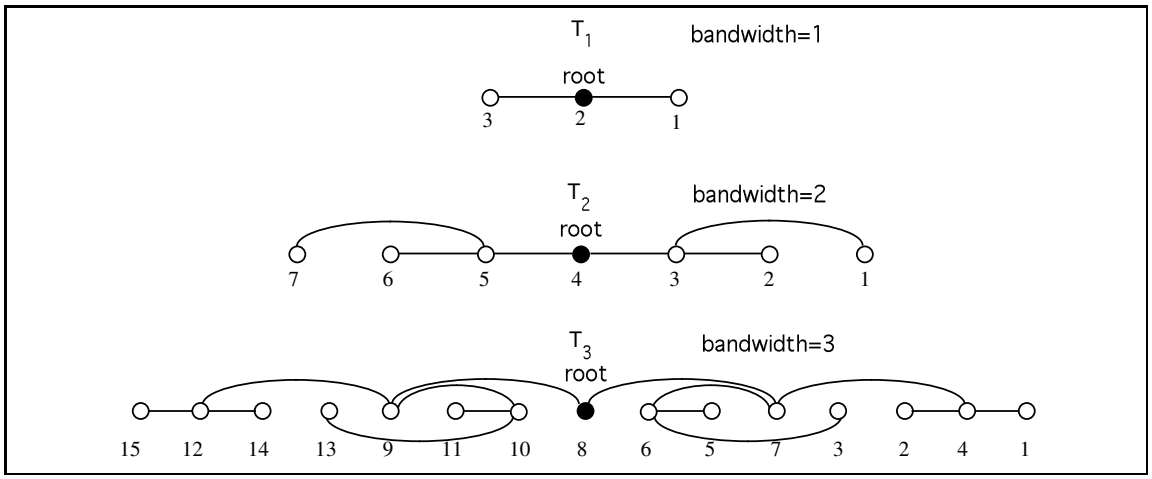


Figure 3.1 *Examples of embedding T_h on a line with a given bandwidth*

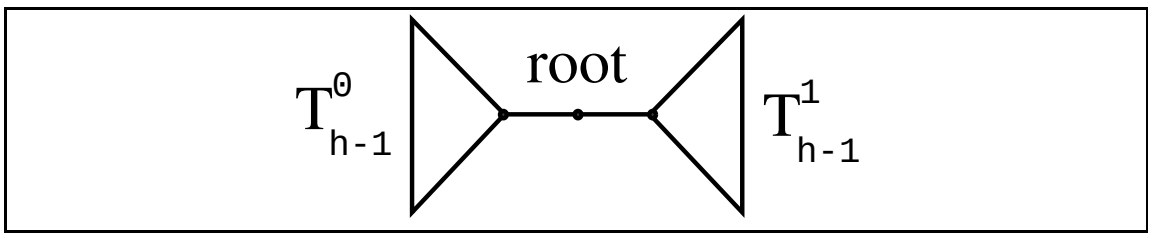


Figure 3.2 *Placement structure used by the* Tree_into_line *algorithm for embedding T_h on a line*

sketch their algorithm and the interested reader is referred to [75] for the technical details and proofs. The Heckmann algorithm places the nodes of one of the sub-trees of the root r, say, T^1_{h-1}[¶] into blocks, B_i from left to right such that B_i is of size $b(T_h) = \left\lceil \frac{2^h-1}{h} \right\rceil$ for $1 \leq i \leq h$ and $B_i = \{(i-1)b(T_h), \ldots, ib(T_h) - 1\}$. In the first step, B_1 is filled with $b(T_h)$ leftmost leaves of T^1_{h-1} in left to right fashion. In the i^{th} step, B_i is filled with the parents of nodes located in B_{i-1} along with the descendents of nodes in B_i that are not already placed. Finally, the remaining free positions are filled the leftmost unplaced leaves of T^1_{h-1}. The root r is placed in the last empty location of block B_h.

Let Tree_into_line(α, T_h) be an algorithm that assigns T_h to nodes on a 1-dimensional hypercycle $\mathcal{G}^{\rho}_m$ where the node α holds the root with $m = |T_h|$ and $\rho = \left\lceil \frac{2^h-1}{h} \right\rceil$. Examples of some tree embeddings in 1-dimensional hypercycles using Heckmann *et. al.*'s algorithm is shown in Fig 3.3. Using this algorithm as the basis, we present a methodology for embedding T_h in multi-dimensional hypercycles with unit dilation that minimizes the connectivity and expansion such that the resultant embedding is comparable if not more efficient than other structures such as the hypercube. It can be observed that the connectivity of the hypercycle is equivalent to the lower bound of the bandwidth, i.e. $b(T_h) = \lceil \frac{2^h-1}{h} \rceil$ while the degree is $2\lceil \frac{2^h-1}{h} \rceil$.

3.3 Embedding T_h in multi-dimensional hypercycles

Observe that even for the lower bound of Eq. 3.2, the degree of the target system increases rapidly for large h and will prove a potential bottleneck in terms of practical realization. In this section, we consider some heuristics to reduce the complexity in the number of links (degree) of the target hypercycle for embedding of a tree T_h with unit dilation by considering multiple dimensions. As a penalty, this may inadvertently lead to a hypercycle with expansion greater than unity. In the simplest case, let us consider embedding T_h in a 2-dimensional hypercycle such that the left subtree of the principal root α consisting of $2^h - 1$ nodes is embedded in a 1-dimensional hypercycle with $m = 2^h$ nodes, $\rho = \lceil \frac{2^{h-1}-1}{h-1} \rceil$. The root α and the right subtree consisting of a total of 2^h nodes can be embedded in another 1-dimensional hypercycle with $m = 2^h$ nodes and $\rho = \lceil \frac{2^{h-1}-1}{h-1} \rceil$.

These two 1-dimensional hypercycles $\mathcal{G}^{\rho}_m$ are subgraphs of a two-dimensional hypercycle $\mathcal{G}^{1\rho}_{2m}$ on which the complete binary tree T_h is now embeddable with a dilation of 1 and

[¶]The other subtree T^0_{h-1} is a mirror image of the embedding of T_h if r is placed in the middle of the line as shown in Fig. 3.2.

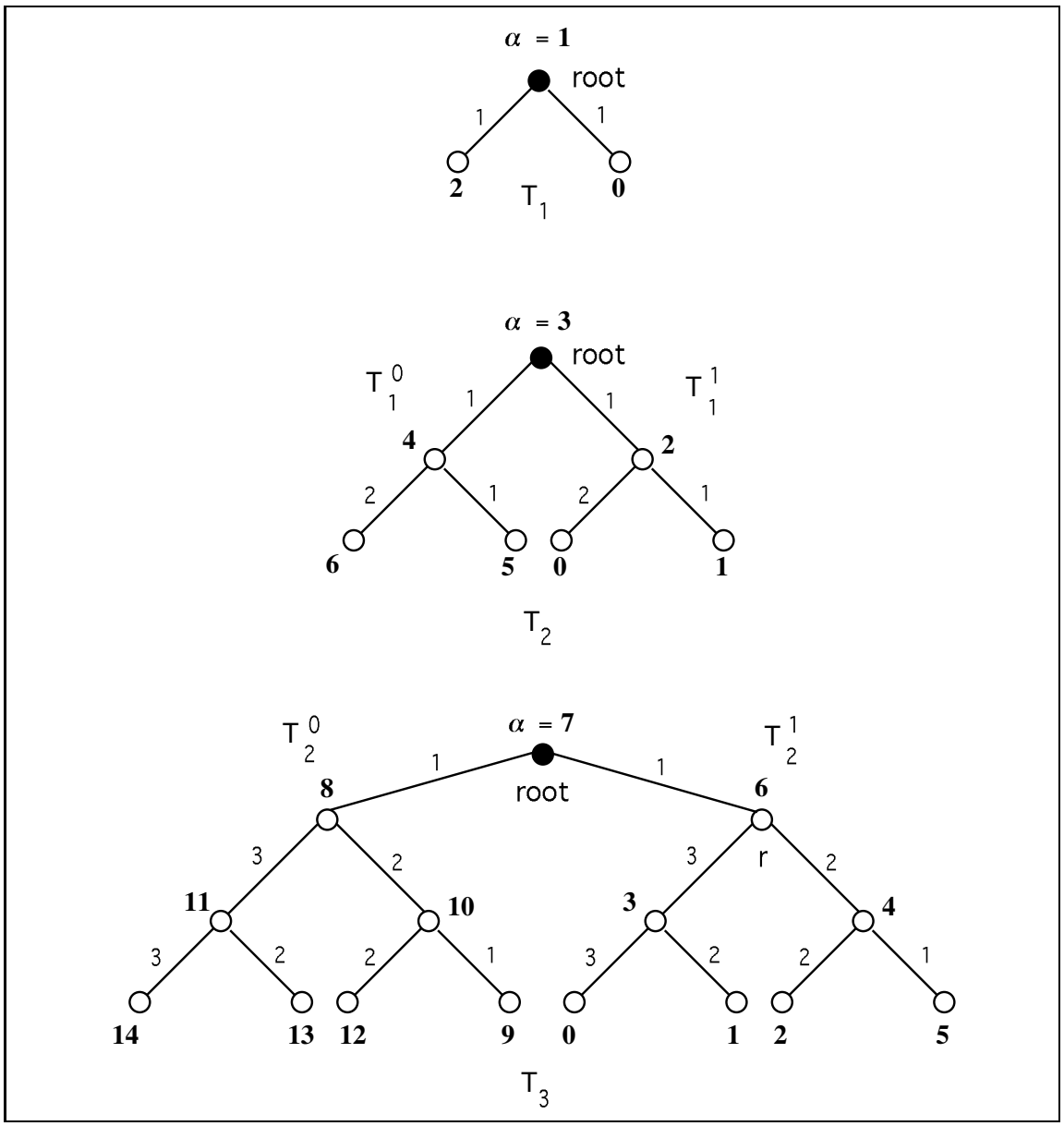


Figure 3.3 *Embedding of T_1, T_2 and T_3 in hypercycles using* Tree_into_line *embedding algorithm. The numbers on the edges represent the difference in connectivity between adjacent nodes*

(1) **Procedure** Two_dim_embed(α,T_h)
(2) **begin**
(3) Name the root of T_h with address α.
(4) Name the right child of the node named α in T_h with address
(5) $\beta_2 = (\alpha_2 - 2) \bmod m_2$, $\beta_1 = \alpha_1$.
(6) Call tree_into_line(κ,T^{11}_{h-2}) where $\kappa_1 = (\beta_1 - 1) \bmod m_1$, $\kappa_2 = \beta_2$.
(7) Call tree_into_line(κ',T^{10}_{h-2}) where $\kappa'_1 = (\beta_1 - 2) \bmod m_1$, $\kappa'_2 = \beta_2$.
(8) Name the left child of the node named α in T_h with address
(9) $\gamma_2 = (\alpha_2 - 1) \bmod m_2$, $\gamma_1 = \alpha_1$.
(10) Call tree_into_line(η,T^{01}_{h-2}) where $\eta_1 = (\gamma_1 - 3) \bmod m_1$, $\eta_2 = \gamma_2$.
(11) Call tree_into_line(η',T^{00}_{h-2}) where $\eta'_1 = (\gamma_1 - 4) \bmod m_1$, $\eta'_2 = \gamma_2$.
(12) **end** /* of Procedure Two_dim_embed */

Figure 3.4 *Procedure Two_dim_embed for embedding a tree T_h in a 2-dimensional hypercycle $\mathcal{G}^{\rho_1\rho_2}_{m_1 m_2}$ where $m_1 = 5$, $m_2 = 2^{h-1} - 1$, $\rho_1 = 2$ and $\rho_2 = \lceil \frac{2^{h-2}-1}{h-2} \rceil$*

expansion of $\frac{2^{h+1}}{2^{h+1}-1}$. The degree of $\mathcal{G}^{1\rho}_{2m}$ is $2\left\lceil \frac{2^{h-1}-1}{h-1} \right\rceil + 1$ which is an improvement as compared to the 1-dimensional case of $2\left\lceil \frac{2^h-1}{h} \right\rceil$ but is still expensive for large h. A question therefore naturally arises whether the expansion and the number of links can be minimized by increasing the dimensionality of the hypercycle. Unfortunately, due to the increased variability in the design parameters of a hypercycle, namely, the dimension r, the radix vector $\vec{m}$ and the connectivity vector $\vec{\rho}$, it is difficult to synthesize an algorithm that will optimize the embedding of T_h. We first discuss a procedure for embedding T_h in a 2-dimensional hypercycle with unit dilation and load factor and then use it for embedding T_h in an r-dimensional hypercycle which will be discussed later on.

Theorem 3.1 *Algorithm* Two_dim_embed *embeds T_h ($h > 2$) with unit dilation in a 2-dimensional hypercycle $\mathcal{G}^{\vec{\rho}}_{\vec{m}}$ where $\vec{m} = [m_1\ m_2] = [5\ 2^{h-1}-1]$ and $\vec{\rho} = [\rho_1\ \rho_2] = [2\ \lceil \frac{2^{h-2}-1}{h-2} \rceil]$.*

Proof. By construction. Algorithm Two_dim_embed is illustrated in Fig. 3.4 for embedding T_h for $h > 2$ in a 2-dimensional hypercycle. From Definition 3.1, it can be observed that there are two T_{h-1} sub-trees and four T_{h-2} sub-trees for a given tree T_h with respect to the root. It has been assumed previously that T_{h-2} is embeddable with unit dilation in dimension j of $\mathcal{G}^{\vec{\rho}}_{\vec{m}}$ such that $\rho_2 = \lceil \frac{2^{h-2}-1}{h-2} \rceil$ and $m_2 = 2^{h-1} - 1 = |T_{h-2}|$. We can

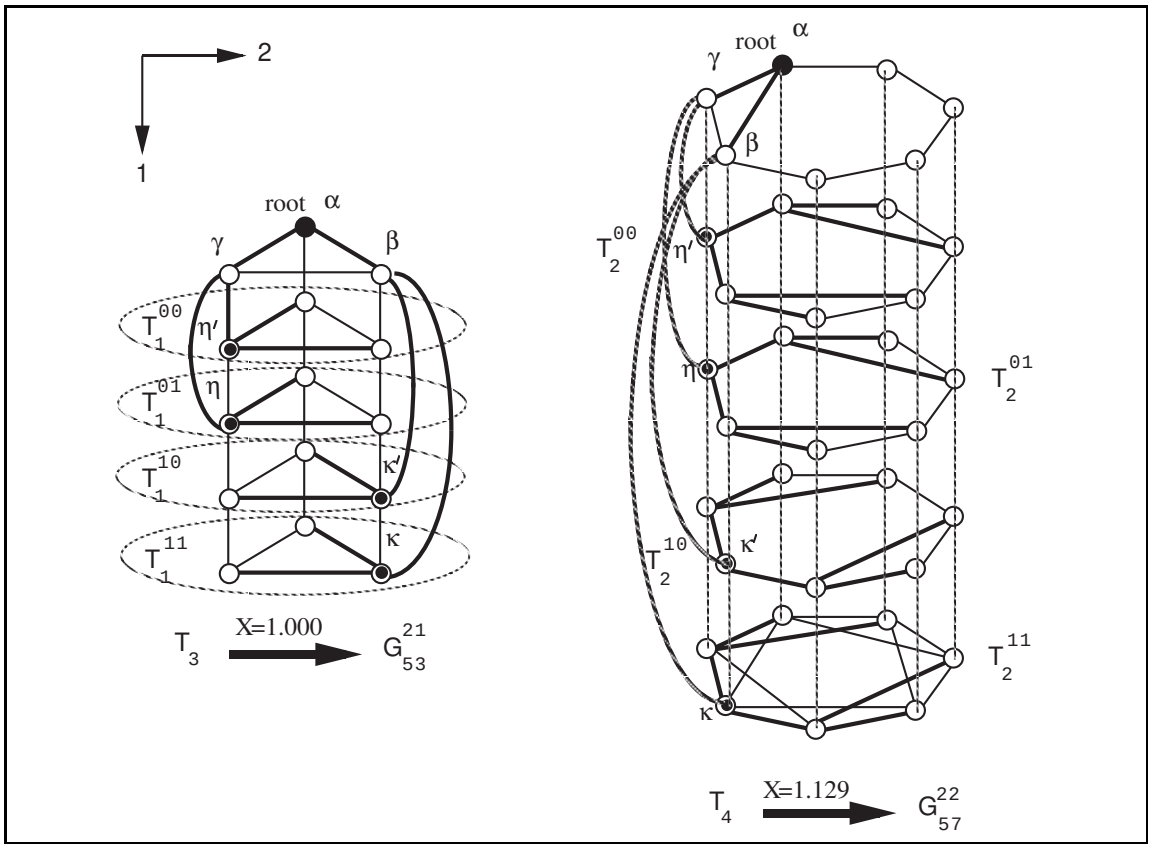


Figure 3.5 *Some examples of unit dilation embedding using Algorithm* Two_dim_embed. *Only the relevant edges are shown for clarity*

synthesize a 2-dimensional hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}} = \mathcal{G}_{m_1}^{\rho_1} \times \mathcal{G}_{m_2}^{\rho_2}$ where $\rho_1 = 2$, $m_1 = 5$, $\rho_2 = 2^{h-3}$ and $m_2 = 2^{h-1} - 1$ such that there are 5 1-dimensional hypercycles $\mathcal{G}_{m_2}^{\rho_2}$. Each of T_{h-2} is embedded in one of the four dimensions while the top 3 nodes of the tree including the root are placed in the fifth 1-dimensional hypercycle. In Lines (3)-(11), the address labels of the children of α, namely β, γ, the grandchildren κ, κ', η, and η', are computed. The labels κ, κ', η, and η' constitute the root addresses of T_{h-2}^{11}, T_{h-2}^{10},T_{h-2}^{01} and T_{h-2}^{00} respectively. By construction, we have ensured that nodes κ, κ', η, and η' are (i) mapped to distinct addresses and (ii) that there are edges in $\mathcal{G}_{m_1 m_2}^{\rho_1 \rho_2}$ which connect κ, κ', η, and η' to their parents (see Lines(6),(7),(10) and (11)). Fig. 3.5 illustrates an example of embedding T_4 in $\mathcal{G}_{57}^{22}$ using algorithm Two_dim_embed. ■

3.3.1 Multidimensional embedding algorithm

We are now in a position to discuss the algorithm for embedding a tree in a multidimensional hypercycle using procedure Two_dim_embed as the basis.

3.3.1.1 Construction

The multidimensional embedding algorithm essentially uses a divide-and-conquer strategy to place the right and left child of the root in non identical dimensions of a r-dimensional hypercycle.

We first present an informal discussion of the algorithm through a bottom-up approach for clarity. Let $T_{h'}$ be a tree that can be embedded using the algorithm discussed in Fig. 3.4. Observe that only three nodes are embedded on the fifth 1-dimensional plane of the 2-dimensional hypercycle (see Fig. 3.5). We shall refer to this plane as the *limiting plane*. The vacancies in the *limiting plane* can be exploited to potentially embed a larger tree by increasing the dimensionality by unity for each increase in h.

If we denote by $\mathcal{G}_{\vec{m_h}}^{\vec{\rho_h}}$, the hypercycle that can embed a tree T_h as per the algorithm shown in Fig. 3.4, then the tree T_{h+1} can be embedded in hypercycle $\mathcal{G}_{2\vec{m_h}}^{1\vec{\rho_h}}$ as follows. Observe that $\mathcal{G}_{2\vec{m_h}}^{1\vec{\rho_h}}$ is composed of two hypercycles $\mathcal{G}_{\vec{m_h}}^{\vec{\rho_h}}$. We can therefore embed the two subtrees of T_{h+1} into the two hypercycles $\mathcal{G}_{\vec{m_h}}^{\vec{\rho_h}}$ as per algorithm Two_dim_embed and place the root in one of the vacancies of the limiting plane of the hypercycle where the right (left) subtree was embedded. We ensure that the root of T_h is embedded in such a way that it is *distance one* away from the roots of T_h^0 and T_h^1. The mapping procedure for the root nodes of the various subtrees will be discussed in Section 3.3.1.2.

The above process is applied recursively as the tree becomes taller until the limiting plane of the right most 2-dimensional hypercycle is filled. The height h of the subtree that we have initially chosen to embed in a 2-dimensional hypercycle will determine the number of nodes in the limiting plane and this will consequently dictate the largest tree that can be embedded using this method.

Given therefore an arbitrary complete binary tree, we can determine the optimum height h of the subtree that we need to choose to embed in a 2-dimensional hypercycle according to algorithm Two_dim_embed in such a manner that the original tree is embeddable according to the scheme oulined above. In the next section, we shall discuss the multidimensional embedding algorithm in detail and identify the optimal hypercycle for embedding T_h.

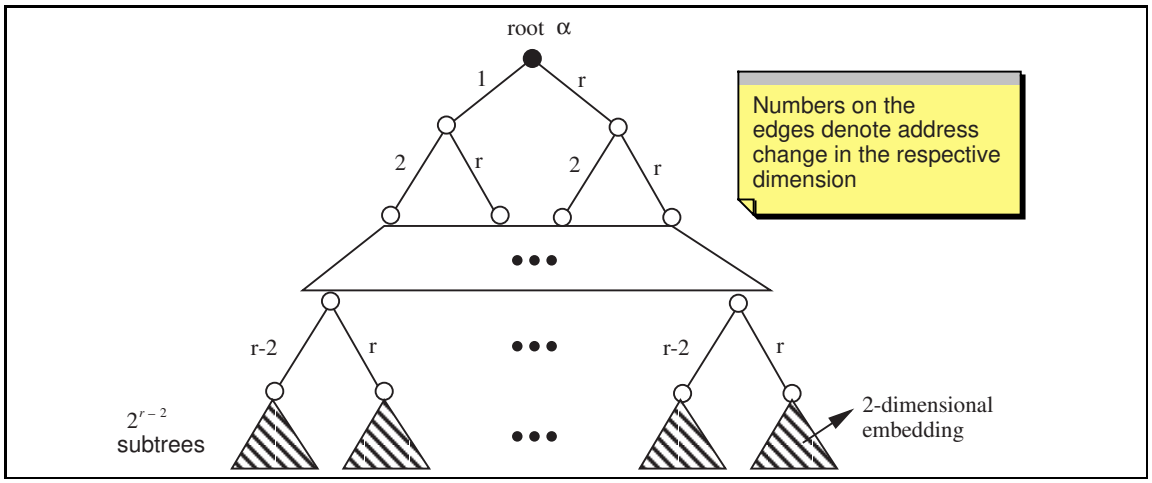


Figure 3.6 *Pictorial representation of Algorithm* Multidim_Tree_embed

3.3.1.2 Discussion

In this scheme as shown in Fig. 3.6, the principal root of T_h is placed at a given address α in the r-dimensional hypercycle. The left child of the root, γ is positioned in dimension i, $1 \leq i \leq r-2$ at address $(\alpha_1, \alpha_2, \ldots, \alpha_i - 1, \ldots, \alpha_r)$ while the right child, β is placed in dimension r with address $(\alpha_1, \alpha_2, \ldots, \alpha_i, \ldots, \alpha_r - 1)$. By recursively repeating this procedure for each subtree T_{h-i}, there exist 2^{r-2} subtrees of height $h-(r-2)$ which can be embedded in two dimensions using procedure Two_dim_embed.

Algorithm Multidim_Tree_embed shown in Fig. 3.7 uses another procedure Node_label given in Fig. 3.8 to perform recursive labelling. In Algorithm Multidim_Tree_embed shown in Fig. 3.7, after the root of T_h is labelled in Fig. 3.7 Line (3), a check is made on the dimension r (Lines(5)-(11)) and one of the appropriate procedures, namely

1. Tree_into_line
2. Two_dim_embed
3. Node_label

is invoked. Algorithm Node_label given in Fig. 3.8 is a recursive labelling procedure that has three parameters passed during its function call. This include the tree, T_h, an index i to mark the dimension where the embedding is to be made and finally a string variable s that unique identifies the subtrees of T_h. In Lines (3) and (4) of Fig. 3.8 computes the addresses of γ (left child) and β (right child) that are distance *one* away with respect to α. In addition, s' and s'' are two string variables (see Definition 3.1 in page 54) that define the addresses of the left and right subtrees, T_{h-i}, $1 \leq i \leq r-2$. In Lines (8) and (9),

```
(1)  Procedure Multidim_Tree_embed(α,r,T_h)
(2)      begin
(3)          Name the root of T_h with address α.
(4)          Initialize st as a null string.
(5)          if r > 2 then
(6)              Call Node_label(α,1,st)
(7)          elseif r = 2 then
(8)              Call Two_dim_embed(α,T_h)
(9)          else /* r=1 */
(10)             Call Tree_into_line(α, T_h)
(11)         end
(12)     end /* of Multidim_Tree_embed */
```

Figure 3.7 *Procedure* Multidim_Tree_embed *for embedding* T_h *in hypercycles*

procedure Node_label labels the roots of $T^{s'}_{h-i}$ and $T^{s''}_{h-i}$ with labels γ and β respectively. Procedure Node_label is recursively called in Lines (11) & (12) $\forall i$, $1 \leq i \leq r-3$. Thus, the total number of address labels, A, (including the principal root α) generated at the end of Line (13) when $i = r-3$ is given by

$$\begin{aligned} A &= 1 + 2^1 + 2^2 + \ldots + 2^{r-3} \\ &= 2^{r-2} - 1 \end{aligned} \tag{3.3}$$

The number of nodes, M_r, that remain to be labelled in T_h is given by

$$\begin{aligned} M_r &= |T_h| - A \\ &= 2^{h+1} - 1 - (2^{r-2} - 1) \\ &= 2^{r-2}(2^{h-(r-3)} - 1) \\ &= 2^{r-2}|T_{h-(r-2)}| \end{aligned} \tag{3.4}$$

From the algorithm, it can be seen that a total of 2^{r-3} invocations of procedure Node_label will be spawned when $i = r-3$ at the end of Line (9). Now, the index variable i is incremented in Line (10) to $r-2$ and each of the 2^{r-3} invocations executes Lines (14) & (15). Hence total number of invocations of procedure Two_dim_embed is $2 \times 2^{r-3} = 2^{r-2}$. At this point, the height of the tree that remains to be embedded is given by $T_{h-(r-2)}$ and we can use procedure Two_dim_embed to embed $T_{h-(r-2)}$ with unit dilation and unit load factor in the $r-1^{th}$ and r^{th} dimensions. The number of nodes labelled by procedure Two_dim_embed

(1) **Procedure** Node_label(α,i,s)
(2) **begin**
(3) Let $\gamma_i = (\alpha_i - 1) \bmod m_i$, $\gamma_k = \alpha_k$, $k = 1, \ldots, r$, $k \neq i$.
(4) Let $\beta_r = (\alpha_r - 1) \bmod m_r$, $\beta_k = \alpha_k$, $k = 1, \ldots, r-1$.
(5) Let $s' = s|0$ where $|$ denotes the concatenation operation.
(6) Let $s'' = s|1$
(7) **if** $i < r - 2$ **then**
(8) Name the root of $T^{s'}_{h-i}$ as γ which is the left child of α
(9) Name the root of $T^{s''}_{h-i}$ as β which is the right child of α
(10) $i = i + 1$
(11) Call Node_label(β,i,s')
(12) Call Node_label(γ,i,s'')
(13) **else**
(14) Call Two_dim_embed(γ,$T^{s''}_{h-(r-2)}$)
(15) Call Two_dim_embed(β,$T^{s'}_{h-(r-2)}$)
(16) **end**
(17) **end** /* of Node_label */

Figure 3.8 *Procedure* Node_label *for recursive labelling of* T_h

is $2^{r-2}|T_{h-(r-2)}|$ which is identical to Eq. (3.4) that we have derived earlier. An example of the above algorithm is illustrated in Fig. 3.9.

From the structure of procedure Multidim_Tree_embed and Fig. 3.9, it follows that for $1 \leq i \leq r-2$, there are 2 nodes in dimension i corresponding to a hypercube since we effectively bifurcate the left and right subtrees on its two planes. The last two dimensions, namely $r-1$ and r hold the subtree $T_{h-(r-2)}$ and this structure corresponds to a 2-dimensional hypercycle whose node radix vector, $\vec{m} = [5\ 2^{x+1}]$ and connectivity vector, $\vec{\rho} = \left[2\ \left\lceil \frac{2^x-1}{x} \right\rceil \right]$ where r and x are some integer parameters. Stated alternatively, procedure Multidim_Tree_embed

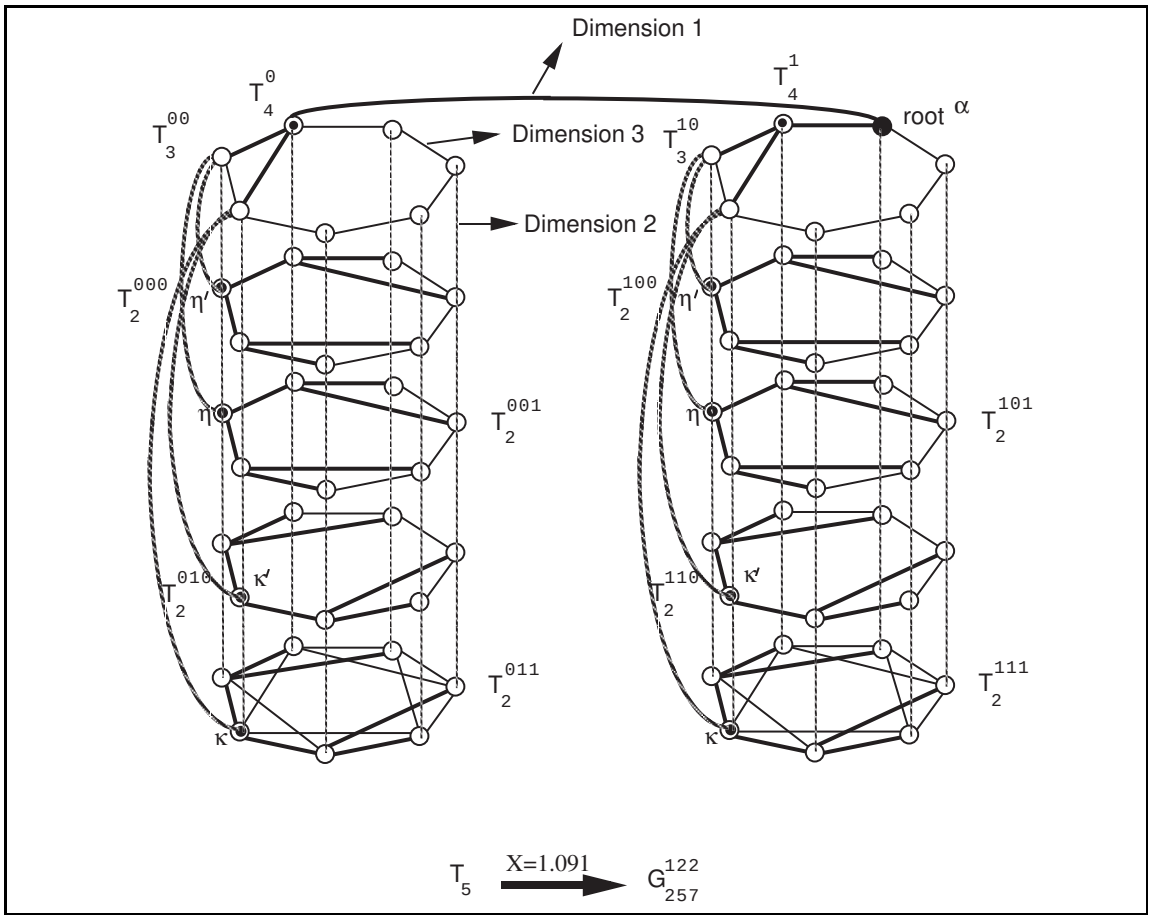


Figure 3.9 *An illustration of T_5 unit dilation embedding in G_{257}^{122} using Algorithm* Multi-dim_Tree_embed

establishes a hypercycle configuration that embeds T_h, $(h > 2)$ with unit dilation such that

$$m_i = \begin{cases} 2 & \text{if } 1 \leq i \leq r-2 \\ 5 & \text{if } i = r-1 \\ 2^{x+1}-1 & \text{if } i = r \end{cases}$$

$$\rho_i = \begin{cases} 1 & \text{if } 1 \leq i \leq r-2 \\ 2 & \text{if } i = r-1 \\ \left\lceil \frac{2^x-1}{x} \right\rceil & \text{if } i = r \end{cases}$$

Through numerical techniques, we shall compute the values for r and x that will realize an optimal embedding of T_h such that $|\mathcal{G}_{\vec{m}}^{\vec{\rho}}| \geq |T_h|$. Before proceeding further, we need to first establish the dependency of r and x to T_h. Without loss of generality, let us consider the path from the principal root to the extreme right leaf of T_h. The address labels of the right descendents β of T_h is given as the root of $T_{h-i}^{\overbrace{11\ldots11}^{i}}$ such that $\beta_r = \langle \alpha_r - i \rangle_{m_r}$, $\beta_j = \alpha_j$, $j = 1, \ldots, r-1$, for $i = 1, \ldots, r-3$. Hence, the number of addresses printed in the r^{th} dimension for the right descendents β of T_h including the principal root α is $r-2$. When $i = r-2$, β constitutes the root of the sub-tree $T_{h-(r-2)}^{\overbrace{11\ldots11}^{r-2}}$ which is embedded by procedure Two_dim_embed. In procedure Two_dim_embed we know that the right and left child of the root (Lines (4) and (8)) in Fig. 3.4) are also embedded in the r^{th} dimension. Thus the total number of nodes that need to be accommodated in the r^{th} dimension is given by $m_r = r - 2 + 3$ (where the number "3" accounts for the root and its two children in the fifth plane of the 2-dimensional case). But $m_r = 2^{x+1} - 1$ where x denotes the height of the tree that can be embedded in the r^{th} dimension. Equating the two quantities, we obtain

$$r_{max} = 2^{x+1} - 2 \tag{3.5}$$

where r_{max} denotes the maximum dimensions that can be used in the embedding of T_h for a given x and this factor will ultimately influence the size of the hypercycle.

The degree of $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ that will embed T_h is given by

$$\begin{aligned} d &= r - 2 + 4 + 2\left\lceil \frac{2^x - 1}{x} \right\rceil \\ &= r + 2 + 2\left\lceil \frac{2^x - 1}{x} \right\rceil \end{aligned} \tag{3.6}$$

where $r \leq r_{max}$.

The number of nodes, $M = |\mathcal{G}_{\vec{m}}^{\vec{\rho}}|$ is given by

$$M = 2^{r-2} \times 5(2^{x+1} - 1) \tag{3.7}$$

Let N_t be the number of node addresses generated by procedure Multidim_tree_embed such that

$$\begin{aligned} N_t &= 2^{r-2} - 1 + 2^{r-2}(3 + 4(2^{x+1} - 1)) \\ &= 2^{r-2} - 1 + 2^{r-2}(2^{x+1+2} - 1) \\ &= 2^{r-2} \times 2^{x+1+2} - 1 \\ &= 2^{x+r+1} - 1 \end{aligned} \tag{3.8}$$

Since each node address corresponds to a unique node in T_h, we obtain $|T_h| = N_t \Rightarrow 2^{h+1} - 1 = 2^{x+r+1} - 1 \Rightarrow h = x + r$. The expansion X of $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is given by the ratio

Table 3.2 *Optimal topologies for binary tree embedding in hypercycles*

Input		Optimal Hypercycle					Hypercube	
h	$\|T_h\|$	r^*	x^*	$M = 5(2^{r^*-2})(2^{x^*+1}-1)$	$X = \frac{M}{T_h}$	d	$X = \frac{\lceil T_{h+1} \rceil}{\lceil T_h \rceil + 1}$	d_h
3	15	2	1	15	1.0000	6	2.1333	5
4	31	2	2	35	1.1290	8	2.0645	6
5	63	3	2	70	1.0910	9	2.0317	7
6	127	4	2	140	1.1024	10	2.0157	8
7	255	5	2	280	1.0980	11	2.0078	9
8	511	6	2	560	1.0958	12	2.0039	10
9	1023	6	3	1200	1.1730	14	2.0020	11
10	2047	7	3	2400	1.1724	15	2.0010	12
11	4095	8	3	4800	1.1722	16	2.0005	13
12	8191	9	3	9600	1.1720	17	2.0002	14
13	16383	10	3	19200	1.1719	18	2.0001	15
14	32767	11	3	38400	1.1719	19	2.0001	16
15	65535	12	3	76800	1.1719	20	≈2.0000	17
16	131071	13	3	153600	1.1719	21	≈2.0000	18
17	262143	14	3	307200	1.1719	22	≈2.0000	19
18	524287	14	4	634880	1.2109	24	≈2.0000	20
19	1048575	15	4	1269760	1.2109	25	≈2.0000	21

$X = M/|T_h| = M/N_t$. Substituting Eqs. (3.7) and (3.8), we obtain

$$\begin{aligned} X &= \frac{5 \times 2^{r-2}(2^{x+1}-1)}{2^{x+r+1}-1} \\ &\leq \frac{5 \times 2^{r-2} \times 2^{x+1}}{2^{x+r+1}-1} \\ &= \frac{5 \times 2^{x+r-1}}{2^{x+r+1}} \times \left(1 - \frac{1}{2^{x+r+1}}\right)^{-1} = \frac{5}{4}\left(1 - \frac{1}{2^{h+1}}\right)^{-1} \\ &\approx \frac{5}{2^2} \qquad \text{for large values of } h \end{aligned}$$

or,

$$X \approx 1.25 \tag{3.9}$$

Since the hypercube is a popular structure with small degree and diameter, we use it as a metric for comparing the optimality of the designed hypercycle structure that can embed

a tree T_h. The asymptotic increase in the size of the hypercycle that will embed T_h with unit dilation is approximately 25 %. It is worth observing that a hypercube of dimension $h+2$ is required for embedding T_h with unit dilation. This entails a 100 % increase in the size of the hypercube. Let us now compare the degree of the two structures. Since $h = x+r$ from above, the unit dilation hypercube has $2^{h+2} = 2^{r+x+2}$ nodes and therefore its degree, d_h is given by $d_h = r + x + 2$. The difference, Δ, between the degrees of the hypercycle (Eq. (3.6)) and hypercube can be computed as follows:

$$\begin{aligned}\Delta &= r + 2 + 2\left\lceil \frac{2^x - 1}{x} \right\rceil - (r + x + 2) \\ &= 2\left\lceil \frac{2^x - 1}{x} \right\rceil - x \leq \left\lceil \frac{m_r}{2} \right\rceil - x\end{aligned}$$

We shall now consider optimization techniques that will produce a hypercycle with the smallest degree and size to embed T_h with unit dilation.

3.4 Optimization

We had shown earlier that procedure Multidim_Tree_embed embeds T_h with unit dilation on a hypercycle with an asymptotic expansion of 1.25. We need to choose the parameters x and r that will produce the smallest possible hypercycle to embed T_h. Stated alternatively, choose the hypercycle configuration such that $|T_h| \leq M$. This optimization problem can be formulated as the minimization of error using the least-squares approach given by:

$$\min_{w.r.t.\ \ r,x} F = (M - |T_h|)^2 \tag{3.10}$$

subject to the constraints

$$h_1 = M - 5(2^{r-2})(2^{x+1} - 1) = 0 \tag{3.11}$$

$$h_2 = h - (x + r) = 0 \quad \text{where } h = \log_2(|T_h| + 1) - 1 \tag{3.12}$$

$$g_1 = |T_h| - M \leq 0 \quad /^*\ M \text{ should be no less than } |T_h|\ ^*/ \tag{3.13}$$

$$g_2 = r - 2^{x+1} + 2 \leq 0 \quad /^*\ \text{from Eq. 3.5}\ ^*/ \tag{3.14}$$

$$g_3 = 2 - r \leq 0 \quad /^*\ \text{Minimum value of } r \text{ is 2}\ ^*/ \tag{3.15}$$

$$g_4 = -x \leq 0 \quad /^*\ x \text{ should not be negative}\ ^*/ \tag{3.16}$$

$$g_5 = 15 - |T_h| \leq 0 \tag{3.17}$$

where r and x are integer design variables. Constraint g_5 has been chosen on the assumption that the multidimensional embedding scheme will be effective for $|T_h| \geq 15$. When $|T_h| < 15$, algorithms Two_dim_embed or Tree_into_line can be used depending on r as shown in Fig. 3.7. The above problem represents a nonlinear, constrained, integer programming problem which

can be solved numerically [76]. The results of the optimization for $h \leq 19$ are summarized in Table 3.2. It is worth noting that the number of dimensions r grows exponentially on x (as given by Eq. 3.5) implying that for a small 1-dimensional tree embedding, a potentially larger tree could be accommodated with a small increase in degree.

3.4.1 Performance Comparison

Speed-up and efficiency are two standard parameters to judge the performance of a machine. In [42], Sarkar argues that these parameters do not adequately characterize a multiprocessor(multicomputer) architecture running a given algorithm since they do not take into account the cost of the system. It is evident that the cost of a parallel machine depends on the cost of the processing nodes and the communication links as well. He therefore introduced the concept of *cost-effectiveness*, a term which measures the performance/cost ratio of the system under consideration.

Cost-effectiveness is a metric that combines the architectural costs and speed-up (efficiency) of a multiprocessing system into one entity and is very useful for comparing the performance of different architectures executing a given parallel algorithm for some problem. Supposing an algorithm achieves the same speed-up in two different architectures. Then the obvious choice would be the cheaper machine and this is what cost-effectiveness quantifies. The higher the cost effectiveness the better is the "overall" performance of the system under consideration.

The cost-effectiveness of a multiprocessing system s with p processors is given by

$$CE_s(p) = \frac{1}{p + ce(p)} S_p^s \tag{3.18}$$

where c is the ratio of the cost of a communication link and processor node while $e(p)$ represents the number of edges in the system. The term S_p^s denotes the speed-up achieved by the multiprocessing system s for a given parallel algorithm. For a given speed-up, S_p^s, the cost-effectiveness is dictated by the denominator term in Eq.(3.18) which can be used as a metric for comparison costs. For the two systems of interest, the hypercube and hypercycle, we can compare the costs for unit dilation tree embeddings with identical speed-ups.

$$Cost_{hypercube}(p) = p + \frac{cp \log_2 p}{2} \qquad \text{where } p = 2^{h+2} \tag{3.19}$$

$$Cost_{hypercycle}(p') = p' + \frac{cp'd}{2} \tag{3.20}$$

where d is the degree of the optimized hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ that embeds T_h such that $|\mathcal{G}_{\vec{m}}^{\vec{\rho}}| = p'$. Fig. 3.10 gives a comparison of the cost for unit dilation, unit load factor embeddings of T_h for the hypercube and hypercycle structures when $c = 0.1$ and $c = 1$ as an example. It

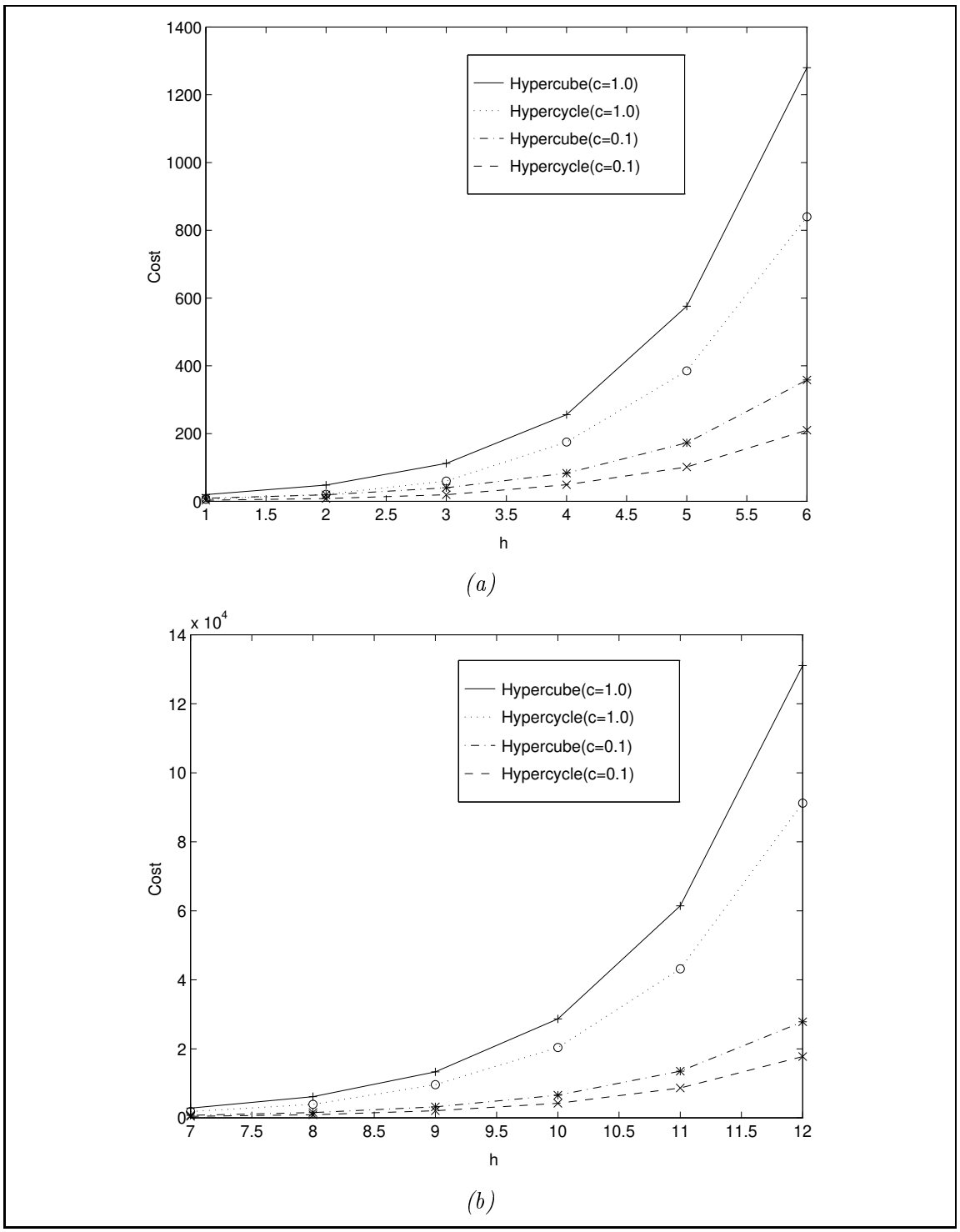


Figure 3.10 *Plot of the cost of hypercube and hypercycle structures for unit dilation tree embeddings for (a)* $h \leq 6$ *and (b)* $7 \leq h \leq 12$.

can be observed from the plot that hypercycles have a lower cost compared to hypercubes. For $h \leq 34$ (corresponding to $r_{max} = 30$, refer Eq. 3.5)), it has been numerically verified that the cost of hypercycles is lower than that of the hypercubes for the above values of c. The analytical proof is difficult due to the discrete nature of the problem and optimization involved.

3.5 Conclusion

The important characteristics of hypercycles, namely, variable connectivity, simple routing, incremental expandability, fault tolerance and ease of implementation make them attractive to simulate a variety of computational structures. In this Chapter, we presented a technique for embedding a complete binary tree in a hypercycle based multicomputer system. with unit dilation. The proposed scheme minimizes the degree by mapping T_h on a hypercube structure for $r-2$ dimensions and the remaining sub-trees are embedded in a 2-dimensional hypercycle that uses an optimal algorithm for tree bandwidth labelling on a line. By running numerical optimization, the optimal hypercycle embedding T_h is identified. This method yields a network with an asymptotic expansion of 1.25 with better cost-effectiveness compared to unit dilation embeddings of T_h in hypercubes where the expansion is 2.00. It is our understanding that, for dilation 2 embeddings of T_h, the hypercube Q_{h+1} is more cost-efficient with $X = |T_{h+1}|/|T_h| \approx 1$. The synthesis of other hypercycle structures which are not hypercubes will invariably result in a smaller cost-effectiveness.

As an extension of the work done in this Chapter, if bounds on the structural parameters of the hypercycle, namely r, $\vec{m}$ and $\vec{\rho}$ for unit dilation tree embeddings could be derived, then the optimality of the proposed multidimensional algorithm could be ascertained. Another interesting problem to pursue further is finding efficient algorithms for embedding arbitrary binary trees in hypercycles. The embedding of other types of communication structures is an open problem as well.

Chapter 4

Hamiltonian cycles in Hypercycles

4.1 Introduction

Embedding Hamiltonian cycles in a parallel computer interconnection network has direct applications in indexing schemes for sorting, algorithms for signal processing (matrix operations) [77], processor allocation/scheduling, embeddings, load balancing, fault-tolerance schemes [78] apart from graph theoretical interests. A graph is *Hamiltonian* if it contains a Hamiltonian cycle and this property is called as *Hamiltonicity*. In this Chapter, we formulate a generalized m-ary gray code algorithm for embedding a Hamiltonian cycle in a given hypercycle and also discuss some properties on Hamiltonian decomposition. Section 4.2 describes the history of Hamiltonian cycles briefly and the Hamiltonian properties in highly symmetric graphs. In Section 4.3, a combinatorial algorithm for embedding a Hamiltonian cycle in any arbitrary hypercycle is proposed and its proof is discussed at length. Section 4.4 is concerned with ranking and unranking [78] of Hamiltonian cycles. Finally, Sections 4.5 and 4.6 conclude the discussion with a brief discussion on Hamiltonian decomposition.

4.2 Problem History and Motivation

The history of Hamilton cycles dates back to the 19^{th} century when the British mathematician W.R. Hamilton investigated graphs that have cycles in which each node is visited exactly once. The property of Hamiltonicity in graphs [79] has become an area of active research with wide applications in Operations Research (e.g., Traveling Salesman problem), and computer network design. In [80], Gould gives an excellent survey of the general Hamiltonian cycle problem. The question of whether or not a given arbitrary graph is Hamiltonian is a classical $\mathcal{NP}$-complete problem [81]. However, for certain regular graphs such as the hypercube[82, 83], it is not $\mathcal{NP}$-complete and algorithms have been found that produce a

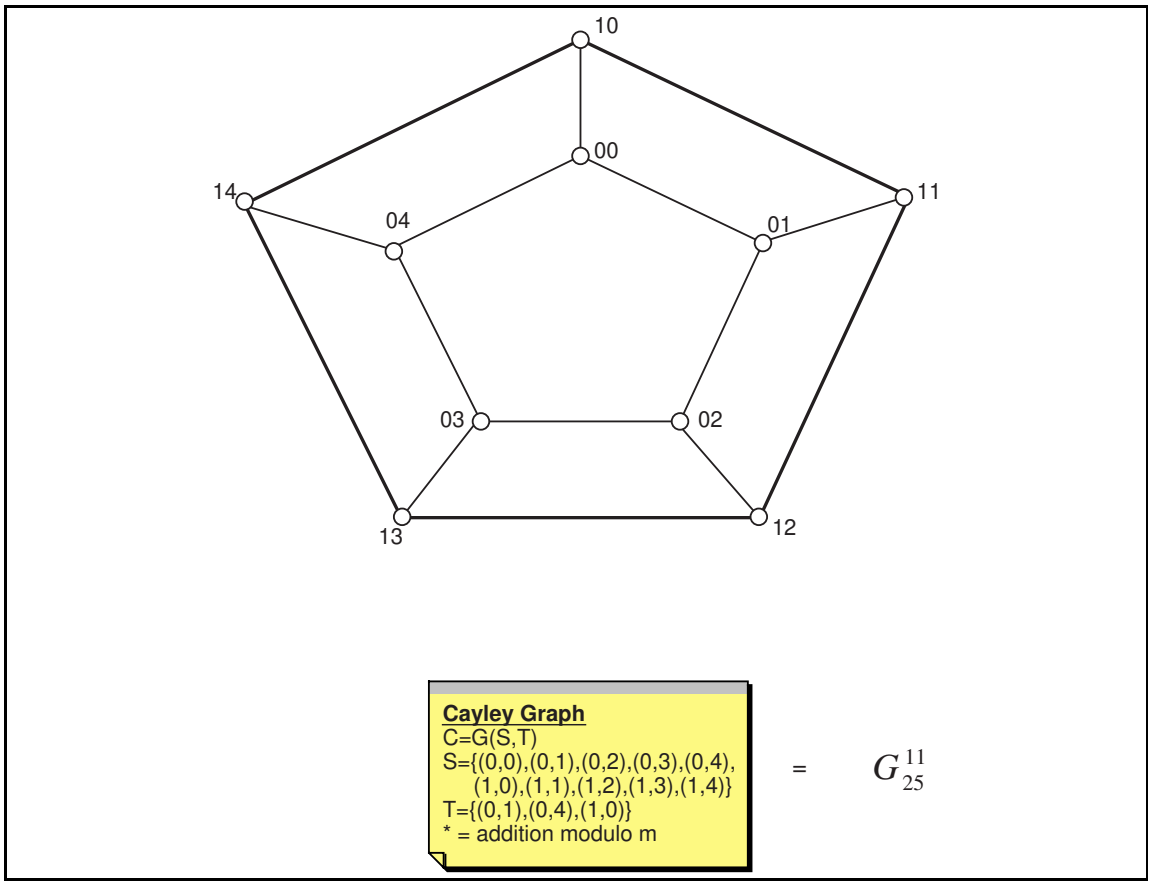


Figure 4.1 *Cayley graph C=G(S,T) isomorphic to* $\mathcal{G}_{25}^{11}$

Hamiltonian cycle. For instance, the *gray* code [84, 85] is an example of such an algorithm for the hypercubes.

In 1969, Lovász conjectured that every connected node symmetric graph has a Hamiltonian path [86]-[89]. Up to now, there are only four known non-trivial connected node transitive graphs that do not have Hamiltonian cycles yet possess Hamiltonian paths. These graphs are not Cayley graphs (e.g., Petersen's graph [90]). This inspired many people to conjecture that every connected Cayley graph with more than two nodes is Hamiltonian. Witte [91] gives a comprehensive collection of papers for the Hamiltonian cycle problem in Cayley graphs. We state the following theorems which are relevant to our analysis.

Theorem 4.1 *The Cayley graph* $C = G(S, T)$ *is node-transitive.*

The proof can be found in [92] (pp. 100).

Theorem 4.2 *Every connected Cayley graph* $C = G(S, T)$ *of an abelian group of order at least three is Hamiltonian.*

This is proved by Marušič in [93].

Lemma 4.1 *The 1-dimensional hypercycle $\mathcal{G}_{m_1}^{\rho_1}$ is a connected Cayley graph.*

Refer to Appendix D.

Lemma 4.2 *The cartesian product of two Cayley graphs is a Cayley graph.*

Proof. Let $C_1 = G(S_1, T_1)$ and $C_2 = G(S_2, T_2)$ be two Cayley graphs as defined before. Then $C = G(S, T)$ is a Cayley graph where $S = S_1 \times S_2$ and $T = \{(T_1 \times I), (I \times T_2)\}$. Indeed if $((a, b), (c, d)) \in E(C)$, then $(a, b)^{-1} * (c, d) \in T \Rightarrow (a^{-1}, b^{-1}) * (c, d) = (a^{-1} * c, b^{-1} * d) \in T$. For the edge $((a, b), (c, d)) \in E(C)$, if $a = c$ then $(0, b^{-1}, d) \in T \Rightarrow b^{-1} * d \in T_2$. Alternately, if $b = d$ then $(a^{-1} * c, 0) \in T \Rightarrow (a^{-1} * c) \in T_1$. ■

Lemma 4.3 *An r-dimensional hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is a connected Cayley graph.*

Proof. Consider an r-dimensional hypercycle defined as a cartesian product given by by Eq.(1.8). By Lemma 4.1, each of $\mathcal{G}_{m_i}^{\rho_i}$ for $(1 \leq i \leq r)$ is isomorphic to a Cayley graph $C_i = G(S_i, T_i)$ where

$$S_i = \{0, 1, \ldots m_i - 1\} \tag{4.1}$$

$$T_i = \cup\{x, m_i - x\} \quad \forall x \in \{1, \ldots, \rho_i)\} \tag{4.2}$$

From Lemma 4.2, it is obvious that the cartesian product of r Cayley graphs is a Cayley graph. Hence $C = G(S, T) = \prod_{i=1}^{r} C_i$ is a Cayley graph which is isomorphic to $\mathcal{G}_{m_1 m_2 \ldots m_r}^{\rho_1 \rho_2 \ldots \rho_r}$. An example of a Cayley graph isomorphic to hypercycle $\mathcal{G}_{52}^{11}$ is shown in Fig. 4.1. ■

Theorem 4.3 *The r-dimensional hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is Hamiltonian.*

Proof.

In Lemma 4.3, we have proved that $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is isomorphic to a connected Cayley graph $C = G(S, T) = \prod_{i=1}^{r} G(S_i, T_i)$ where S_i and T_i are defined by Eqs. (4.1) & (4.2) respectively. We know that $(S, *)$ is an abelian group since the cartesian product of abelian groups is abelian. In addition $|\mathcal{G}_{\vec{m}}^{\vec{\rho}}| \geq 2$ by definition†. Hence, from Theorem 4.2, the r-dimensional hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is *Hamiltonian.* ■

Since the Hamiltonicity of connected Cayley graphs was established and since we proved that hypercycles are Cayley graphs, we can conclude that hypercycles do contain at least

†It is trivial to show that $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ has a Hamiltonian path when $|\mathcal{G}_{\vec{m}}^{\vec{\rho}}| = 2$.

one Hamiltonian cycle. In the subsequent sections, we shall present several algorithms which can construct Hamiltonian cycles on arbitrary hypercycles.

Definition 4.1 *A hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is said to be* **uniform** *if $m_1 = m_2 = \ldots = m_r$.*

Definition 4.2 *A hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is said to be* **even** *if $\exists m_i$, $1 \leq i \leq r$ such that m_i is even.*

Definition 4.3 *A hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is said to be* **odd** *if $\forall i, 1 \leq i \leq r$ m_i is odd.*

4.3 Hamiltonian Cycle Algorithm

In this section, we consider combinatorial techniques to enumerate a Hamiltonian cycle in a hypercycle. Though a great deal of work has been done in binary reflected gray codes, the literature seems to be relatively silent concerning the generation of m-ary reflected Gray codes [94, 95]. In [96], a recursive formula is given for generating an m-ary Gray code sequence which in effect constitutes a Hamiltonian cycle. Stanton [97, pp. 15] also proposes a similar combinatorial algorithm for the above problem. The main drawback of the above schemes is that they work only for specific cases and are inapplicable to general hypercycles. For instance, the methods proposed in [94, 96] can be applied only to a uniform hypercycle while the algorithm given by Stanton holds good for the uniform and even hypercycles. None of these schemes generate a Hamiltonian cycle for an odd, non-uniform hypercycle.

In this section, we present an algorithm for embedding a Hamiltonian cycle in any arbitrary hypercycle which encompasses the uniform, even and odd categories. Our algorithm is a generalization of the m-ary Gray code enumeration algorithm [96] and is based on *node traversal* along each dimension of a hypercycle $\mathcal{G}_{m_1 m_2 \ldots m_r}^{1\ 1\ \ldots\ 1}$ since such a graph is an embedding in $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$. We define an implicit ordering in the dimensions of a hypercycle such that dimension 1 and dimension r have the lowest and highest precedence respectively. Using the prioritized dimension scheme, the successive node addresses are incremented by unity starting from the highest dimension. This generates a Hamiltonian cycle for the *uniform* category similar to that discussed in [96]. We shall prove by suitable algebraic manipulation, that the generation technique for the uniform hypercycle can be adapted for the *even* category and certain special configurations of hypercycles as well. For the odd category, the enumeration is performed by sectioning the hypercycle into two partitions and generating the Hamiltonian paths for the respective partitions. Before proceeding further to the various cases, we define some of the variables used in our algorithm. The starting address vector for the Hamiltonian cycle is initialized to $x = (s_1, s_2, \ldots, s_r)$[‡] An r-dimensional variable *sign*

[‡]All node addresses are evaluated modulo m_i, $i = 1, \ldots, r$.

```
(1)  Procedure Uniform()
(2)      begin
(3)          sign_i = 1, i = 1, ..., r /* Initialize direction */
(4)          x_i = (x_i - sign_i) mod m_i, i = 1, ..., r
(5)          Path_Uniform(1) /* Compute Hamiltonian Path */
(6)      end /* of procedure Uniform */
```

Figure 4.2 *Procedure Uniform for computing a Hamiltonian cycle in* $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$. *Assume that variables* x *and* $sign$ *are globally defined*

```
(1)      Procedure Path_Uniform(i)
(2)          begin
(3)              Repeat m_i times
(4)                  x_i = (x_i + sign_i) mod m_i
(5)                  if i ≠ r call Path_Uniform(i + 1)
(6)                  else print x_j (j = 1, ..., r)
(7)              end repeat
(8)              x_i = (x_i - sign_i) mod m_i
(9)          end /* of procedure Path_Uniform */
```

Figure 4.3 *Procedure Path_Uniform for computing a Hamiltonian path in* $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$

is introduced to monitor the direction of traversal of the path such that $sign_i$ can assume values 1 or -1 for dimension i, $i = 1, \ldots, r$.

4.3.1 Uniform Hypercycle

Let $m = m_1 = m_2 = \ldots\ldots = m_r$ be the node radices of a uniform hypercycle. The procedure for computing a Hamiltonian cycle for the uniform case, is shown in Fig. 4.2. The direction vector $sign$ is initialized in Line (4) and the starting address x is decremented in Line (4) to facilitate the generation of the Hamiltonian path. An example of a Hamiltonian cycle produced by Procedure Uniform is given in Fig. 4.4.

Lemma 4.4 *Procedure Path_Uniform prints* $M = \prod_{i=1}^{r} m_i$ *nodes in* $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ *where* $\vec{\rho} = [1 \ \ldots \ 1]$.

Proof. By induction. Assume $r = 1$. When procedure Path_Uniform is invoked initially with the command Path_Uniform(1), it will sequence through Lines (3)-(7) printing m_1 node addresses. When $r = 2$, there will be m_1 recursive calls to Path_Uniform(2) at Line (5), and Path_Uniform(2) will then iterate m_2 times. Hence the total number of node addresses printed is $M = m_1 m_2$. Let procedure Path_Uniform$(i+1)$ traverse through $\prod_{j=i+1}^{r} m_j$ nodes. When a call to Path_Uniform(i) is made, Path_Uniform$(i+1)$ will be invoked m_i times and the total number of addresses printed is $m_i \prod_{j=i+1}^{r} m_j = \prod_{j=i}^{r} m_j$. Since initially $i = 1$, we have $M = \prod_{j=1}^{r} m_j = |\mathcal{G}_m^{\rho}|$. ■

Lemma 4.5 *Each node traversed by procedure Path_Uniform is unique*

Proof. From the algorithm, it can be observed that each printed digit in the j^{th} dimension $(1 \leq j \leq r)$ appears exactly once and the j^{th} digit repeats only when there is a change in the i^{th} dimension for some $i < j$. Therefore no r-dimensional address vector repeats itself during the execution of Path_Uniform. Also one can calculate the repetition frequency of the digits of a given starting node s. From the algorithm, t $(1 \leq t \leq r)$ least significant digits of s will repeat after every κ steps where

$$\begin{aligned} \kappa &= 1 + m + m^2 + \ldots m^t \\ &= \frac{m^{t+1} - 1}{m - 1} \end{aligned} \tag{4.3}$$

For r digits, the number of steps required is given by $\frac{m^{r+1}-1}{m-1} > m^r$ (from Lemma 4.4). ■

Theorem 4.4 *Procedure Uniform generates a Hamiltonian cycle in a uniform hypercycle* $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$.

Proof. From Lemmas 4.4, 4.5 and procedure Path_Uniform, we can infer that a Hamiltonian path is traversed since each successive address generated differs from the preceding address in just one dimension and then by one (See Lines (4) and (8) of Fig. (4.3). There will be m^{i-1} invocations of Path_Uniform(i) for $1 \leq i \leq r$. Initially, the i^{th} digit of the starting address is set as $(s_i - sign_i) \bmod m$ due to Line (5) in Fig. 4.2. The i^{th} digit will attain the value $\langle s_i - sign_i + (m - sign_i) \rangle_m$ when procedure Path_Uniform(i) is exited at Line (9) of Fig. 4.3. Since Path_Uniform(i) is called m^{i-1} times, the i^{th} digit will be incremented by a factor of $m^{i-1}(m - sign_i)$. However, this quantity has to be offset by $sign_i$ when the last node of the Hamiltonian path is printed since Line (8) has no effect on the final printed address. Therefore with an initial value of s_i, after m^{i-1} invocations of

Path_Uniform(i), the i^{th} digit will be printed with the value given by

$$\hat{x}_i = \langle s_i - sign_i + m^{i-1}(m - sign_i) + sign_i \rangle_m \tag{4.4}$$

It then follows from Eq.(4.4) that

$$\hat{x}_i = \begin{cases} s_i & \text{if } 2 \leq i \leq r \\ s_1 - sign_1 & \text{if } i = 1 \end{cases}$$

which is adjacent to the starting node $x = (s_1, s_2, \ldots, s_r)$ and hence the path constitutes a Hamiltonian cycle. Fig. 4.4 gives a pictorial view of a Hamiltonian cycle in $\mathcal{G}_{44}^{11}$ enumerated by procedure Uniform.

■

4.3.2 Even Hypercycle

Procedure Even illustrated in Fig. 4.5 is invoked when the radices m_i are unequal $\forall i = 1, \ldots, r$ and there is at least one m_i that is *even*. Without loss of generality, assume that the highest radix contains an even number of nodes (i.e., m_1 is even) while the other radices $m_2, \ldots m_r$ may be even or odd.

The rationale behind case II is to generate a Hamiltonian cycle H by the concatenation of two paths P_1 and P_2 such that $|P_1| = |P_2| = (m_1/2)\prod_{j=2}^{r} m_j = M/2$. Since m_1 is even, paths P_1 and P_2 can be rendered by exploiting the symmetry of the hypercycle in dimension 1 and which in effect, manifest as reflected images of each other. A new r-dimensional variable, *count* is introduced to facilitate the generation of node addresses in P_1 and P_2.

Proof of Correctness:

Theorem 4.5 *Procedure Even generates a Hamiltonian cycle in even hypercycle.*

Proof. The proof follows the same argument as given in Theorem 4.4. We shall generate two paths P_1 and P_2 whose concatenation will constitute a Hamiltonian cycle. As in procedure Uniform, the direction vector *sign* is initialized in Line (3) of Fig. 4.5.

Path 1: Given a starting node $x = (s_1, s_2 \ldots, s_r)$ and direction vector $sign_i = sign_i^{0\,\S}$ $(i = 1, \ldots, r)$ for path P_1, the node address x is decremented by unity in Line (4) of Fig. 4.5 for proper initialization and procedure Path_Even(1) is called at Line (5). Essentially, Path_Even is identical to Path_Uniform except for the addition of Lines (3) & (4) which are

§The superscript of $sign_i^0$ denotes the initial assignment.

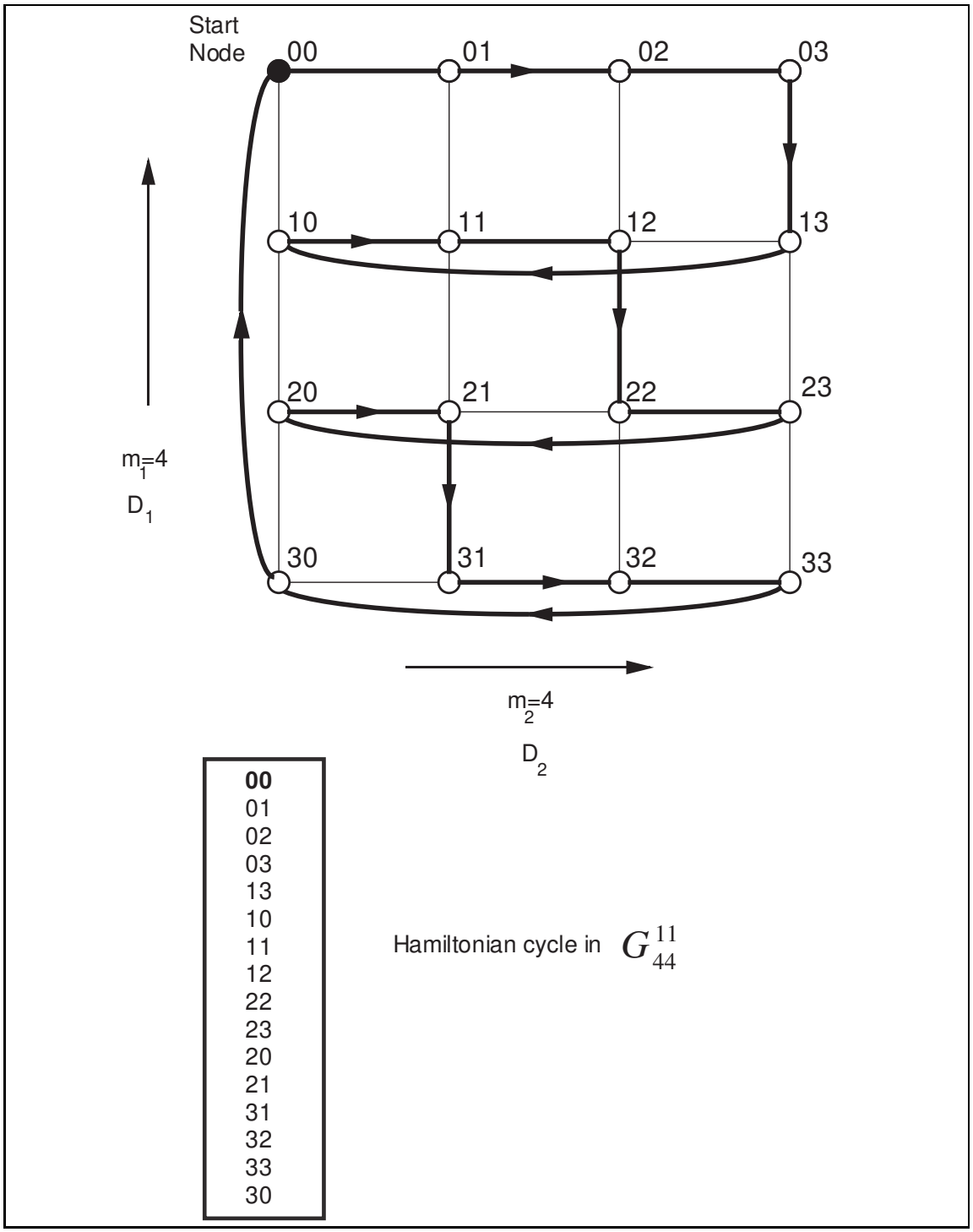


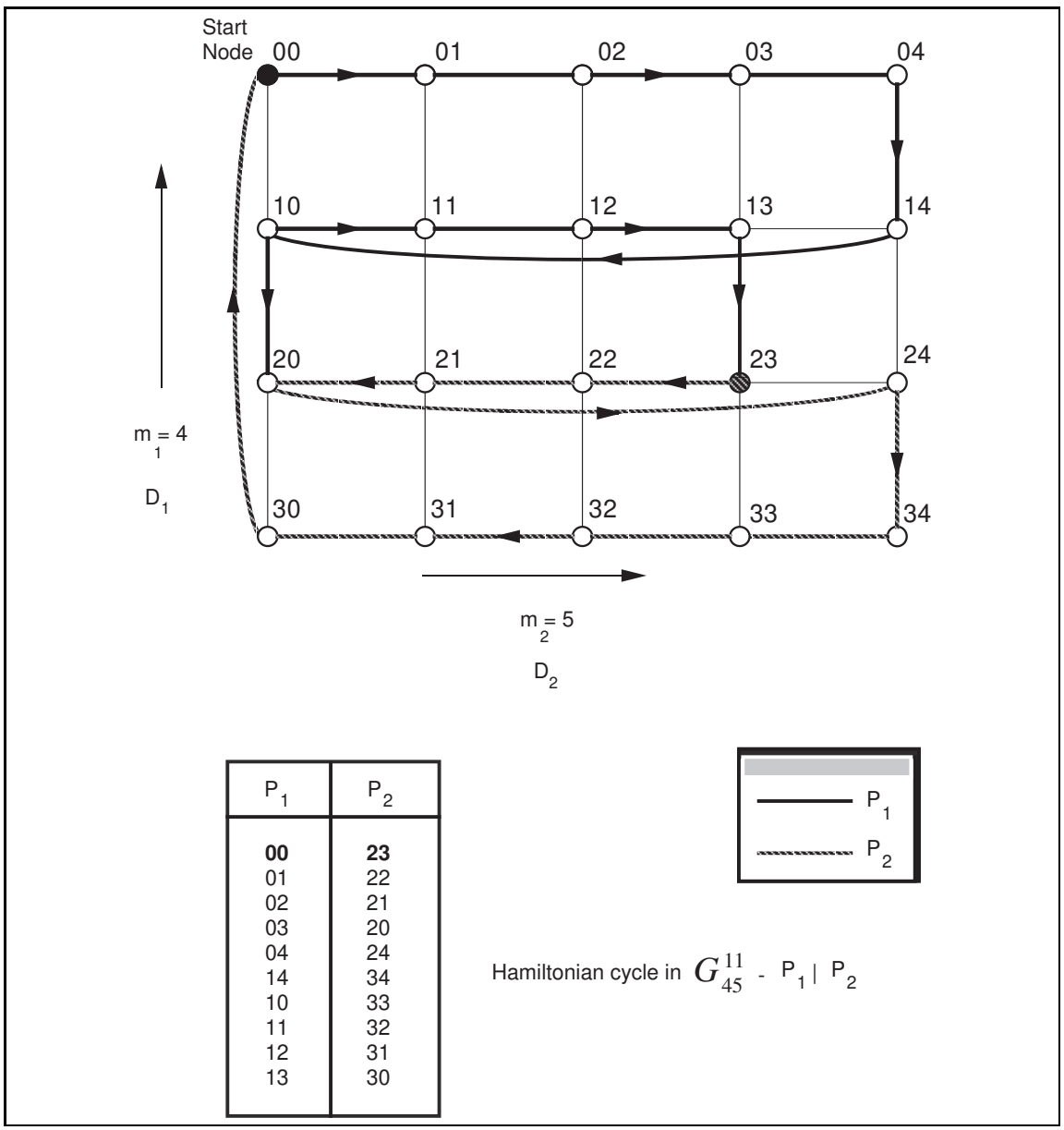
Figure 4.4 *An example of a Hamiltonian Cycle traced in hypercycle* $\mathcal{G}_{44}^{11}$ *for the Uniform category with node (0,0) as the starting point*

```
(1)  Procedure Even()
(2)      begin
(3)          sign_i = 1, i = 1,...,r          /* Initialize direction of traversal */
(4)          x_i = (x_i - sign_i) mod m_i, i = 1,...,r
(5)          Path_Even(1)                     /* Compute Path P_1 */
(6)          x_i = (x_i + 2sign_i) mod m_i, i = 2,...,r
(7)          x_1 = (x_1 + sign_i) mod m_1
(8)          sign_i = -sign_i, i = 2,...,r /* Reverse direction of traversal */
(9)          Path_Even(1)                     /* Compute Path P_2 */
(10)     end /* of Procedure Even */
```

Figure 4.5 *Procedure Even for computing a Hamiltonian cycle for an even hypercycle*

```
(1)      Procedure Path_Even(i)
(2)          begin
(3)              if i = 1 count_i = m_i/2
(4)              else count_i = m_i
(5)              repeat count_i times
(6)                  x_i = (x_i + sign_i) mod m_i
(7)                  if i ≠ r call Path_Even(i + 1)
(8)                  else print x_j (j = 1,...,r)
(9)              end repeat
(10)             x_i = (x_i - sign_i) mod m_i
(11)         end /* of procedure Path_Even */
```

Figure 4.6 *Procedure Path_Even for computing a Hamiltonian path in* $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$



P_1	P_2
00	**23**
01	22
02	21
03	20
04	24
14	34
10	33
11	32
12	31
13	30

Figure 4.7 *An example of a Hamiltonian Cycle traced in hypercycle $\mathcal{G}_{45}^{11}$ for the Even category with node (0,0) as the starting point*

required for tracking the number of nodes traversed in P_1 and P_2. The last address vector printed in Line (8) of Fig. 4.6 is given by

$$\hat{x}_i = \begin{cases} \left\langle s_i - sign_i^0 + \frac{\prod_{j=1}^{i-1} m_j}{2} sign_i^0 \, (m_i - 1) + sign_i^0 \right\rangle_{m_i} = \left\langle s_i - \frac{\prod_{j=1}^{i-1} m_j}{2} \right\rangle_{m_i} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{if } 2 \leq i \leq r \\ \langle s_1 - sign_1^0 + sign_1^0 \, (\frac{m_1}{2}) \rangle_{m_1} \qquad\qquad\qquad\qquad \text{if } i = 1 \end{cases}$$

On exiting procedure Path_Even at Line (11) of Fig. 4.6, the computed address vector x is represented as:

$$x_i = \begin{cases} \langle s_i - sign_i^0 \left(\frac{\prod_{j=1}^{i-1} m_j}{2} \right) - sign_i^0 \rangle_{m_i} \qquad \text{if } 2 \leq i \leq r \\ \langle s_1 - sign_1^0 + sign_1^0 \, (\frac{m_1}{2}) - sign_1^0 \rangle_{m_1} \qquad \text{if } i = 1 \end{cases}$$

Path 2:

Before P_2 can be generated, proper initialization of the address vector x, as shown in Lines (6) and (7) of Fig. 4.5 is required in order to obtain the correct starting point for P_2. Thus, at the end of Line (8) of procedure Path_Even at Line (9), we have

$$x_i = \begin{cases} \langle s_i - sign_i^0 \frac{\prod_{j=1}^{i-1} m_j}{2} + sign_i^0 \rangle_{m_i} \qquad \text{if } 2 \leq i \leq r \\ \langle s_1 - sign_1^0 + sign_1^0 \, (\frac{m_1}{2}) \rangle_{m_1} \qquad \text{if } i = 1 \end{cases}$$

The direction vector, *sign* is negated in Line (9) in all dimensions in order to retrace the traversed path P_1 except in dimension 1. This will ensure that the desired starting point in P_2 which is printed in Line (7) of Fig. 4.6 is given by

$$\hat{x}_i = \begin{cases} \langle s_i - sign_i^0 \frac{\prod_{j=1}^{i-1} m_j}{2} \rangle_{m_i} \qquad \text{if } 2 \leq i \leq r \\ \langle s_1 + sign_1^0 \, (\frac{m_1}{2}) \rangle_{m_1} \qquad \text{if } i = 1 \end{cases} \tag{4.5}$$

which is adjacent to the last printed address of P_1 given by Eq. (4.5). The final printed address of P_2 on exiting Path_Even is given by

$$\hat{x}_i = \begin{cases} \left\langle \left(s_i - sign_i^0 \left(\frac{\prod_{j=1}^{i-1} m_j}{2} \right) + sign_i^0 \right) - sign_i^0 \left(\frac{\prod_{j=1}^{i-1} m_j}{2} \right) (m_i - 1) - sign_i^0 \right\rangle_{m_i} = \langle s_i \rangle_{m_i} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{if } 2 \leq i \leq r \\ \left\langle \left(s_1 - sign_1^0 + sign_1^0 \left(\frac{m_1}{2} \right) \right) + sign_1^0 \frac{m_1}{2} - sign_1^0 + sign_1^0 \right\rangle_{m_1} = \left\langle s_1 - sign_1^0 \right\rangle_{m_1} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{if } i = 1 \end{cases}$$

```
(1)   Procedure Path_Odd(i)
(2)       begin
(3)           Repeat count_i times
(4)               x_i = (x_i + sign_i) mod m_i
(5)               if i ≠ r call Path_Odd(i + 1)
(6)               else print x_j (j = 1, ..., r)
(7)           end repeat
(8)           sign_i = −sign_i
(9)           x_i = (x_i − sign_i) mod m_i
(10)      end /* of procedure Path_Odd */
```

Figure 4.8 *Procedure Path_Odd for computing a Hamiltonian path in* $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$

which is adjacent to the initial starting address $x = (s_1, s_2, \ldots, s_r)$. Hence, it is obvious that $H = P_1|P_2$ constitutes a Hamiltonian cycle where | denotes the concatenation operation. ■

The Hamiltonian cycle generated by procedure Even can be considered as a generalized, m-ary, partially reflected gray code with the reflection occurring at the highest dimension and is different from the enumeration algorithm given in [97, pp.15]. Fig. 4.7 depicts the Hamiltonian cycle traced by procedure Even for a 2-dimensional hypercycle $\mathcal{G}_{45}^{11}$. In general, for procedure Even to generate a Hamiltonian cycle in $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$, we determine the first radix m_p $(1 \leq p \leq r)$ that is even and swap it with m_1. Since hypercycles are symmetrical graphs that are *node transitive*, the new hypercycle obtained as a result of swapping is isomorphic to the original one. The node addresses that have been altered as a result can be restored to the original graph configuration by re-swapping the appropriate digits.

4.3.3 Odd Hypercycle

When the number of nodes in $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is odd, given a starting node x, the Hamiltonian cycle can be computed by decomposing the original graph into two graphs $\mathcal{G}'$ and $\mathcal{G}''$ in any dimension i, $i = 1, \ldots, r$. By generating a Hamiltonian path in each of $\mathcal{G}'$ and $\mathcal{G}''$, we can show that a Hamiltonian cycle can be obtained in $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$.

We present procedure Path_Odd in Fig. 4.8 which is similar to Path_Even except for the addition of Line (8) in which the direction of traversal is reversed in each dimension.

This will essentially create the effect of *hairpin bend* in the enumerated path and as will be seen, the address vector in each dimension will vary between the extremities $\langle s_i \rangle_{m_i}$ and $\langle s_i - sign_i^0 \rangle_{m_i}$ alternately.

Lemma 4.6 *Let* $x = (\langle s_1 - sign_1^0 \rangle_{m_1}, \langle s_2 - sign_2^0 \rangle_{m_2}, \ldots, \langle s_r - sign_r^0 \rangle_{m_r})$, $count_i = m_i$ *and* $sign_i^0 = 1(-1)$, $\forall i \in \{1, \ldots, r\}$ *be the assigned values before invoking procedure Path_Odd(1). If* $\prod_{i=1}^{r} m_i$ *is odd, then procedure Path_Odd just before termination will print address* $(\langle s_1 - sign_1^0 \rangle_{m_1}, \langle s_2 - sign_2^0 \rangle_{m_2}, \ldots, \langle s_r - sign_r^0 \rangle_{m_r})$

Proof. When Path_Odd(i) is invoked for the first time, the value attained by x_i at the end of Line (7) is given by $\langle s_i - sign_i^0 + m_i sign_i^0 \rangle_{m_i} = \langle s_i - sign_i^0 \rangle_{m_i}$. Given that $\prod_{i=1}^{r} m_i$ is odd, there will be $\left\lceil \frac{\prod_{j=1}^{i-1} m_j}{2} \right\rceil$ invocations of Path_Odd(i) with $sign_i = sign_i^0$ and $\left\lfloor \frac{\prod_{j=1}^{i-1} m_j}{2} \right\rfloor$ invocations with $sign_i = -sign_i^0$ since $sign_i$ is negated in Line (8) during each invocation. Thus, due to Line (9), the address vector will be offset by a factor of $\left(\left\lceil \frac{\prod_{j=1}^{i-1} m_j}{2} \right\rceil - \left\lfloor \frac{\prod_{j=1}^{i-1} m_j}{2} \right\rfloor \right) sign_i^0$. Therefore, the i^{th} printed digit of the address vector x is given by

$$\begin{aligned} \hat{x}_i &= \left\langle s_i - sign_i^0 + \left(\left\lceil \frac{\prod_{j=1}^{i-1} m_j}{2} \right\rceil m_i - \left\lfloor \frac{\prod_{j=1}^{i-1} m_j}{2} \right\rfloor m_i \right) sign_i^0 \right. \\ &\quad \left. + \left(\left\lceil \frac{\prod_{j=1}^{i-1} m_j}{2} \right\rceil - \left\lfloor \frac{\prod_{j=1}^{i-1} m_j}{2} \right\rfloor - 1 \right) sign_i^0 \right\rangle_{m_i} \qquad (4.6) \\ &= \left\langle s_i - sign_i^0 + m_i sign_i^0 \right\rangle_{m_i} \\ &= \langle s_i - sign_i^0 \rangle_{m_i} \qquad (4.7) \end{aligned}$$

The "-1" term in Eq. (4.6) is due to the fact that for $\prod_{j=1}^{i-1} m_j$ being odd, the i^{th} address digit is printed for the last time before its corresponding variable is incremented by $sign_i^0$ as per Line (9). ■

Lemma 4.7 *Let* $\prod_{i=1}^{r-1} m_i$ *be odd and* $x = (s_1, s_2, \ldots, s_r)|_{m_1, m_2, \ldots, m_r}$ *be an address vector in* $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$. *If* $count_i = m_i$ *and* $sign_i = \{(-1)^{\prod_{j=1}^{i-1} m_j}\} sign_i^0 = -sign_i^0$, $\forall i \in \{1, \ldots, r-1\}$, $count_r = m_r - 1$ *and* $sign_r = sign_r^0$ *be the values of variables just before invoking Path_Odd(1), then address* $(s_1, s_2, \ldots, s_r - sign_r^0)$ *will be printed by procedure Path_Odd just before termination.*

Proof. Similar to Lemma 4.6, we can derive the printed address of the x_i^{th} digit such that

for $1 \leq i \leq r-1$, we have

$$\begin{aligned}\hat{x}_i &= \left\langle s_i - \left(\left\lceil \frac{\prod_{j=1}^{i-1} m_j}{2} \right\rceil m_i - \left\lfloor \frac{\prod_{j=1}^{i-1} m_j}{2} \right\rfloor m_i \right. \\ &\qquad \left. + \left\lceil \frac{\prod_{j=1}^{i-1} m_j}{2} \right\rceil - \left\lfloor \frac{\prod_{j=1}^{i-1} m_j}{2} \right\rfloor - 1 \right) sign_i^0 \Big\rangle_{m_i} \\ &= \left\langle s_i - m_i sign_i^0 \right\rangle_{m_i} = \langle s_i \rangle_{m_i} \end{aligned} \tag{4.8}$$

and

$$\begin{aligned}\hat{x}_r &= \left\langle s_r + \left(\left\lceil \frac{\prod_{j=1}^{i-1} m_j}{2} \right\rceil (m_r - 1) - \left\lfloor \frac{\prod_{j=1}^{r-1} m_j}{2} \right\rfloor (m_r - 1) \right. \\ &\qquad \left. + \left\lceil \frac{\prod_{j=1}^{r-1} m_j}{2} \right\rceil - \left\lfloor \frac{\prod_{j=1}^{r-1} m_j}{2} \right\rfloor - 1 \right) sign_r^0 \Big\rangle_{m_r} \\ &= \left\langle s_r + (m_r - 1) sign_r^0 \right\rangle_{m_r} = \langle s_r - sign_r^0 \rangle_{m_r} \end{aligned} \tag{4.9}$$

■

Lemma 4.8 *Each node traversed by procedure Path_Odd is unique when* $count_i = m_i$, $sign_i = 1(-1)$, $i = 1, \ldots, r$.

Proof. It can be seen from the structure of the algorithm given in Fig. 4.8 that starting from dimension 1 (of lowest precedence), each digit in the i^{th} dimension, $1 \leq i \leq r$ appears exactly once since the i^{th} digit is incremented $count_i$ times and and a digit in dimension j , $j > i$ repeats only when a digit in dimension i changes. The path traced by procedure Path_Odd is given by

$$\begin{aligned} P = \{ & (s_1, s_2, \ldots, s_{r-2}, s_{r-1}, s_r) \\ & \vdots \quad \vdots \quad \text{after } m_r - 1 \text{ steps} \\ & (s_1, s_2, \ldots, s_{r-2}, s_{r-1}, \langle s_r - sign_r \rangle_{m_r}) \\ & (s_1, s_2, \ldots, \langle s_{r-1} + sign_{r-1} \rangle_{m_{r-1}}, \langle s_r - sign_r \rangle_{m_r}) \\ & \vdots \quad \vdots \quad \text{after } 2m_r - 1 \text{ steps} \\ & (s_1, s_2, \ldots, \langle s_{r-1} + sign_{r-1} \rangle_{m_{r-1}}, s_r) \\ & \vdots \quad \vdots \quad \text{after } m_1 m_2 \ldots m_r \text{ steps} \\ & (\langle s_1 - sign_1 \rangle_{m_1}, \langle s_2 - sign_2 \rangle_{m_2}, \ldots, \langle s_{r-1} - sign_{r-1} \rangle_{m_{r-1}}, \langle s_r - sign_r \rangle_{m_r}) \} \end{aligned}$$

and the enumerated path alternates back and forth in each dimension like a *hair pin* bend between $\langle s_i \rangle_{m_i}$ and $\langle s_i - sign_i \rangle_{m_i}$.

■

Let $\mathcal{G}^{\rho_1\rho_2\cdots\rho_r}_{m_1m_2\ldots m_r}$ be an r-dimensional hypercycle such that $\prod_{i=1}^{r} m_i$ is odd. Then one can embed the graphs, $\mathcal{G}'$ and $\mathcal{G}''$ in $\mathcal{G}^{\rho}_{m}$ as follows:

1. $\mathcal{G}'^{\rho_1\rho_2\cdots\rho_{r-1}}_{m_1m_2\ldots m_{r-1}}$ – This is an (r-1) dimensional hypercycle
2. $\mathcal{G}'' = \mathcal{G}'^{\rho_1\rho_2\cdots\rho_{r-1}}_{m_1m_2\ldots m_{r-1}} \times P_{m_r-1}$ where P_{m_r-1} is a one dimensional mesh (path) such that $|P_{m_r-1}| = m_r - 1$. Let $\alpha = (\alpha_1, \alpha_2, \ldots \alpha_r)|_{m_1,m_2,\ldots,(m_r-1)}$ and $\beta = (\beta_1, \beta_2, \ldots \beta_r)|_{m_1,m_2,\ldots,(m_r-1)}$ be two nodes in $\mathcal{V}(\mathcal{G}'')$. Then $(\alpha, \beta) \in \mathcal{E}(\mathcal{G}'')$ iff the following two conditions are satisfied.
   (a) There exists some j, $1 \le j \le r-1$, for $\mathcal{G}'^{\rho_1\rho_2\cdots\rho_{r-1}}_{m_1m_2\ldots m_{r-1}}$ such that
   $$\begin{aligned} \beta_j &= (\alpha_j \pm \xi_j) \bmod m_j, \text{for } 1 \le \xi_j \le \rho_j \qquad (4.10)\\ \alpha_i &= \beta_i \quad \text{for } i \ne j, \forall i = 1, \ldots, r \end{aligned}$$
   (b) In the r^{th} dimension of $\mathcal{G}''$, the edges corresponding to the Path are given by
   $$\begin{aligned} \beta_r &= (\alpha_r \pm 1) \qquad (4.11)\\ (\alpha_r + 1) \bmod (m_r - 1) &\ne 0 \qquad (4.12)\\ (\alpha_r - 1) \bmod (m_r - 1) &\ne m_r - 2 \qquad (4.13)\\ \alpha_i &= \beta_i \quad \forall i = 1, \ldots, r-1 \qquad (4.14) \end{aligned}$$

Obviously, $\mathcal{G}''$ is a embedding of $\mathcal{G}^{\vec{\rho}}_{\vec{m}}$ since P_{m_r-1} and $\mathcal{G}'$ are subgraphs of $\mathcal{G}^{\vec{\rho}}_{\vec{m}}$ and it follows that $(\mathcal{G}' \times P_{m_r-1}) \subset \mathcal{G}^{\vec{\rho}}_{\vec{m}}$.

Mapping:

1. if $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{r-1}) \in \mathcal{G}' \leftrightarrow (\alpha_1, \alpha_2, \ldots \alpha_{r-1}, s_r) \in \mathcal{G}^{\vec{\rho}}_{\vec{m}}$
2. if $\beta = (\beta_1, \beta_2, \ldots \beta_r)|_{m_1,m_2,\ldots,(m_r-1)} \in \mathcal{G}'' \leftrightarrow (\beta_1, \beta_2, \ldots \beta_{r-1}, \gamma_r)|_{m_1,m_2,\ldots,m_r} \in \mathcal{G}^{\vec{\rho}}_{\vec{m}}$ such that $\gamma_r \in \{0, 1, \ldots, s_r - 1, s_r + 1, \ldots, m_r\}$ and $\gamma_r = \langle sign_r(\beta_r + 1) + s_r \rangle_{m_r}$ where $sign_r \in \{1, -1\}$.

An example of the mapping scheme is given in Fig. 4.9 for $\mathcal{G}^{11}_{35}$. Note that a starting node with value $s = 02$ is assumed here for illustration.

Theorem 4.6 *Procedure Odd given in Fig. 4.10 generates a Hamiltonian cycle in an odd hypercycle.*

Proof. A Hamiltonian cycle can be computed using procedure Odd by tracing the Hamiltonian paths H_1 and H_2 in graph embeddings $\mathcal{G}'$ and $\mathcal{G}''$ of $\mathcal{G}^{\vec{\rho}}_{\vec{m}}$ for an odd hypercycle as detailed below:

Path H_1 in $\mathcal{G}'$:

The variables *sign* and *count* are initialized as shown in Lines (3)-(5) of Fig. 4.10. Let $x = (s_1, s_2, \ldots, s_r)$ be the starting node which is decremented in Line (6) for proper

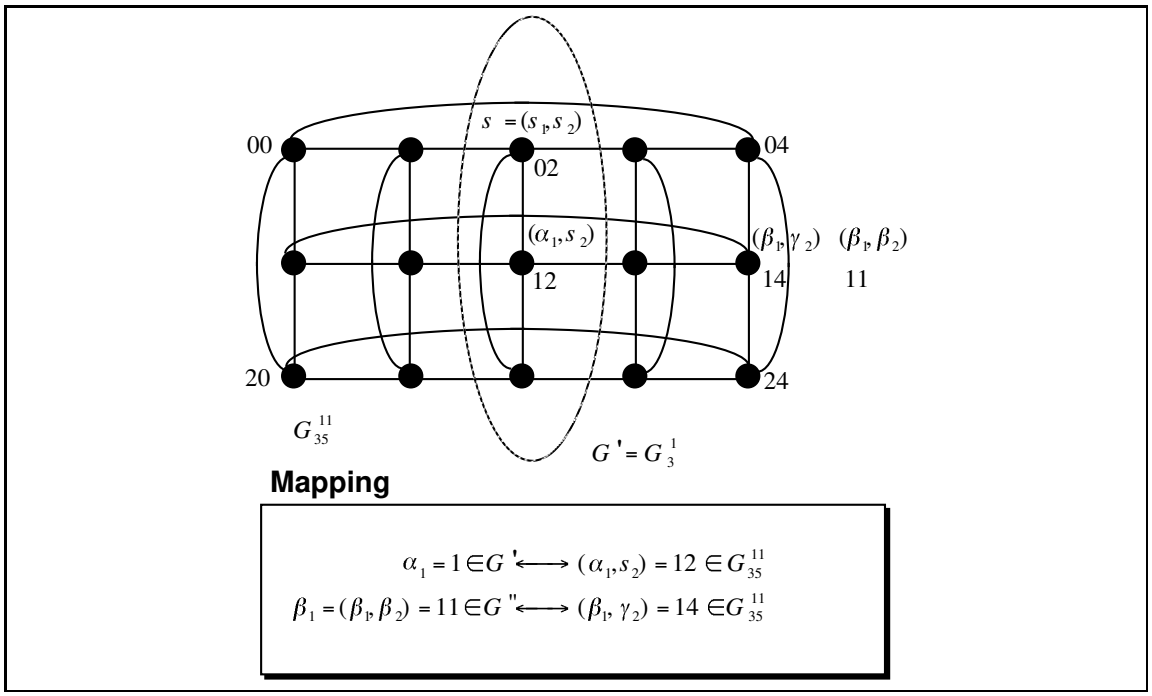


Figure 4.9 *Node mapping in* $\mathcal{G}_{35}^{11}$

enumeration of the Hamiltonian path by procedure Path_Odd in Line (7). Starting at node $x = (s_1, s_2, \ldots s_{r-1})$ of $\mathcal{G}'$ corresponding to $(s_1, s_2, \ldots, s_r) \in \mathcal{G}_{\vec{m}}^{\vec{\rho}}$, one traverses using procedure Path_Odd to arrive at node $x = (\langle s_1 - sign_1 \rangle_{m_1}, \langle s_2 - sign_2 \rangle_{m_2}, \ldots, \langle s_{r-1} - sign_{r-1} \rangle_{m_{r-1}}) = (\langle s_1 - 1 \rangle_{m_1}, \langle s_2 - 1 \rangle_{m_2}, \ldots, \langle s_{r-1} - 1 \rangle_{m_{r-1}}) \in \mathcal{G}'$ as per Lemma 4.6 which is equivalent to node $(\langle s_1 - 1 \rangle_{m_1}, \langle s_2 - 1 \rangle_{m_2}, \ldots, \langle s_{r-1} - 1 \rangle_{m_{r-1}}, s_r)$ of $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$. The last node of H_1 viz. $((\langle s_1 - 1 \rangle_{m_1}, \langle s_2 - 1 \rangle_{m_2}, \ldots, \langle s_{r-1} - 1 \rangle_{m_{r-1}}, \langle s_r \rangle_{m_r})$ is connected to $(\langle s_1 - 1 \rangle_{m_1}, \langle s_2 - 1 \rangle_{m_2}, \ldots, \langle s_{r-1} - 1 \rangle_{m_{r-1}}, \langle s_r + 1 \rangle_{m_r})$ of $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ or node $(\langle s_1 - 1 \rangle_{m_1}, \langle s_2 - 1 \rangle_{m_2}, \ldots, \langle s_{r-1} - 1 \rangle_{m_{r-1}}, \langle 0 \rangle_{m_r - 1})$ in $\mathcal{G}''$ which will eventually become the first printed address in H_2 of $\mathcal{G}''$.

Path H_2 in $\mathcal{G}''$:

Though $\mathcal{G}''$ is not a hypercycle, we shall shortly show that the address increment operations in procedure Path_Odd do not violate the edge definitions in $\mathcal{G}_m^{\rho}$. The variable *count* is set in Line (8) of Fig. 4.10 to properly sequence the Hamiltonian path H_2 in $\mathcal{G}''$. From procedure Path_Odd, we note that the i^{th} component of the direction vector viz, $sign_i$ is negated at Line (8) of Fig. 4.8 during each invocation of Path_Odd(i). Since there are $\prod_{j=1}^{i-1} m_j$ calls to procedure Path_Odd(i), the effective value of $sign_i$ on exit of the procedure is given by $sign_i = (-1)^{\prod_{j=1}^{i-1} m_j} sign_i^0$. Since $\prod_{j=1}^{r} m_i$ is odd $sign_i = -sign_i^0$, $i = 1, \ldots, r$. However, for

(1) **Procedure** Odd()
(2) **begin**
(3) $sign_i = 1,\ i = 1, \ldots, r$ /* Initialize direction and count vector */
(4) $count_i = m_i,\ i = 1, \ldots, r-1$ /* in $\mathcal{G}'$ */
(5) $count_r = 1$
(6) $x_i = (x_i - sign_i) \bmod m_i,\ i = 1, \ldots, r$
(7) Path_Odd(1) /* Compute Hamilton Path H_1 */
(8) $count_r = m_r - 1$
(9) $sign_r = -sign_r\ i = 1, \ldots, r-1$ /* Reverse direction in r^{th} dimension
(10) $x_i = s_i,\ i = 1, \ldots, r$ /* Initialize for proper starting point in H_2 */
(11) Path_Odd(1) /* Compute Hamilton Path H_2 */
(12) **end** /* of Procedure Odd */

Figure 4.10 *Procedure Odd for computing a Hamiltonian Cycle in an odd hypercycle*

the r^{th} dimension, $sign_r$ is negated again in Line (9) of Fig. 4.10 to restore the desired initial condition such that $sign_r = sign_r^0$. The address vector x is re-initialized in Line (10) such that $x_i = s_i$, $1 \leq i \leq r$ and on invoking procedure Path_Odd in Line (11), the first address printed in H_2 is given by $(\langle s_1 - sign_1\rangle_{m_1}, \ldots, \langle s_{r-1} - sign_{r-1}\rangle_{m_{r-1}} \langle s_r + sign_r\rangle_{m_r}) = (\langle s_1 - 1\rangle_{m_1}, \ldots, \langle s_{r-1} - 1\rangle_{m_{r-1}} \langle s_r + 1\rangle_{m_r})$ Applying Lemma 4.7, the final node reached in $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is given by $(s_1, \ldots s_{r-1}, \langle s_r - 1\rangle_{m_r})$ which is identical to $(s_1, \ldots, s_{r-1}, \langle m_r - 2\rangle_{m_r-1})$ in $\mathcal{G}''$ since $count_r = m_r - 1$.

As per the mapping scheme explained earlier, the starting node in $\mathcal{G}''$ is associated as follows: $(s_1 - 1, \ldots s_{r-1} - 1, \langle s_r + 1\rangle_{m_r}) \in \mathcal{G}_{\vec{m}}^{\vec{\rho}} \leftrightarrow (s_1 - 1, \ldots s_{r-1} - 1, \langle 0\rangle_{m_r}) \in \mathcal{G}''$ Since $sign_r = 1$, at the start of Path_Odd in Line (12), the address component in the r^{th} dimension will increment between $\langle 0\rangle_{m_r-1}$ and $\langle m_r - 2\rangle_{m_r-1}$ and in the second pass will decrement from $\langle m_r - 2\rangle_{m_r-1}$ to $\langle 0\rangle_{m_r-1}$ since $sign_r = -1$. Hence no edge is generated that connects $(\langle 0\rangle_{m_r-1}, \langle m_r - 2\rangle_{m_r-1})$ in $\mathcal{G}''$ which implies that the edge $(\langle s_r + 1\rangle_{m_r}, \langle s_r - 1\rangle_{m_r})$ in the r^{th} dimension of is not part of the Hamiltonian path in $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$. Hence the adjacency criterion of $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ is not violated. Also, it is worth observing that the last node of H_2, namely, $(\langle s_1\rangle_{m_1}, \ldots, \langle s_{r-1}\rangle_{m_{r-1}}, \langle s_r - 1\rangle_{m_r})$ is adjacent to the starting node $(s_1, \ldots, s_{r-1}, s_r)$ in the r^{th} dimension. Therefore $H = H_1|H_2$ constitutes a Hamiltonian cycle. In Fig. 4.11, a Hamiltonian cycle for $\mathcal{G}_{75}^{11}$ computed by the algorithm for Case III is illustrated. ■

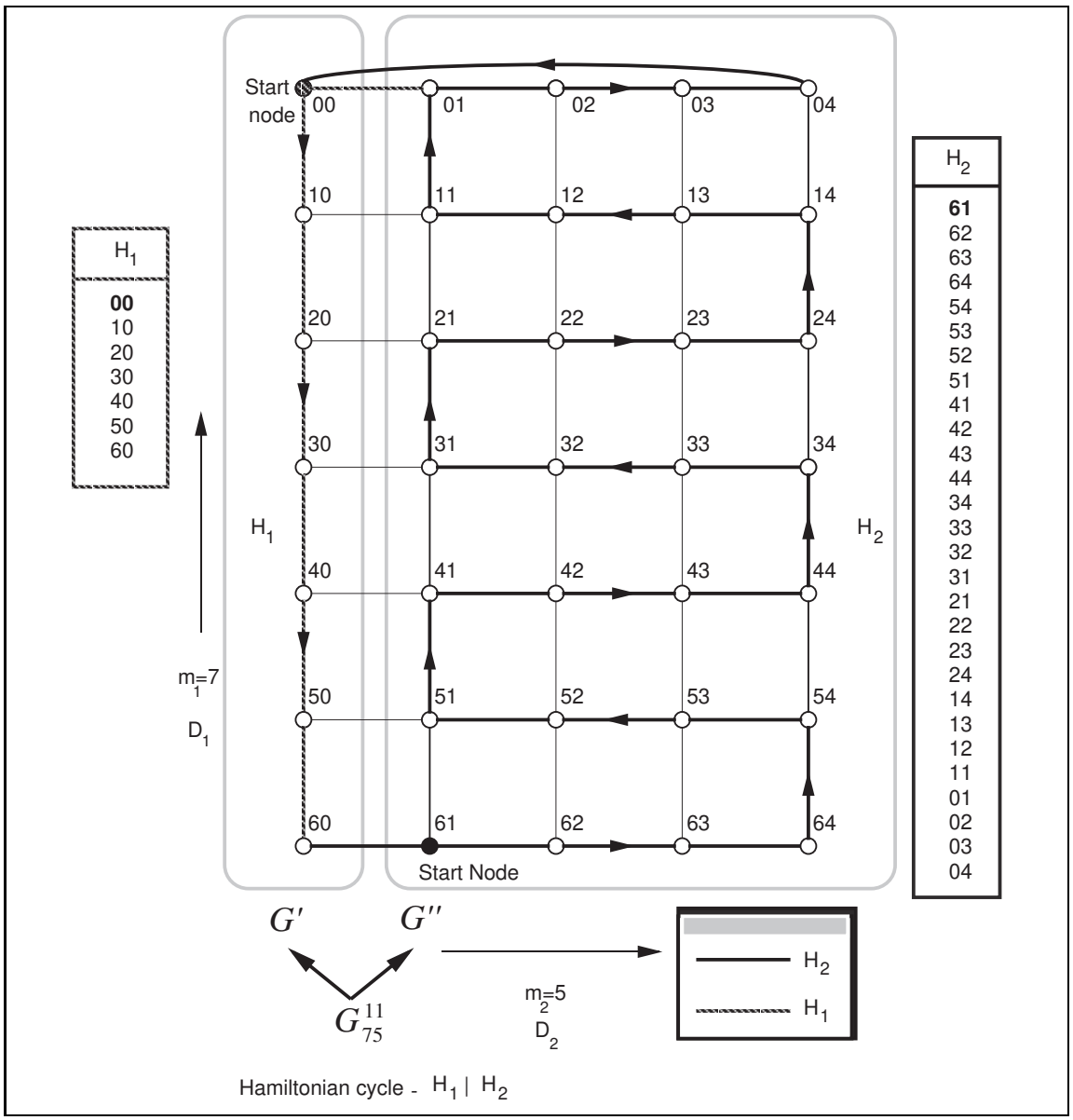


Figure 4.11 *Example of a Hamiltonian Cycle in* $\mathcal{G}_{75}^{11}$ *for the Odd category with node (0,0) as the starting point*

4.3.4 Time complexity

The time required by the three algorithms to enumerate the Hamiltonian cycle is primarily dictated by the total number of nodes M and the dimension r of a hypercycle $\mathcal{G}_m^\rho$. It has been shown previously that it takes exactly $\prod_{i=1}^{r} m_i = \mathcal{O}(M)$ steps to generate a Hamiltonian cycle in $\mathcal{G}_m^\rho$. The time taken to print each node address is $\mathcal{O}(r)$. Hence, the total time complexity is given by

$$\tau = \mathcal{O}(M * r) \tag{4.15}$$

4.3.5 Extension

In section 4.3, we had discussed an algorithm for computing a Hamiltonian cycle in an uniform hypercycle. This can be extended to another class of hypercycles as illustrated by the following corollary:

Corollary 4.1 *Procedure Uniform is valid for a hypercycle* $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ *if the ratios* (m_1/m_2) *and* (m_j^2/m_{j+1}) *are integers for* $j = 2, \ldots, r-1$ *and* $r \geq 2$.

Proof. Assume $r \geq 2$ and $m_1 \neq m_2 \neq \ldots \neq m_r$. Define $\eta_{i-1} = \prod_{j=1}^{i-1} m_j$ for $i = 1, \ldots, r$ with $\eta_0 = 1$. Then Eq. (4.4) can be modified to obtain the x_i^{th} digit as follows:

$$\begin{aligned} \hat{x}_i &= \langle s_i - sign_i + \eta_{i-1}(m_i - sign_i) + sign_i \rangle_{m_i} \\ &= \langle s_i - \eta_{i-1} \rangle_{m_i} \end{aligned} \tag{4.16}$$

If Eq.(4.16) is to be satisfied for all $x_i \in \{0, 1, \ldots, m_i - 1\}$, $i = 1, \ldots, r$, then $\eta_{i-1} \bmod m_i = 0$ which implies that

$$\begin{aligned} m_1 &= q_1 m_2 \\ m_1 m_2 &= q_2 m_3 \\ &\vdots \\ m_1 m_2 \ldots m_{r-1} &= q_{r-1} m_r \end{aligned}$$

where $q_1, \ldots, q_r$ are some integer constants and the recurrence relation to compute them is shown below:

$$q_1 = \frac{m_1}{m_2} \tag{4.17}$$

$$q_i = q_{i-1} \left(\frac{m_i^2}{m_{i+1}} \right), \quad i = 2, \ldots, r-1 \tag{4.18}$$

Since the radices m_i take integer values, Eqs. (4.17) and (4.18) will be complied only if m_i/m_{i+1} is a positive integer when $i = 1$ and and for $i = 2 \ldots r-1$, the ratio m_i^2/m_{i+1}

should also be an integer. It is obvious that when $m_i/m_{i+1} = 1$, the case of a uniform hypercycle is obtained. ■

4.4 Ranking and Unranking of Hamiltonian cycles

Let s be the starting node of a Hamiltonian cycle and $x = (x_1, x_2, \ldots, x_r)$ be any arbitrary node in $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$. Then finding the length of the path from node s to x traced along the Hamiltonian cycle is called the *Ranking* problem. The converse, namely determining the address x from a given rank and starting address is called the *Unranking* problem [78]. In the following section, we derive the relevant formulas for the three cases discussed previously.

The distance between nodes s and x along the Hamiltonian cycle can be obtained by successively calculating the distance traveled in each dimension starting from the highest dimension. Let $s^0 = (s_1^0, s_2^0, \ldots, s_r^0)$[¶] denote the starting address. From procedure Uniform, it can be observed that for each increment in the address digit corresponding to the first dimension, the number of nodes traversed (and hence the distance covered) is $\prod_{j=2}^{r} m_j$. Therefore, for an increment of $c_1 = \langle sign_1(x_1 - s_1^0))\rangle_{m_1}$, the distance traveled is given by $c_1 \prod_{j=2}^{r} m_j$ at which point $s_1^1 = x_1$. This will in turn affect the starting address for dimensions 2 to r. For instance, the second address digit would be offset by a factor of $\langle sign_2 c_1(m_2 - 1)\rangle_{m_2}$ when the address digit in the first dimension is incremented by c_1. Hence the new starting address for dimensions 2 to r at the end of the first iteration is given by:

$$\begin{aligned}
s_2^1 &= \langle s_2^0 + \langle sign_2 c_1(m2 - 1)\rangle_{m_2}\rangle_{m_2} = \langle s_2^0 - \langle sign_2 c_1\rangle_{m_2}\rangle_{m_2} \\
s_3^1 &= \langle s_3^0 + \langle sign_3 c_1 m_2(m_3 - 1)\rangle_{m_3}\rangle_{m_3} = \langle s_3^0 - \langle sign_3 c_1 m_2\rangle_{m_3}\rangle_{m_3} \\
&\vdots \\
s_r^1 &= \langle s_r^0 + \langle sign_r c_1 m_2 m_3 \ldots m_{r-1}(m_r - 1)\rangle_{m_r}\rangle_{m_r} \\
&= \langle s_r^0 - \langle sign_r c_1 m_2 m_3 \ldots m_{r-1}\rangle_{m_r}\rangle_{m_r}
\end{aligned}$$

In the second iteration, the second digit would be incremented by $c_2 = \langle x_2 - s_2^1\rangle_{m_2}$ in order to attain the value of x_2 and consequently a distance of $c_2 \prod_{j=3}^{r} m_j$ is traversed. The first digit remains unaltered since it has already attained its desired value of x_1. The new starting address for dimensions 3 upto r is given as

$$\begin{aligned}
s_3^2 &= \langle s_3^1 - \langle sign_3 c_2\rangle_{m_3}\rangle_{m_3} \\
s_4^2 &= \langle s_4^1 - \langle sign_4 c_2 m_3\rangle_{m_4}\rangle_{m_4}
\end{aligned}$$

[¶]The superscript denotes the iteration number. The initial iteration has a superscript of 0.

$$
\begin{aligned}
&\vdots \\
s_r^2 &= \langle s_r^1 + \langle sign_r c_2 m_3 m_4 \ldots m_{r-1}(m_r - 1)\rangle_{m_r}\rangle_{m_r} \\
&= \langle s_r^1 - \langle sign_i c_2 m_3 m_4 \ldots m_{r-1}\rangle_{m_r}\rangle_{m_r}
\end{aligned}
$$

Thus, when each c_i, $1 \leq i \leq r$ is computed, the new starting address s_j^i, $j = i+1, \ldots, r$ is calculated as given above. The r^{th} coefficient, c_r at the beginning of the r^{th} iteration is given by $c_r = \langle sign_r(x_r - s_r^{r-1})\rangle_{m_r}$ and when the distance of c_r is covered all the source and destination address digits would be matched.

In general, the distance between nodes s and x on the Hamiltonian path computed by procedure Uniform is given as follows.

$$R_{unif} = R_p(s, x, m, r, sign) \tag{4.19}$$

where

$$
\begin{aligned}
R_p(s, x, \vec{m}, r, sign) &= \sum_{i=1}^{r} c_i w_i \\
c_i &= \langle sign_i(x_i - s_i^{i-1})\rangle_{m_i} \\
w_i &= \frac{M}{\prod_{j=1}^{i} m_j} \\
s_j^i &= \begin{cases} \langle s_j^{i-1} - \langle sign_j c_i\rangle_{m_j}\rangle_{m_j} & \text{if } j = i+1 \\ \langle s_j^{i-1} - \langle sign_j c_i \prod_{k=i+1}^{j-1} m_k\rangle_{m_j}\rangle_{m_j} & \text{Otherwise} \end{cases} \\
j &= i+1, \ldots, r, \quad i \neq r
\end{aligned} \tag{4.20}
$$

As for the unranking problem, we can use the techniques developed for the ranking problem. Given a starting node $s^0 = (s_1^0, s_2^0, \ldots, s_r^0)$ and a rank D, the methodology is to successively compute the address digits x_i, $(i = 1, \ldots, r)$ starting with the highest dimension. Observe that the most significant digit x_1 of the destination will be different from the most significant digit s_1^0 of the starting address if $D^0 = D \geq \prod_{i=2}^{r} m_i = M/m_1 = w_1$. Then x_1 is given as $x_1 = \langle s_1^0 + sign_1 c_1\rangle_{m_1}$ where $c_1 = \left\lfloor \frac{D^0}{w_1} \right\rfloor$. Thus, at a distance of $c_1 w_1$ from s, the new starting address will be given by

$$
\begin{aligned}
s_1^1 &= x_1 \\
s_2^1 &= \langle s_2^0 - \langle sign_2 c_1\rangle_{m_2}\rangle_{m_2} \\
s_3^1 &= \langle s_3^0 - \langle sign_3 c_1 m_2\rangle_{m_3}\rangle_{m_3} \\
&\vdots \\
s_r^1 &= \langle s_r^0 - \langle sign_r c_1 \prod_{k=2}^{r-1} m_k\rangle_{m_r}\rangle_{m_r}
\end{aligned}
$$

while the remaining distance at the end of the first iteration is given by

$$\begin{aligned} D^1 &= D^0 - c_1 w_1 = D^0 - \left\lfloor \frac{D^0}{w_0} \right\rfloor w_1 \\ &= D^0 \bmod w_1 \end{aligned}$$

Since $D^1 < w_1$, then for the remaining steps needed to cover D^1, the most significant digit x_1 will not change. Thus recursively, one can generate the remaining address digits x_i, $2 \leq i \leq r$ updating the starting address digits for dimensions j, $\forall j$ such that $i < j \leq r$. The general formula for unranking a uniform Hypercycle is given by

$$\begin{aligned} U_{unif}(s, D, \vec{m}, r, sign) &= (x_1, \ldots, x_r) \qquad (4.21) \\ x_i &= \langle s_i^{i-1} + sign_i c_i \rangle_{m_i} \quad i = 1, \ldots, r \\ w_i &= \frac{M}{\prod_{j=1}^{i} m_j} \\ D^i &= D^{i-1} \bmod w_i \\ c_i &= \left\lfloor \frac{D^{i-1}}{w_i} \right\rfloor \\ s_j^i &= \begin{cases} \langle s_j^{i-1} - \langle sign_j c_i \rangle_{m_j} \rangle_{m_j} & \text{if } j = i+1 \\ \langle s_j^{i-1} - \langle sign_j c_i \prod_{k=i+1}^{j-1} m_k \rangle_{m_j} \rangle_{m_j} & \text{Otherwise} \end{cases} \\ j &= i+1, \ldots, r \quad i \neq r \end{aligned}$$

It is worth mentioning that the ranking and unranking formulas for the Hamiltonian cycle computed by procedure Uniform are also valid for the special set of hypercycles mentioned in Corollary 4.1.

When the hypercycle is *even*[‖], the ranking formula can be obtained in terms of the formula derived for the uniform case since the Hamiltonian cycle is computed in two parts P_1 and P_2 using procedure Even as a reflected gray code in dimension 1 such that $|P_1| = |P_2| = \frac{\prod_{i=1}^{r} m_i}{2} = M/2$. The distance traveled in the first dimension is given by $\langle sign_1(x_1 - s_1^0) \rangle_{m_1} \prod_{i=2}^{r} m_i$. If this distance is less than M/2, then $\langle sign_1(x_1 - s_1^0) \rangle_{m_1} \prod_{i=2}^{r} m_i < M/2$ which implies that $\langle sign_1(x_1 - s_1^0) \rangle_{m_1} < m_1/2$ and that x lies on P_1. Thus, by comparing the offset in the first dimension, namely, $\langle sign_1(x_1 - s_1^0) \rangle_{m_1}$ to $m_1/2$, it can be ascertained whether the path leading to x lies in P_1. If this is the case, then the ranking formula used for the uniform case (c.f. Eq. (4.19)) can be applied. On the other hand, if x lies in P_2, we first find the start node of P_2 (say node s') and then complete the path to x, now entirely within P_2. The algorithm followed is similar to that of the uniform case but with the direction

[‖]Note that m_1 is even here.

vector, $sign$, negated in all dimensions except in dimension 1 (c.f. Fig. 4.5, Line (8)). The ranking formula for the even case is given as:

$$
\begin{aligned}
R_{even} &= \begin{cases} R_p(s, x, \vec{m}, r, sign) & \text{if } \langle sign_1(x_1 - s_1)\rangle_{m_1} < \frac{m_1}{2} \\ \frac{M}{2} + R_p(s', x, \vec{m}, r, sign') & \text{otherwise} \end{cases} \\
sign_i &= sign_i^0 \qquad\qquad i = 1, \ldots, r \qquad (4.22) \\
s'_i &= \begin{cases} \langle s_i^0 + sign_i^0 \left(\frac{m_i}{2}\right)\rangle_{m_i} & \text{if } i = 1 \\ \left\langle s_i^0 - sign_i^0 \left(\frac{\prod_{j=1}^{i-1} m_j}{2} \right) \right\rangle_{m_i} & \text{otherwise} \end{cases} \\
sign'_i &= \begin{cases} sign_i^0 & \text{if } i = 1 \\ -sign_i^0 & \text{otherwise} \end{cases}
\end{aligned}
\qquad (4.23)
$$

The formula for the unranking of an even hypercycle follows directly from the derivation for the unranking of the uniform case and the above discussion as follows:

$$
U_{even} = \begin{cases} U_p(s', D', \vec{m}, r, sign') & \text{if } D \geq \frac{M}{2} \\ U_p(s, D, \vec{m}, r, sign) & \text{otherwise} \end{cases}
$$

where $D' = D - M/2$.

Before presenting the analysis for the odd case, we derive the ranking of the path computed by procedure Path_Odd in order to calculate the distance in partitions $\mathcal{G}'$ and $\mathcal{G}''$. This is similar to the formula derived for the uniform case. From procedure Path_Odd, we know that the direction vector in dimension i is negated alternately. Starting from the first dimension, for an increment of $c_1 = \langle sign_1(x_1 - s_1^0)\rangle_{m_1}$, the number of nodes traversed is $w_1 = \prod_{i=2}^{r} m_i$ and hence the distance covered is $c_1 w_1$. The new starting address digits in the remaining j dimensions ($j = 2, \ldots, r$) is given by (c.f. Lemma 4.6) such that

$$
\begin{aligned}
s_2^1 &= \left\langle s_2^0 + \left\lceil \frac{c_1}{2} \right\rceil sign_2^0(m_2 - 1) - \left\lfloor \frac{c_1}{2} \right\rfloor sign_2^0(m_2 - 1) \right\rangle_{m_2} \\
&= \left\langle s_2^0 - \left(\left\lceil \frac{c_1}{2} \right\rceil - \left\lfloor \frac{c_1}{2} \right\rfloor \right) sign_2^0 \right\rangle_{m_2} \\
s_3^1 &= \left\langle s_3^0 + \left\lceil \frac{c_1 m_2}{2} \right\rceil sign_3^0(m_3 - 1) - \left\lfloor \frac{c_1 m_2}{2} \right\rfloor sign_3^0(m_3 - 1) \right\rangle_{m_3} \\
&= \left\langle s_3^0 - \left(\left\lceil \frac{c_1 m_2}{2} \right\rceil - \left\lfloor \frac{c_1 m_2}{2} \right\rfloor \right) sign_3^0 \right\rangle_{m_3} \\
&\vdots \\
s_r^1 &= \left\langle s_r^0 + \left\lceil \frac{c_1 m_2 m_3 \ldots m_{r-1}}{2} \right\rceil sign_r^0(m_r - 1) \right. \\
&\qquad \left. - \left\lfloor \frac{c_1 m_2 m_3 \ldots m_{r-1}}{2} \right\rfloor sign_r^0(m_r - 1) \right\rangle_{m_r} \\
&= \left\langle s_r^0 - \left(\left\lceil \frac{c_1 m_2 m_3 \ldots m_{r-1}}{2} \right\rceil - \left\lfloor \frac{c_1 m_2 m_3 \ldots m_{r-1}}{2} \right\rfloor \right) sign_r^0 \right\rangle_{m_r}
\end{aligned}
$$

In general, for an increment of $c_i = \langle sign_i(x_i - s_i^{i-1}) \rangle_{m_i}$, the starting address digits for dimensions j, $j = i+1, \ldots, r$, $i \neq r$ is given by

$$
s_j^i = \begin{cases} \left\langle s_j^{i-1} - (\lceil \frac{c_i}{2} \rceil - \lfloor \frac{c_i}{2} \rfloor)\, sign_j^0 \right\rangle_{m_j} & \text{if } j = i+1 \\ \left\langle s_j^{i-1} - \left(\left\lceil \frac{c_i \prod_{l=i+1}^{j-1} m_l}{2} \right\rceil - \left\lfloor \frac{c_i \prod_{l=i+1}^{j-1} m_l}{2} \right\rfloor \right) sign_j^0 \right\rangle_{m_j} & \text{Otherwise} \end{cases}
$$

Observe that when $\prod_{i=1}^{r-1} m_i$ is odd, the above equations simplify as follows.

$$
\begin{aligned}
s_j^i &= \begin{cases} \left\langle s_j^{i-1} \right\rangle_{m_j} & \text{if } c_i \text{ is even} \\ \left\langle s_j^{i-1} - sign_j^0 \right\rangle_{m_j} & \text{if } c_i \text{ is odd} \end{cases} \\
j &= i+1, \ldots, r \quad i \neq r
\end{aligned}
$$

Hence the ranking formula for the path computed by Path_Odd is given by

$$
\begin{aligned}
R_{path_odd}(s, x, \vec{m}, r, sign) &= \sum_{i=1}^{r} c_i w_i \\
c_i &= (sign_i(x_i - s_i^{i-1})) \bmod m_i \\
w_i &= \frac{M}{\prod_{j=1}^{i} m_j} \\
s_j^i &= \begin{cases} \langle s_j^{i-1} - sign_j^0 \rangle_{m_j} & \text{if } c_i \text{ is odd} \\ \langle s_j^{i-1} \rangle_{m_j} & \text{if } c_i \text{ is even} \end{cases} \\
j &= i+1, \ldots, r, \quad i \neq r
\end{aligned}
$$

For the *odd* hypercycle, it is necessary to determine whether node x falls into either $\mathcal{G}'$ or $\mathcal{G}''$. By comparing the r^{th} dimensional digits x_r and s_r, the appropriate graph partition can be identified. If $x_r = s_r$, we can conclude that node x lies in $\mathcal{G}'$ since from the discussion in Theorem 4.6, we know that the r^{th} dimensional digit is unaltered in the Hamiltonian path for $\mathcal{G}'$. On the other hand if $x_r \neq s_r$, then x lies in $\mathcal{G}''$. Thus, if node $x \in \mathcal{G}'$, the rank can be computed directly using the formula derived R_{path_odd} initializing the appropriate parameters such as the the dimension, starting node and current node corresponding to $\mathcal{G}'$. On the other hand, if node x lies in $\mathcal{G}''$, the new starting point as explained for the odd case is given by $(\langle s_1^0 - sign_1^0 \rangle_{m_1}, \ldots, \langle s_{r-1}^0 - sign_{r-1}^0 \rangle_{m_{r-1}}, \langle s_r^0 + sign_r^0 \rangle_{m_r})$. The rank in this case is obtained by adding $M' = |\mathcal{G}'|$ to the distance traversed in $\mathcal{G}''$. The variables x'', s'' and $sign''$ are computed based on the theory and mapping scheme for $\mathcal{G}''$ discussed for the Odd case. The ranking formula for the odd hypercycle is given by

$$R_{odd} = \begin{cases} R_{path_odd}(s', x', \vec{m}', r', sign') & \text{if } s_r = x_r \\ \\ M' + R_{path_odd}(s'', x'', \vec{m}'', r, sign'') & \text{otherwise} \end{cases}$$

where

$$\begin{aligned}
r' &= r - 1 \\
M' &= \prod_{i=1}^{r'} m_i \\
\vec{m}' &= [m_1' \; m_2' \; \ldots m_j' \ldots \; m_{r'}'] \text{ such that } \; m_j' = m_j, \;\; j = 1, \ldots, r' \\
s' &= \left(\langle s_1^0 \rangle_{m_1'}, \langle s_2^0 \rangle_{m_2'}, \ldots \langle s_{r'}^0 \rangle_{m_{r'}} \right) \\
x' &= \left(\langle x_1 \rangle_{m_1'}, \langle x_2 \rangle_{m_2'}, \ldots \langle x_{r'} \rangle_{m_{r'}} \right) \\
sign_j' &= sign_j^0, \;\; \text{for } j = 1, \ldots, r' \\
\vec{m}'' &= [m_1'' \; m_2'' \; \ldots m_j'' \ldots \; m_r''] \text{ such that } \qquad (4.24) \\
m_j'' &= \begin{cases} m_j & \text{for } j = 1 \text{ to } r-1 \\ \\ m_j - 1 & \text{for } j = r \end{cases} \\
s'' &= \left(\langle s_1^0 - sign_1^0 \rangle_{m_1''}, \langle s_2^0 - sign_2^0 \rangle_{m_2''}, \ldots, \langle s_{r-1}^0 - sign_{r-1}^0 \rangle_{m_{r-1}''}, \langle 0 \rangle_{m_r''} \right) \\
x'' &= \left(\langle x_1 \rangle_{m_1''}, \langle x_2 \rangle_{m_2''}, \ldots \langle x_{r-1} \rangle_{m_{r-1}''}, \langle \langle sign_r(x_r - s_r) - 1 \rangle_{m_r} \rangle_{m_r''} \right) \\
sign_j'' &= \begin{cases} -sign_j^0 & \text{for } j = 1 \text{ to } r-1 \\ \\ sign_j^0 & \text{for } j = r \end{cases}
\end{aligned}$$

The unranking for the odd hypercycle can be obtained in terms of an unranking formula for procedure Path_Odd whose derivation is similar to that of the unranking for the uniform case except for the manner in which the starting address is updated. The simplified formula is given by

$$\begin{aligned}
U_{path_odd}(s, D, \vec{m}, r, sign) &= (x_1, \ldots, x_r) \\
x_i &= \langle s_i^{i-1} + sign_i c_i \rangle_{m_i} \quad i = 1, \ldots, r \\
w_i &= \frac{M}{\prod_{j=1}^{i} m_j} \\
D^i &= D^{i-1} \bmod w_i \\
c_i &= \left\lfloor \frac{D^i}{w_i} \right\rfloor \\
s_j^i &= \begin{cases} \langle s_j^{i-1} - \langle sign_j \rangle_{m_j} \rangle_{m_j} & \text{if } c_i \text{ is odd} \\ \langle s_j^{i-1} \rangle_{m_j} & \text{if } c_i \text{ is even} \end{cases} \\
j &= i+1, \ldots, r \quad i \neq r
\end{aligned}$$

By comparing the distance D with M', the node address x can be calculated from either $\mathcal{G}'$ or $\mathcal{G}''$ using the mapping technique described earlier:

$$U_{odd} = \begin{cases} U_{path_odd}(s'', D'', \vec{m}'', r, sign'') & \text{if } D \geq M' \\ U_{path_odd}(s', D, \vec{m}', r', sign') & \text{otherwise} \end{cases}$$

where $D'' = D - M$.

4.5 Hamiltonian Decomposition

Based on the general definition for Hamiltonian decomposition given in [98], we state the following with respect to hypercycles.

Definition 4.4 *An r-dimensional hypercycle is said to have a Hamiltonian decomposition if its edge set can be partitioned into* $\lfloor \frac{d}{2} \rfloor$ *Hamiltonian cycles where d is the degree of* $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ *and* $M \geq 3$.

Since there are $Md/2$ edges in any hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$, the maximum number of edge-disjoint Hamiltonian cycles is given by $\left\lfloor \frac{M(\frac{d}{2})}{M} \right\rfloor = \lfloor \frac{d}{2} \rfloor$. The complete graph K_n and the n-cube have a Hamiltonian decomposition [98] for all n. Alspach [98] raises the question whether every connected Cayley graph on an abelian group has a Hamiltonian decomposition. For graphs

upto degree 5, he has shown that this is indeed true. In [99], Bermond *et. al* have shown that any t^{th} power of a cycle** can be decomposed into t-Hamiltonian cycles if $2t < m$. Obviously, the 1-dimensional hypercycle $\mathcal{G}_{m_1}^{\rho_1}$ is isomorphic to the t^{th}-power of a cycle when $t = \rho_1$ by definition. For the case, when $m_1 = 2t$, the hypercycle is fully connected with a degree of $2t - 1$. Hence, from Definition 4.4 and Theorem 1 in [98, pp. 10], the maximum number of Hamiltonian decompositions is given by $t - 1 = \rho_1 - 1$ along with a perfect matching.(See Section 1.5 in Chapter 1 for the definition of perfect matching)

Lemma 4.9 *For a 1-dimensional hypercycle $\mathcal{G}_{m_1}^{\rho_1}$ when (m_1, j) are mutually prime $\forall j$, $2 \leq j \leq \rho_1$ except for the case $(m_1, 1)$, there exists a set of $\rho_1 = \frac{d}{2}$ edge-disjoint Hamiltonian cycles defined by $\sigma = \{H_1, H_2, \ldots, H_i, \ldots, H_{\rho_1}\}$ where $H_i = (\langle s \rangle_{m_1}, \langle s + i \rangle_{m_1}, \langle s + 2i \rangle_{m_1}, \ldots, \langle s + (m_1 - 1)i \rangle_{m_1})$††, $\forall i$, $1 \leq i \leq \rho_1$ and s being a given starting node.*

Proof. Firstly we show that each node in H_i is unique. Secondly, we prove that paths in H_i and H_j are edge-disjoint where $1 \leq i \neq j \leq \rho_1$.

- Assume any two nodes $\langle s + ci \rangle_{m_1}$ and $\langle s + c_1 i \rangle_{m_1}$ are identical where $0 \leq c \neq c_1 \leq m_1 - 1$. This implies that $\langle (c - c_1)i \rangle_{m_1} = 0$ which is a contradiction as given below.

  Case 1: $i = 1$. Then $0 < \langle (c - c_1) \rangle_{m_1} \leq m_1 - 1$ since $c \neq c_1$.

  Case 2: $2 \leq i \leq \rho_1$. In this case, $\langle (c - c_1)i \rangle_{m_1} \neq 0$ since $0 < \langle (c - c_1) \leq m_1 - 1$ and (m_1, i) are mutually prime.

- Let there exist some edge $(x, x + i) \in H_i$ and $(x, x + j) \in H_j$ where $1 < i \neq j \leq \rho_1$ and x being an arbitrary node. If these two edges are identical, then $\langle x + i \rangle_{m_1} = \langle x + j \rangle_{m_1} \Rightarrow \langle (i - j) \rangle_{m_1} = 0$ is a contradiction since $i \neq j$.

■

With respect to the question of Hamiltonian decomposability in multidimensional hypercycles, we state the following proposition.

Proposition 4.1 *There exist at least $\check{\sigma}$ edge-disjoint Hamiltonian decompositions in an r-dimensional hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ where*

$$\check{\sigma} \quad = \quad r \left(\min_{1 \leq i \leq r} \left\lfloor \frac{d_i}{2} \right\rfloor \right), \qquad d_i \text{ being the degree in dimension } i \tag{4.25}$$

Proof. Alspach [98, pp. 11] has shown that if $\mathcal{G}_1, \mathcal{G}_2, \ldots \mathcal{G}_r$ can be decomposed into n cycles each, then the cartesian product $\mathcal{G}_1 \times \mathcal{G}_2 \times \ldots \times \mathcal{G}_r$ can be decomposed into rn Hamiltonian cycles. We know that any r-dimensional hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ has at least one Hamiltonian cycle.

**The t^{th} power of a cycle of order n is a circulant in which node i is connected node j if $|i - j| \bmod n \leq t$.

††Here node addresses are evaluated modulo m_1.

Since $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ can be expressed as a cartesian product of r 1-dimensional hypercycles (refer Eq. (1.8)), the smallest Hamiltonian decomposition in any of the product 1-dimensional hypercycles is given by $\min_{1\leq i\leq r}\lfloor\frac{d_i}{2}\rfloor$ as discussed previously. Hence from the above, we can conclude that the lower bound $\breve{\sigma}$ is given by $r\left(\min_{1\leq i\leq r}\lfloor\frac{d_i}{2}\rfloor\right)$. ■

4.5.1 Extension

From an analysis of the address pattern generated by procedure Uniform for a 2-dimensional uniform hypercycle, one can make the following observation on Hamiltonian decomposability as a special case.

Corollary 4.2 *For a 2-dimensional uniform hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$, where $m = m_1 = m_2 \geq 3$ and $sign_1 = sign_2 = 1(-1)$, there exists 2 edge disjoint Hamiltonian cycles H and $\hat{H}$ where $\hat{H}$ is obtained from H by swapping the address digits printed by procedure Uniform in dimensions 1 and 2.*

Proof. Assuming $\rho_1, \rho_2 \geq 1$, we can infer that there exists at least 2 edge-disjoint Hamiltonian cycles from Proposition 4.1 since $m \geq 3$ and $\lfloor\frac{d_1}{2}\rfloor, \lfloor\frac{d_2}{2}\rfloor \geq 1$. Let H be the Hamiltonian cycle generated by procedure Uniform for a given starting address $s = (s_1^0, s_2^0)$ and direction vector $sign$ be initialized such that $sign_1 = sign_2 = 1(-1)$. Let $\hat{H}$ be another Hamiltonian cycle obtained by swapping the address digits of each node of H. Observe that $\hat{H}$ is also Hamiltonian since each successive address is distinct and differs by unity from the previous in exactly one dimension. Let $((x_1, x_2), (x_1 + sign_1, x_2))$[‡‡] and $((x_1, x_2), (x_1, x_2 + sign_2))$ be two possible edges in H. If $\hat{H}$ is edge-disjoint with H, then we need to show that there exists no common edge between $\hat{H}$ and H as follows.

1. If $((x_1, x_2), (x_1 + sign_1, x_2)) \in H$ then $((x_2, x_1), (x_2, x_1 + sign_1)) \notin H$.
   Observe that whenever increments occur in dimension 1 between nodes (x_1, x_2) and $(x_1 + sign_1, x_2)$, the address component in dimension 2 remains unaltered. Since it has been assumed above that $((x_1, x_2), (x_1 + sign_1, x_2)) \in H$ after covering a distance of $m\langle sign_1(x_1 + sign_1 - s_1^0)\rangle_m$ in dimension 1 from the start node, the new starting address in dimension 2, namely, s_2^1 will be identical to x_2 and
   $$s_2^1 \quad = \quad \langle s_2^0 - sign_2 sign_1(x_1 + sign_1 - s_1^0)\rangle_m = x_2 \tag{4.26}$$
   We can prove that $((x_2, x_1), (x_2, x_1 + sign_1)) \notin H$ by calculating the relative distance between nodes (x_2, x_1) and $(x_2, x_1 + sign_1)$ in H and showing that it is greater than unity. Since these two nodes have the same component in dimension 1, namely, x_2, we

[‡‡] All node addresses are evaluated modulo m.

need to ascertain which of the two nodes appears first in H. From the ranking formula for the uniform case, we can determine the starting point (x_2, s_2^1) in dimension 2 from the original starting point $s = (s_1^0, s_2^0)$. Observe that $s_2^1 = \langle s_2^0 - sign_2 sign_1(x_2 - s_1^0)\rangle_m$. Substituting for x_2 from Eq. (4.26), we have

$$\begin{aligned} s_2^1 &= \langle s_2^0 - sign_2 sign_1(s_2^0 - sign_2 sign_1(x_1 + sign_1 - s_1^0) - s_1^0)\rangle_m \\ &= \langle s_2^0 - sign_2 sign_1 s_2^0 + sign_2^2 sign_1^2 x_1 + sign_2^2 sign_1^3 - sign_2^2 sign_1^2 s_1^0 \\ &\quad + sign_2^2 sign_1^2 s_1^0\rangle_m \\ &= \langle s_2^0 - sign_2 sign_1 s_2^0 + x_1 + sign_1\rangle_m \\ &= \langle s_2^0(1 - sign_2 sign_1) + x_1 + sign_1\rangle_m \end{aligned}$$

Since $sign_1 = sign_2 = 1(-1)$, the new starting address in dimension 2 is given by $s_2^1 = \langle x_1 + sign_1\rangle_m$. Hence, node $(x_2, x_1 + sign_1)$ occurs before (x_2, x_1) in H and the distance separating these two nodes is given by $\langle sign_2(x_1 - (x_1 + sign_1)\rangle_m = \langle -sign_2 sign_1\rangle_m = m - 1 > 1$ for $m \geq 3$. Consequently $((x_2, x_1), (x_2, x_1 + sign_1)) \notin H$.

2. If $(x_1, x_2), (x_1, x_2 + sign_2) \in H$ then $((x_2, x_1), (x_2 + sign_2, x_1)) \notin H$.

   Let $((x_1, x_2), (x_1, x_2 + sign_2)) \in H$. As before, we need to show that the edge obtained after swapping the nodes (x_1, x_2) and $(x_1, x_2 + sign_2)$, namely, $((x_2, x_1), (\langle x_2 + sign_2\rangle_m, x_1)) \notin H$. The distance (rank) of node (x_2, x_1) from (s_1^0, s_2^0) is given by

$$\begin{aligned} R' &= m\langle sign_1(x_2 - s_1^0)\rangle_m + \langle sign_2(x_1 - s_2^1)\rangle_m \\ &= m\langle sign_1(x_2 - s_1^0)\rangle_m + \langle sign_2(x_1 - (s_2^0 - sign_2 sign_1(x_2 - s_1^0))\rangle_m \end{aligned} \quad (4.27)$$

   Similarly the distance of $(x_2 + sign_2, x_1)$ from (s_1^0, s_2^0) is given as

$$\begin{aligned} R'' &= m\langle sign_1(x_2 + sign_2 - s_1^0) + sign_2(x_1 - s_2^1)\rangle_m \\ &= m\langle sign_1(x_2 + sign_2 - s_1^0) \\ &\quad + sign_2(x_1 - (s_2^0 - sign_2 sign_1(x_2 + sign_2 - s_1^0))\rangle_m \end{aligned} \quad (4.28)$$

   Subtracting the difference between Eqs. (4.27) and (4.28) assuming node (x_2, x_1) occurs before $(x_2 + sign_2, x_1)$, we obtain

$$\begin{aligned} R' - R'' &= m\langle -sign_1 sign_2\rangle_m - \langle sign_1 sign_2^3\rangle_m \\ &= m(m-1) - 1 > 1 \quad \text{(Since } sign_1 = sign_2 = 1(-1) \text{ and } m > 2.) \end{aligned}$$

   On the other hand, if node $(x_2 + sign_2, x_1)$ supersedes (x_2, x_1) in H, then the difference between Eqs. (4.28) and (4.27),

$$\begin{aligned} R'' - R' &= m\langle sign_1 sign_2\rangle_m + \langle sign_1 sign_2^3\rangle_m \\ &= m + 1 > 1 \end{aligned}$$

Therefore $((x_2, x_1), (x_2 + sign_2, x_1)) \notin H$ and $\hat{H}$ is edge disjoint with H.

■

4.6 Conclusion

In this Chapter, hypercycles are proved to be Hamiltonian and a generalized m-ary gray code algorithm for computing a Hamiltonian cycle for any r-dimensional hypercycle has been presented. This algorithm can be applied to other interconnection networks such as toruses, k-ary n-cubes and generalized hypercubes which are encompassed by hypercycles. Expressions for the ranking and unranking of the Hamiltonian cycle generated by the algorithm were derived including some observations on Hamiltonian decomposability. The Hamiltonian cycle algorithm discussed here can be distributively computed at each node based on the prioritized algorithm discussed for the three cases, the starting address and the distance traversed in each dimension upto the current node and hence can be easily implemented in hardware. Other interesting problems to investigate would be to determine an upper bound on the number of Hamilton cycles in hypercycles.

Chapter 5

Deadlock Preventing Routing Algorithms for Hypercycles

Efficient routing of messages is an issue of paramount importance in the design of a parallel computer system. Inter processor communication is accomplished by sending messages that are routed between source-destination nodes across links in the network. The situation becomes complex when several messages coexist and compete for communication resources at any given moment giving rise to potential deadlocks and network congestion.

Many routing algorithms for interconnection networks have been reported in recent years [100, 101]. Routing implies switching mechanism as well as the algorithm itself. There are generally three types of switching mechanisms for interprocessor communication, namely

1. Circuit switching
2. Packet switching
3. Virtual-cut-through

In circuit switching, an entire physical path is established between the source and destination addresses before a message is transmitted. Whereas in the second scheme, the message is buffered at intermediate nodes and forwarded to the destination in a store-and-forward fashion. This implies that each processor takes part in the actual routing process and the communication latency depends on the message length, queueing delay and number of hops required to reach the destination. Furthermore, there is the possibility of information being lost due to overflow of the buffers owing to a sudden surge in network traffic. Virtual-cut-through mechanism [102] attempts to combine the benefits of circuit and packet switching by breaking the message into small packets that are pipelined through the network. As soon as enough routing information is available at an intermediate node, and the output links are free, the packets are forwarded even before the entire message is received.

If a no output links are free, the packets are buffered at the blocked node and no links are held. At heavy loads, virtual-cut-through behaves like packet switching mechanism while at light loads it is like circuit switching.

Wormhole routing [29] is a special case of the virtual-cut-through flow control protocol in which the message is divided into a sequence of constant sized packets. Each packet contains one or more flow control digits *flits* that provide routing and sequencing information. The flit at the head of the message governs the route and the remaining flits follow in a pipeline fashion. The buffer size is usually small, equivalent to the size of the message header and latency is minimal compared to store-and-forward [103].

Definition 5.1 *A routing scheme is said to be* **oblivious** *or* **deterministic** *if the path a message takes is solely determined by the source-destination address.*

Example of the oblivious routing scheme include the e-cube algorithm for hypercubes [29] and the x-y-z routing for 3-dimensional grids [15].

Definition 5.2 *A routing algorithm is said to be* **adaptive** *if it can use more than one path between a source-destination pair.*

An adaptive routing protocol [104] makes use of the network state such as bottlenecks, congestion and link failures to make the routing decision. These can be further classified into minimal and non-minimal routing. In minimal routing (greedy strategy) the length of the path chosen is always minimal. Such algorithms include the hyperswitch [17] where the recent history of the path is carried in the header of the message and the router decides to backtrack if all the channels[†] that can be used to extend the path are occupied. Pure backoff and random selection among the available channels to extend the path in a circuit switched environment have been introduced in [33]. In addition, delaying the back-off decision for a small interval of time (persist backtracking), increases the probability that a non-deadlocking path would be able to extend its path, while the finite time-out guaranteed that any deadlocks would be resolved [105], and this resulted in a higher throughput. In [106], virtual channels, i.e., logical channels which are multiplexed over physical channels, were introduced to break circular dependencies and priorities were assigned to channels to accomplish deadlock-free routing.

Duato [107] gives methodologies for designing deadlock-free adaptive protocols for wormhole routing by separating virtual channels into the restricted and unrestricted channels. Any virtual channel belonging to the unrestricted group can be used for routing provided

[†]Channels are the physical communication links between nodes in the network.

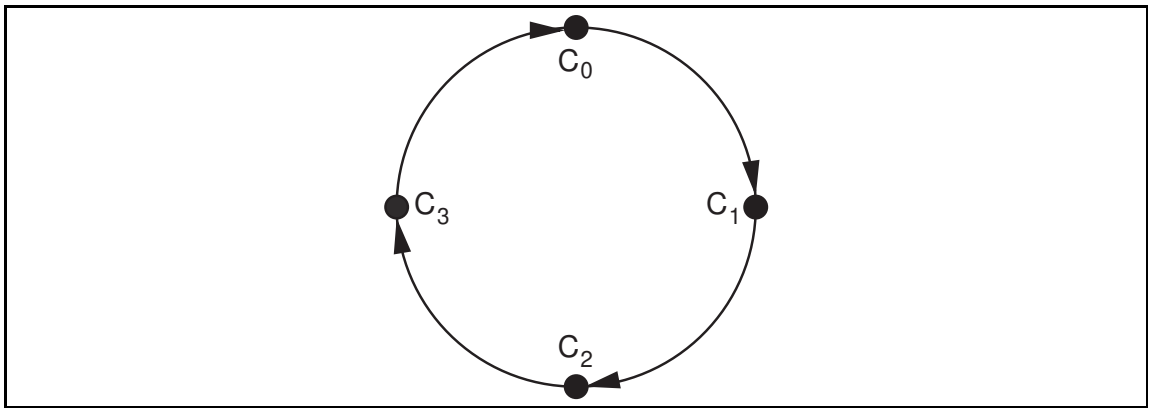


Figure 5.1 *Channel Dependency graph*

there is no block. If a message gets blocked, then it is routed on the restricted channel using deadlock-free routing such as the e-cube.

For non-minimal adaptive routing, Valiant [108] suggested a two-phase routing scheme in which a message is first routed to a random intermediate node and from there to its destination. Examples of implementations of randomized routing [109] include the T9000 of Inmos and the Connection Machine CM5. Konstandinidou [110] has introduced the Chaos router where a message may be derouted (i.e. send towards a node that is not on the shortest path towards its destination) if the node's buffer pool is filled up. The Chaos router is non deterministic, in the sense that the message is propagated on a randomly chosen dimension from the profitable ones and that a random message is chosen from the node's pool to be derouted.

A very important feature of routing algorithms is that they should be free from deadlocks. A deadlock is defined as the situation wherein no process in a system can proceed without acquiring a resource already held by some other process. In the case of interconnection networks, deadlocks can occur when resources (links) are assigned so that the completion of a partial path requires a segment which has been already allocated to some other path which in turn needs a link from the first partial path. Deadlocks may occur easily in cases where the paths are chosen at random. Hence routing algorithm must have some form of mechanism to avoid deadlocks or a way to recover from this situation.

Deadlocks can be analyzed in terms of the *channel dependency graph* [106] which is a directed graph $DG = (V, E)$ where $V(DG)$ is the vertex set corresponding to the links (channels) in the network while $E(DG)$ denotes the set of edges that connect a pair of channels. If $(c_i, c_j) \in E(DG)$, then input link c_i is connected to the outgoing link c_j. An

example of a dependency graph is shown in Fig. 5.1. Dally [111] gives a comprehensive article on deadlock free routing in multiprocessor networks. He gives the necessary and sufficient conditions for a routing algorithm in a network to be deadlock free by proving that its corresponding channel dependency graph should be acyclic.

5.1 Prioritized Dimension *e*-cube routing

One of the popular methods for deadlock avoidance is the imposition of a routing restriction in the algorithm. In *e*-cube or dimension order routing, links are assigned priorities such that a lower order link cannot be issued unless a higher order link is obtained. If messages are routed in order of decreasing dimension and hence decreasing link priorities, no cycles are formed in the dependency graph and the e-cube algorithm is deadlock free. The *e*-cube routing makes use of only one path from source to destination even though several alternate paths may exist and hence the bandwidth utilization is poor.

The generalized *e*-cube algorithm cannot be applied to hypercycles directly due to the existence of cycles in each dimension. This can be circumvented by imposing constraints on the routing scheme. In Chapter 1, we presented a routing function that establishes at least one path of minimum length (greedy strategy) from a source to a destination node. Based on the shortest path strategy, we have introduced an *e*-cube based prioritized dimension routing algorithm [112] for a subset of hypercycles and proved that it is indeed deadlock-preventing. To do that, we utilize the notion of a dependency graph, which describes the interdependencies between the communicating processes that are formed in the network and show that it is acyclic. In the subsequent discussion, we include the analysis found in [112] and later extend it with the introduction of a prioritized mixed *e*-cube backtracking algorithm.

Definition 5.3 *Given a graph $\mathcal{G}$ on which a circuit switching routing is used, we denote by P_i the partially completed path between a source S_i and a destination D_i, by I_i the last node of the partial path P_i, and by L_i the set of all legal outgoing links from node I_i which can be used in order to complete the partial path to the destination D_i.*

Observe that if $L_i = \emptyset$, then $I_i = D_i$, i.e. the path is complete, while if $P_i = \emptyset$, then $I_i = S_i$, and the path has not commenced yet.

Definition 5.4 *With reference to a graph $\mathcal{G}$ on which circuit switching routing is used, we define the corresponding dependency graph $P = \{N, E\}$, where N is the set of partially completed paths, and $(P_i, P_j) \in E$ iff $\exists l \in L_i$ such that $l \in P_j$.*

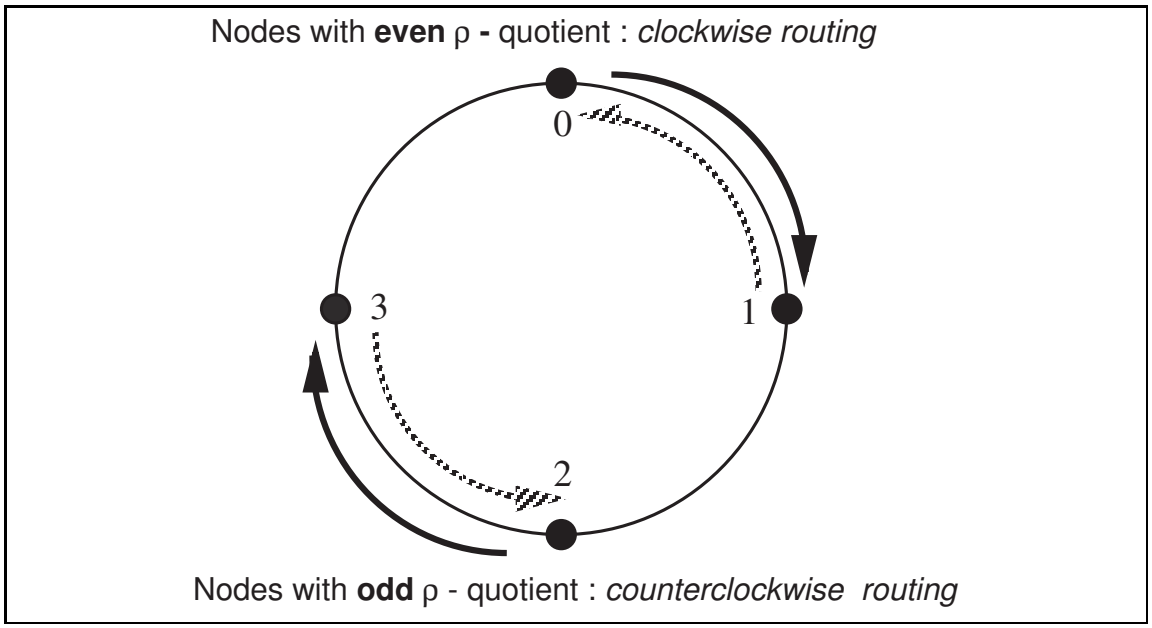


Figure 5.2 *Odd-Even preference routing scheme in* $\mathcal{G}_4^1$

Definition 5.5 *We say that a set of source-destination pairs* $(S_i, D_i); i = 1 \ldots \mu$ *is deadlocked if* $\forall l \in L_i, \exists j$ *such that* $l \in P_j$*;* $j = 1 \ldots \mu$*,* $j \neq i$ *where* L_i *and* P_i *are the sets of outgoing links and partial paths respectively.*

Stated alternatively, a set of source-destination pairs is deadlocked if all the links leading out of the formed partial paths are already contained in the other partial paths.

Lemma 5.1 *If a set of source-destination pairs* $(S_i, D_i); i = 1 \ldots \mu$ *is deadlocked, then the corresponding dependency graph contains at least one cycle.*

Proof. If there are no cycles in the dependency graph, then the graph possesses at least one terminal node P. Since P is a terminal node, there does not exist a succeeding partial path connected to P, i.e. $\forall l \in L \;\; \neg\exists j$ such that $l \in P_j$ for some j. But this contradicts Definition 5.5. ■

In order to achieve deadlock preventing routing, we introduce asymmetry in the way each node routes messages. Consider the example of a 4-node hypercycle, $\mathcal{G}_4^1$ depicted in Fig. 5.2. Node 2 can be reached from node 0 traveling either in a clockwise or counter clockwise direction. Similarly, node 3 can be reached from node 1 traversing the hypercycle in two directions. If the directionality is chosen at random, or identically for all nodes, a potential deadlock can occur. For instance, suppose that paths 012 and 210 are formed

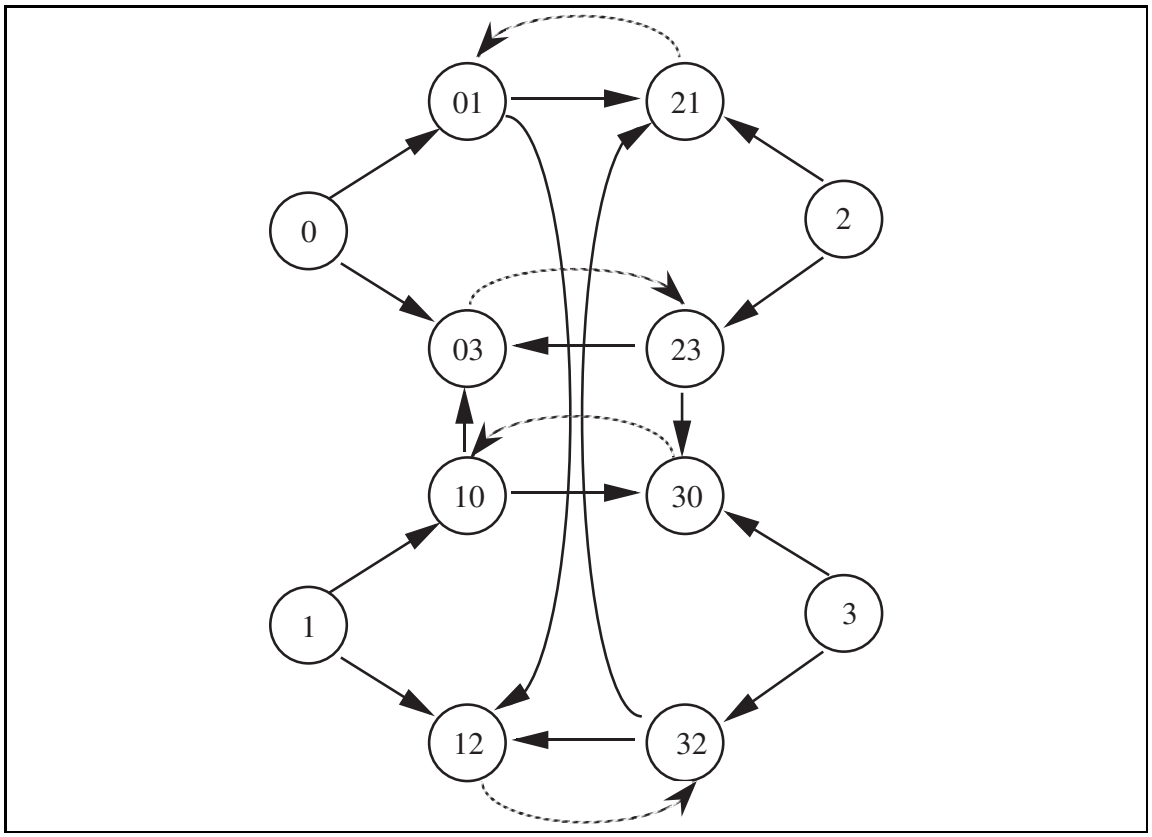


Figure 5.3 *Dependency graph for* $\mathcal{G}_4^1$

concurrently by two processes. Let the first process acquire the segment 01 while the second one occupies segment 21. Both paths meet at Node 1 and cyclically wait for each other to release the required resource *ad infinitum*. If on the other hand, the directionality is constrained by introducing asymmetry in routing, then deadlocks can be prevented as we shall prove subsequently.

Definition 5.6 *In a 1-dimensional hypercycle* $\mathcal{G}_m^\rho$*, where a destination node can be reached through two alternate routes i.e. when* $(\alpha-\beta) \bmod m = (\beta-\alpha) \bmod m = |\alpha, \beta|$*, where* α *is the source node and* β *is the destination node, we define the* odd/even preference *routing so as in the greedy mode, routes originating at source nodes with an even* ρ*-quotient*[‡]*, proceed in the clockwise (counterclockwise) direction, while routes originating at nodes with an odd* ρ*-quotient proceed in the counterclockwise (clockwise) direction. If* $\alpha*$ *denotes the next*

[‡]For an integer q, its ρ-quotient is defined as $\lfloor \frac{q}{\rho} \rfloor$.

intermediate node, then

$$\alpha^* = \begin{cases} (\alpha+\rho) \bmod m & \text{if } \left\lfloor \frac{\alpha}{\rho} \right\rfloor \text{ is even} \\ \\ (\alpha-\rho) \bmod m & \text{if } \left\lfloor \frac{\alpha}{\rho} \right\rfloor \text{ is odd} \end{cases} \tag{5.1}$$

Example: To illustrate the definitions stated above, we present the dependency graph in Fig. 5.3 for the odd/even routing in hypercycle $\mathcal{G}_4^1$. To make the representation of the dependency graph legible, we have omitted the nodes corresponding to paths 012, 103, 230 and 321. These nodes are terminal nodes and as such do not contribute to any possible loops. As it can be verified, this dependency graph is devoid of loops and thus the odd/even routing in $\mathcal{G}_4^1$ is deadlock preventing. If arbitrary routing were permitted, then the edges such as the one depicted by the dashed line in the Fig. 5.3 would be permitted, giving rise to loops and therefore possible deadlocks during routing. We state the following Theorem whose proof is detailed in [105, 112].

Theorem 5.1 *In a 1-dimensional hypercycle $\mathcal{G}_4^1$, the odd/even preference routing is deadlock preventing.*

In the subsequent treatment, we make use of the terms maximum links and link length. One dimensional hypercycles can be visualized as rings with chords. Thus, any node will have links connecting it to its two immediate neighbors lying to its left and right (i.e. to nodes $(\alpha \pm 1) \bmod m$) but also to more distant nodes through the chords (i.e. to nodes $(\alpha \pm \xi) \bmod m, \xi \leq \rho$). Links therefore can be differentiated as to whether they correspond to the chords of the ring or not.

Definition 5.7 *The length of a link connecting nodes α and $(\alpha \pm \xi) \bmod m$, $\xi \leq \rho$ in a one-dimensional hypercycle $\mathcal{G}_m^\rho$ is defined to be ξ. If $\xi = \rho$ then such a link is called a maximum link.*

Theorem 5.2 *A 1-dimensional hypercycle $\mathcal{G}_m^\rho$ with diameter 2 for which $\lfloor \frac{m}{2} \rfloor < 2\rho$ has a deadlock preventing routing scheme.*

The proof can be found in [112].

Theorem 5.3 *A 1-dimensional hypercycle $\mathcal{G}_m^\rho$ with $m = 4\rho$ has a deadlock preventing odd/even preference routing.*

Refer to [112] for the proof.

Remark: In essence, Theorem 5.3 establishes a class of 1-dimensional hypercycles of diameter 2 for which deadlock preventing routing can be established. Through this Theorem,

one can the nodes partition the nodes of $\mathcal{G}_m^\rho$ into ρ groups of four nodes, where each group is defined as follows.

$$g_j \quad = \quad \{i\rho + j; i = 0, 1, 2, 3\}, j = 0, \ldots, \rho - 1 \tag{5.2}$$

By mapping each of g_j to $\mathcal{G}_4^1$, deadlock-free routing can be accomplished based on the constrained odd-even routing scheme. These hypercycles can be distinguished by the fact that source-destination pairs are separated by a maximum distance of two. They can be reached either through a maximal-length link followed by a non-maximal-length, or if two maximal-length links are required for the path, then there exists one path which connect the source to destination without deadlocks.

Example: Consider the hypercycle $\mathcal{G}_8^2$ whose diameter is 2 since $m = 4\rho$. From the above theorem, we have $g_0 = \{0, 2, 4, 6\}$ and $g_1 = \{1, 3, 5, 7\}$. Nodes numbered 0, 1, 4, and 5, route on a clockwise orientation while nodes 2,3,6 and 7 route on a counterclockwise orientation. Also observe that both intra and inter group routing can be accomplished in one or two steps using odd-even routing with links of maximum length 1 or 2 as the case may be.

Theorem 5.4 *Fully connected graphs are deadlock free.*

Proof. Since the graph is fully connected, all partial paths between any source-destination pair have lengths at most one. Therefore, the corresponding dependency graph is devoid of cycles since if (P_i, P_j) is an edge in the dependency graph, then $P_i = \emptyset$, and thus, according to Definition 5.4, there cannot be another edge of the form (P_x, P_i) from some x. ■

Corollary 5.1 *A 1-dimensional hypercycles $\mathcal{G}_m^\rho$ with diameter 1 is deadlock-free under greedy routing.*

Proof. It is obvious that a hypercycle of diameter 1 is fully connected. Hence there can be no cycles in the precedence graph for greedy routing. ■

Corollary 5.2 *A 1-dimensional hypercycle $\mathcal{G}_m^\rho$ with $m = 4\rho - i$ where $i = 1, \ldots, 2\rho - 2$ is deadlock-free under greedy routing.*

Proof. Since $\lfloor \frac{m}{2} \rfloor = \left\lfloor \frac{4\rho - i}{2} \right\rfloor < 2\rho$ and the diameter k is 2 for $i = 1, \ldots 2\rho - 2$, Theorem 5.2 then applies. ■

Corollary 5.3 *A 1-dimensional hypercycle $\mathcal{G}_m^\rho$ with $m = 4\rho - i$ where $i = 2\rho - 1, 2\rho$ is deadlock-free under greedy routing.*

Proof. When $i = 2\rho - 1$ or 2ρ, the diameter k, is given by:

$$k = \left\lceil \frac{\lfloor \frac{4\rho - i}{2} \rfloor}{\rho} \right\rceil = \left\lceil \frac{\lfloor 2\rho - i/2 \rfloor}{\rho} \right\rceil = \left\lceil \frac{\rho}{\rho} \right\rceil = 1$$

and Theorem 5.1 then applies. ■

Theorem 5.5 *Generalized e-cube routing is deadlock preventing on a graph that is the product of graphs each of which possesses deadlock preventing routing.*

Proof. Assume that there is a cycle in the corresponding dependence graph with partial paths $P_i; i = 0, 1, 2, \ldots \mu - 1$, such that $P_i, L_i \neq \emptyset$ and $\forall i$, $\exists l_i \in L_i$ such that $l_i \in P_{(i+1) \bmod \mu}$. From the generalized e-cube routing algorithm, we know that $l_i \prec l_{(i+1) \bmod \mu}$, $i = 0 \ldots \mu - 1$, since a link to a partial path cannot be allocated unless all the required links at a lower or equal dimension (of higher priority) have been allocated. This implies that all the requested links must lie at the same dimension. Thus, one can form a cycle in the precedence graph, consisting of portions of the partial paths relevant to this dimension. But this is not possible, since we assumed that each of the component graphs in the product graph possesses a deadlock preventing routing. ■

5.1.1 E-cube Algorithm

Define the following precedence for dimensions $D_1, D_2, \ldots D_r$ for an r-dimensional hypercycle, $D_1 \prec D_2 \prec \ldots \prec D_r$. In other words, dimension 1 has the highest priority while dimension r has the least priority. The e-cube routing proceeds as follows: Routing from a source to destination node is accomplished in order of decreasing dimensions starting from D_i for $1, \ldots, r$. When a destination address is presented, the algorithm checks whether the destination has been reached in dimension D_i by comparing the source (current) and destination address bits. If it has not, then the next shortest path address in dimension i based on the greedy strategy is generated based on the odd-even preference scheme. The computed address is then validated and the message is forwarded to the next node; otherwise a wait signal is generated. On the other hand, if the destination has indeed been reached in the ith dimension, routing proceeds in a similar fashion in the remaining D_j dimensions (of lower priority) for $j = i + 1, \ldots, r$ till the source and destination address bits are equalized.

The deadlock preventing routing algorithm discussed above will form a part of a routing engine capable of being configured for a number of hypercycles and a variety of routing policies. The implementation of the routing engine will be the subject of discussion in Chapter 6. Oblivious protocols such as the e-cube make inefficient use of the bandwidth,

blocking when alternative paths are available. The main advantage is that they guarantee a connection at high loads and the ease of implementation. Adaptive routing protocols like backtracking strategies can use alternative paths in addition to providing resilience to failures. In the next section, we investigate whether the two policies could be merged.

5.2 Prioritized Mixed E-cube Backtracking Routing

One of the desirable feature of routing algorithms [113] is adaptivity, i.e., the ability of the messages to use multiple paths towards their destinations when congestion and faulty links are encountered. In [32], extensive simulation studies on the performance of a deadlock-free adaptive routing scheme called the *Backtrack-to-the-Origin-and-Retry routing* (BTOR) were made and a hardware component implemented in [114]. The throughput and delay characteristics of the BTOR scheme were established for several topologies of Hypercycles including binary n-cubes and these were compared with the e-cube routing algorithm for binary n-cubes. Though backtracking methods have high throughput for moderate loads, they suffer from excessive thrashing at heavy loads compared to e-cube. This can be attributed to the fact that the BTOR scheme uses alternate paths to the destination in contrast to the single path routing by the e-cube algorithm. Under *heavy* loads, the performance of the e-cube was better than the BTOR scheme. The reason for this behavior is that there is a high probability for a path being blocked under heavy loads when the messages have to be transmitted over a greater distance. This consequently results in frequent dissolution of the partial paths resulting in lower throughput.

Simulation studies of the backtracking and e-cube routing algorithms have shown that the performance of the backtracking scheme is better than e-cube for non-saturating (moderate) traffic. However, as the network starts saturating, longer paths are at a disadvantage resulting in lower throughput compared to e-cube. Network saturation occurs earlier in e-cube than the backtracking strategies but in saturation, the throughput of e-cube is better. Furthermore e-cube favours longer paths than backtracking schemes at higher loads since it is guaranteed to establish the connection as opposed to the heavy thrashing encountered by backtracking. Performance analysis of various random backtracking strategies on Hypercycles such as *backtracking to an intermediate node, persist backtracking* were reported in [105] and these studies indicate that random backtracking exhibits lower saturation but better performance than BTOR at saturating traffic.

Based on the above observation, it would be intuitive to combine the advantages of both e-cube and backtracking strategies into an algorithm called the PRIioritized Mixed

E-cube-backtracking Routing scheme (PRIMER) that we are proposing. In this scheme, the hypercycle can be partitioned into a fixed number of clusters such that routing across clusters follows the e-cube scheme while routing within each cluster can be either e-cube (provided it is permissible) or a backtracking strategy as the case may be. One metric for the choice of a routing policy within a cluster can be the following: Bigger clusters, having longer paths (or diameter) may be allowed to follow an e-cube routing while smaller cluster with shorter paths can be allowed to follow the backtracking strategy. By amalgamating these two policies, we can expect reasonably higher throughput and lower delay at different saturation levels compared to the individual schemes.

Theorem 5.6 *Backtracking algorithms that dissolve their paths are deadlock-preventing.*

Proof. Consider a cycle formed by a sequence of partial paths P_i, $i = 0 \ldots \mu - 1$. In the backtracking strategy, the partial path P_i between a source and destination pair (S_i, D_i) for some i is dissolved as soon as a block is encountered. The forwarding link $l \in L_i$ that may result in a possible deadlock is released and hence the occurrence of a cycle in the channel dependency graph is avoided. ■

As defined earlier, let the dimensions of an r-dimensional hypercycle have the following precedence, $D_1 \prec D_2 \prec \ldots \prec D_r$. Since an r-dimensional hypercycle can be expressed as a cartesian product of hypercycles, we can group the product hypercycles to form a second level of abstraction. For instance, let us consider n clusters of an r-dimensional hypercycle $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ with c_i product hypercycles each such that $r = \sum_{i=1}^{n} c_i$. then we can express

$$\mathcal{G}_{\vec{m}}^{\vec{\rho}} = \mathcal{G}_{\vec{m_1'}}^{\vec{\rho_1'}} \times \ldots \times \mathcal{G}_{\vec{m_i'}}^{\vec{\rho_i'}} \times \ldots \times \mathcal{G}_{\vec{m_n'}}^{\vec{\rho_n'}}, \qquad 1 \leq i \leq n \tag{5.3}$$

where

$$\mathcal{G}_{\vec{m_i'}}^{\vec{\rho_i'}} = \mathcal{G}_{m_{a_i}}^{\rho_{a_i}} \times \ldots \times \mathcal{G}_{m_{b_i}}^{\rho_{b_i}}, \quad 1 \leq i \leq n \tag{5.4}$$

such that

$$a_i = 1 + \sum_{j=1}^{i-1} c_j \tag{5.5}$$

$$b_i = \sum_{j=1}^{i} c_j \tag{5.6}$$

The variables a_i and b_i indicate the starting and finishing dimensions within cluster i. Let $D_1', D_2', \ldots, D_n'$ denote the dimensionality of the second level of abstraction with the precedence relation $D_1' \prec D_2' \prec \ldots \prec D_n'$. Define a deadlock-free routing strategy such that across the clusters in dimensions D', we employ e-cube routing while within the clusters

in $\mathcal{G}^{\vec{\rho_i'}}_{\vec{m_i'}}$, $1 \leq i \leq n$, there can be either e-cube or backtracking scheme. Then, we have the following corollary.

Corollary 5.4 *Prioritized mixed e-cube and backtrack routing is deadlock preventing on a r-dimensional hypercycle defined by Eqs. 5.3 to 5.6 where n is the number of clusters such that routing across dimensions D_i', $1 \leq i \leq n$ follows an e-cube scheme while each of $\mathcal{G}^{\vec{\rho_i'}}_{\vec{m_i'}}$ is closed under either e-cube or backtrack routing.*

Proof. From theorems 5.5 and 5.6. it is obvious that both backtracking and e-cube algorithms avoid/prevent deadlocks. Thus each hypercycle $\mathcal{G}^{\vec{\rho_i'}}_{\vec{m_i'}}$, for $1 \leq i \leq n$ is deadlock-preventing within dimension D_i'. Since the algorithm is prioritized across dimensions D_i', no cycles are formed in the dependency graph. ■

We can now formulate the PRIMER algorithm that mixes both e-cube and backtracking schemes as follows.

5.2.1 PRIMER Algorithm

Routing from a given source to destination node is accomplished in order of increasing dimensionality in D_i' as per the precedence relation outlined above for $i = 1, \ldots, n$. Additionally, if e-cube routing is used within each cluster then the precedence relation for D_i as stated earlier is valid within that cluster. When a destination address is presented, routing is performed depending on whether an e-cube or backtracking strategy is adopted within cluster D_i'. A check is made to determine whether the destination has been reached in dimension D_i' by comparing the source (current) and destination address bits within D_i'. If it has not, then the forwarding address in D_i' is generated based on the e-cube or backtracking schemes; otherwise a break signal within the cluster belonging to D_i' is generated if the scheme is backtracking or a wait signal if it is e-cube routing. Note that when a break signal is generated in dimension D_i' for a partial path corresponding to a backtracking strategy, the path is collapsed to the last valid point as intended by the algorithm in D_i'. For instance, if backtrack-to-the-origin-and-retry scheme is used, then the path is dissolved upto the last point in the previous dimension D_{i-1}'. If random backtracking is used, then the connection is collapsed to some arbitrary node in the already established path within D_{i-1}'. On the other hand, if the destination has indeed been reached in the D_i' dimension, routing proceeds in a similar fashion in the remaining D_j' dimensions for $j = i+1, \ldots, n$ till the source and destination address bits are matched.

Discussion

Let us consider the limiting cases of the PRIMER algorithm When $n = 1$, the prioritized mixed strategy reduces to a pure e-cube or backtracking scheme as the case may be. When $n = r$, if e-cube routing is implemented in each of the clusters provided it is allowable, then the PRIMER algorithm reduces to the generalized e-cube algorithm presented in Section 5.1.1. On the other hand, if backtracking is employed for $n = r$ clusters, then the PRIMER algorithm reduces to a pseudo e-cube policy.

The PRIMER algorithm combines the desirable features of both e-cube and backtracking strategies. It can be therefore customized depending on the application requirements, network traffic ranging from pure backtrack to generalized e-cube to pseudo e-cube with a blend of various backtracking schemes. It would be interesting to simulate the PRIMER algorithm and characterize its performance on hypercycles in addition to the implementation aspect for the future work.

5.3 Conclusion

In this chapter, we introduced two deadlock-preventing algorithms, namely, the generalized e-cube algorithm and a pseudo adaptive algorithm for hypercycles.

Chapter 6

Hardware Implementation

6.1 Introduction

Multicomputers consist of an ensemble of interconnected nodes where each node is a programmable computer with its own processor and local memory. The interconnection between nodes is achieved through point to point links. As mentioned in Chapter 1, communication efficiency is an important design factor in a multicomputer interconnection network.

When a message needs to be transmitted between a pair of nodes, a path is established using a sequence of point to point links via intermediate nodes to reach the final destination.

In the case of store-and-forward routing methodology, intermediate nodes take part in the actual routing process consuming computation and memory bandwidth thereby increasing the network latency. For instance in Ncube/10, the Axis operating system manages the routing of messages in a store-and-forward fashion.

Subsequent development in routing policies coupled with improvements in hardware technology brought the concept of the *direct-connect router* or *routing engine*.

The routing engine is responsible for decoding, forwarding and generally managing the necessary operations for establishing a source to destination path and avoiding deadlocks. For example in iPSc/2 and Ametek 2010, the routing engines [103], present at each intermediate node, close a series of switches to establish the path and messages are pipelined using the wormhole routing technique. Only the sending and receiving nodes participate in the actual routing while the intermediate nodes continue with their normal activities. Thus communication latency is significantly reduced by the hardware router.

Our primary focus is centered in the design of efficient routing engines for interconnection networks that incorporate circuit switching techniques. Such an engine should be amenable to hardware implementation supporting a wide gamut of routing policies.

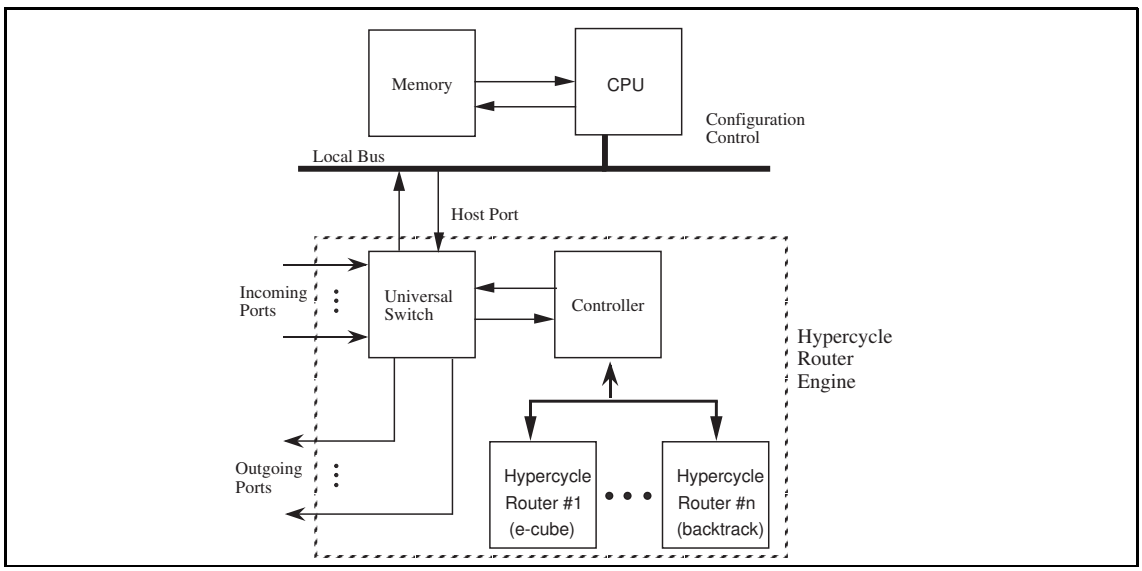


Figure 6.1 *Configuration of a hypercycle routing engine in a network node*

A general framework of a routing engine for circuit switching consists of the following components:

1. Controller
2. Universal switch (Crossbar)
3. Router.

The Controller provides the interface between the host, universal switch and the router and implements a chosen routing strategy. It intercepts messages arriving at the routing engine either from a local host or its neighbours. The universal switch is a cross-bar with incoming and outgoing ports connected to the various nodes of the network depending on the configuration. The controller extracts relevant information from the message header such as the destination address and passes it to the router module which will in turn compute the forwarding port[†] for a chosen routing policy. The controller then configures the switch to effect the prescribed connection and the message header continues its path. Figure 6.1 illustrates a routing engine specific to hypercycle networks. Observe that a number of different routers can be used that implement a variety of policies and the choice of the appropriate router can be made by the controller based on the traffic characteristics.

In Chapter 5, we presented the e-cube algorithm for a class of hypercycle networks that routes messages using a prioritized dimension approach and shown that it is deadlock

[†]A port is defined as an I/O link through which messages are sent/received at each node.

preventing. In this Chapter, we propose a design for a routing engine that incorporates e-cube routing and also demonstrate a practical implementation of its prime components, viz, the router and the controller. We shall first deal with the router.

6.2 Hypercycle Router

The basic function of the hypercycle router is to present a forwarding port address to the controller in order to route the message towards the destination. The router accepts as input the following parameters - the number of dimensions r, destination address, current address, population, connectivity and the number of available ports for routing. For a given routing policy (e-cube in our case) and a particular network configuration, the router checks for validity of the chosen policy. If it passes, a next port address is generated and compared with the available free ports. The controller is then informed of the status of the routing decision through the following output signals

1. NEXT PORT ADDRESS - To route messages
2. READY - Informs the availability of the next port address as soon as it is ready.
3. E-CUBE - If this line is asserted, it implies that the network conforms to the e-cube routing scheme.
4. DESTINATION REACHED - if the source and destination address bits match.
5. WAIT - This signal signifies a break condition during routing due to the nonavailability of the computed forwarding port.

6.2.1 Architecture

Based on the above discussion, a general architecture of an e-cuber based hypercycle router can be formulated as shown in Fig. 6.2. It consists of the following modules:

1. r modules of e-cube decoder.
2. r modules of Next Port Generators (NPG).
3. Port Selector and Validator.

To achieve higher throughput, parallelism has been introduced in the form of multiple e-cube decoders and Next Port Generators (one for each dimension). The e-cube decoders determine whether a given hypercycle configuration can perform e-cube routing or not. The NPG module computes the shortest path using odd-even preference, greedy routing [114] to the destination address based on the network configuration and current address. The

available ports which is an input to the router, is updated dynamically by the controller during each routing decision. The Port selector and Validator modules then determine a forwarding port based on the current available ports and the prioritized dimension algorithm discussed previously.

6.2.2 Implementation

The implementation of the router was targeted towards a high performance VLSI circuit since this module is a key component (specialized unit) in the routing engine. The Canadian Microelectronics Corporation's 1.2 μm CMOS technology was chosen as the medium for fabricating the design. The following assumptions were made on the structure of the router from the standpoint of available silicon real-estate, I/O pin count and ease of implementation.

1. The maximum dimensionality[‡] of the hypercycle topology is 4.
2. The maximum number of available ports (degree) is 16.
3. Each dimension can have a maximum population of 15 nodes.
4. The maximum connectivity (ρ_i) in each dimension is 5.

Figure 6.3 shows the major functional blocks of the router:

- Input Registers (Memory bank)
- Deadlock-Free Router Logic block
- Output registers
- Control block

The Memory bank consists of four 19-bit registers and one 16-bit register while the output register is 8-bit in length. The control block provides the necessary timing signals for latching the data into the input/output registers.

6.2.3 Data Path

Each NPG in the i^{th} dimension, $i = 1, \ldots, 4$ requires the following parameters, namely the connectivity vector, current address, destination address, population and offset for its operation. Each of the above inputs in dimension i is 4 bits in length except the connectivity vector which is a 3-bit number. Therefore, 4 register banks of size 19 bits each are required and they are accessed by addressing the respective registers as shown in Table 6.1. In addition, there is a 16-bit input register that stores the available ports with each bit

[‡]It was decided to have the number of dimensions and ports as a power of 2.

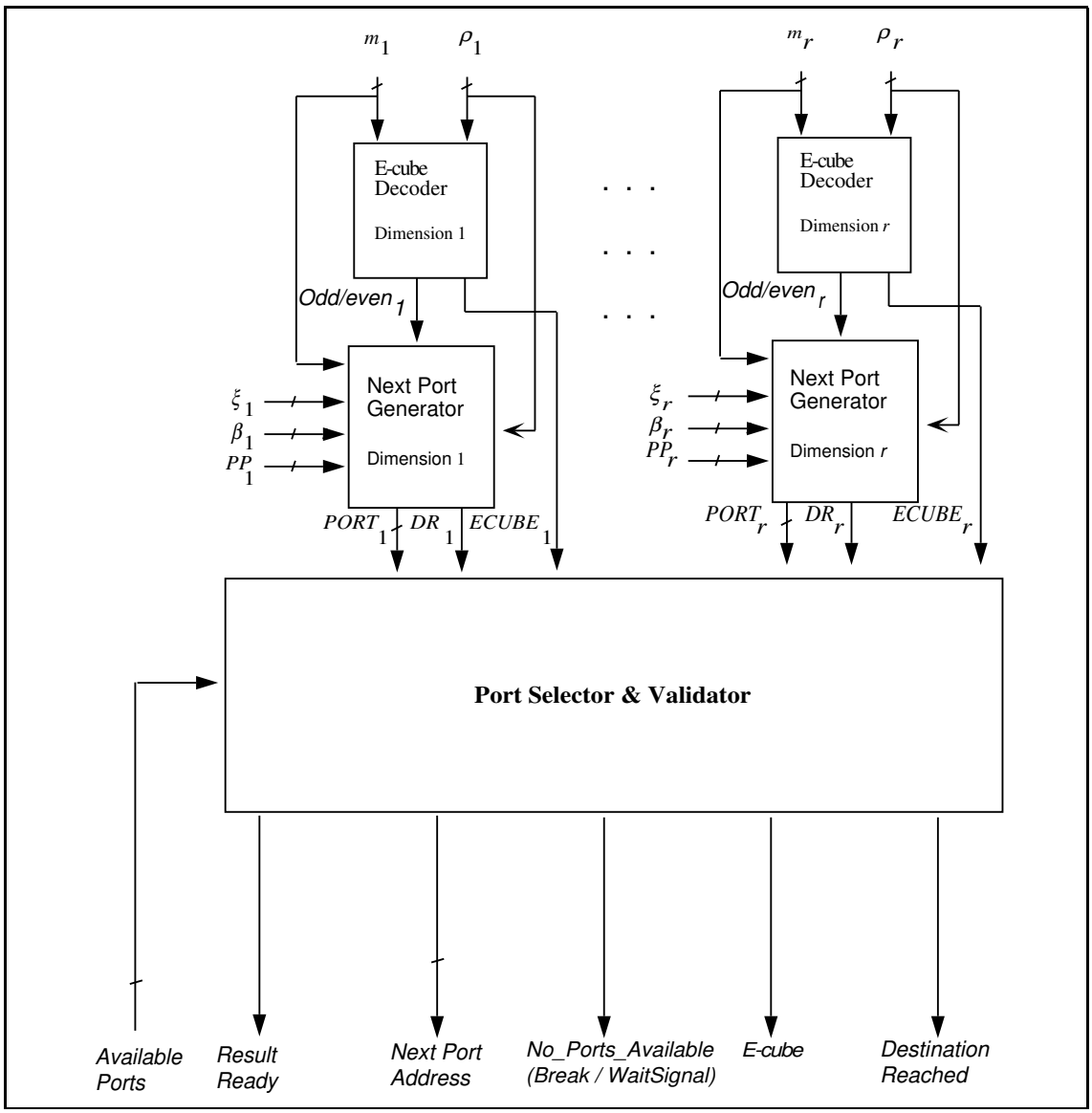


Figure 6.2 *Block diagram of the internal components of e-cube router*

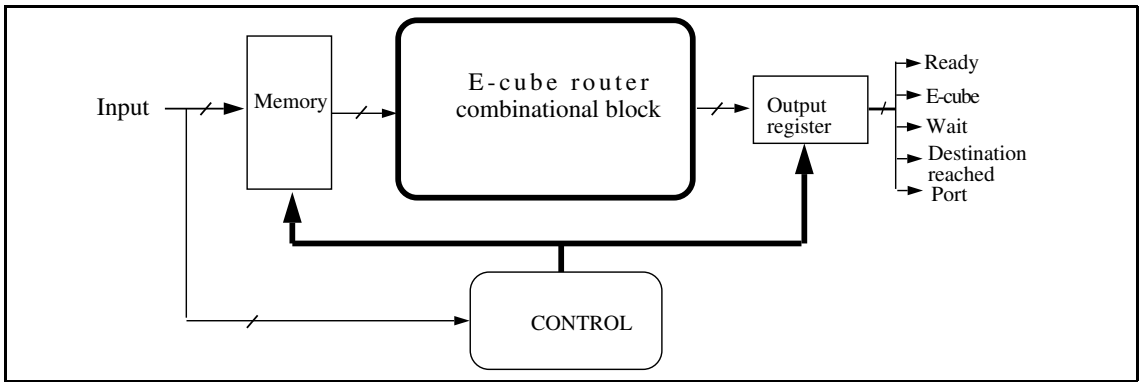


Figure 6.3 *Block diagram of deadlock-free router*

Table 6.1 *Register Designation*

Address	Parameter
0	Connectivity vector
1	Source/current address
2	Destination address
3	Population
4	Port offset of previous dimensions
5	Available ports

indicating whether the corresponding port is available for routing or not. The 9-bit output register latches the computed output port (5-bit result) along with the 4 status signals.

6.2.4 Operating Modes

The hypercycle router operates in the following phases:

1. *Configure phase*
2. *Load phase*
3. *Execute phase*

In the *Configure phase*, the router is initialized with data needed for its operation during system configuration. In this phase, the controller loads the router with information such as the connectivity, current address, population, and port offset. Note that the above data is needed only once during configuration.

In the *Load phase*, the destination address and available ports are loaded into the chip every time a new route is requested.

In the *Execute phase*, the data loaded into the router is processed and the resulting next port address (if an available one is found) together with the destination reached or possible break is generated. The router operates on a synchronous mode with the clock, providing a result every 4 clock cycles after the data has been loaded into the memory bank. It takes one clock cycle to load each register. The total execution time for each decision is therefore dictated by the time taken for the load cycle and the execute cycle. Note that the load cycle time is maximum during system power up when the six registers have to be initialized with the relevant parameters. During normal operation, it suffices to load the destination address and the available ports since these change dynamically. The load phase therefore takes 2 clock cycles to be initialized.

The *throughput* T_p of the router is defined as the number of decision cycles executed per second

$$\begin{aligned} T_p &= \frac{1}{T_l + T_e} \\ &= \frac{1}{2T + 4T} \\ &= F/6 \end{aligned} \tag{6.1}$$

where T_l and T_e are the load and execute phase delays. Thus, the effective throughput of the router is one sixth of the clock frequency.

6.2.5 VLSI Chip

The layout of the deadlock-free router in 1.2 μm CMOS technology and its floorplan are shown in Figures 6.4 and 6.5 respectively. It has been designed for a 4-dimensional network with a maximum degree of 16 spanning upto 50,625 nodes. The CAD tools used in the design process comprise of Cadence, SILOS, Verilog and Logic-III (UVic). The designed router contains 3828 logic gates using 19,906 transistors. and occupies an area of 4880 x 4080 sq. μm. The chip has 51 pins housed in a 68 pin PGA. The chip is synchronous and is designed to operate at a maximum frequency of 125 Mhz. It is fully programmable and checks whether a given configuration is admissible for e-cube routing. The chip has been designed based on Level Sensitive Scan Design (LSSD) DFT techniques [115]. Separate scan-in and scan-out pins have been provided to increase the controllability and observability of the combinational, control and memory modules.

6.2.6 Test results

The above deadlock-preventing router was fabricated by the Canadian Microelectronics Corporation using the Northern Telecom's CMOS4S process (MPC9206CA) in November 1992 and 10 devices were returned in March 1993. Out of 10 chips, 7 have passed fault and functional testing conducted on the IMS XL-60 Automatic test equipment.

Random testing was initially performed with 100 vectors to shortlist the non-working chips and 3 out of 10 devices failed. The CATPAG [116] software was used to generate vectors for the combinational part of the chip using the single stuck-at fault (SSF) model with a fault coverage of 97 %. Fault analysis was also conducted with LOGIC III (UVic) to corroborate the result. Only 795 vectors were required to achieve this fault coverage and these patterns were downloaded into the IMS resulting in 8747 test sequences. All the 7 working chips passed the test. The results of the fault testing are summarized in Table. 6.2. In addition, the control unit of the router was tested by scanning in test patterns and observing the required state changes/outputs.

Manufacturing/silicon defects in the die could be the possible reason for the failure of Chips 1,7 and 8.

A functional testing of the router with 1000 vectors was carried out to verify the correctness of the implemented algorithm and the 7 *good* chips passed the test. The AC characterization of the working chips was performed and a snapshot of timing diagram is presented in Fig. 6.6. As seen in Fig. 6.6, with a clock frequency of 60 Mhz, the network parameters such as the source, destination address, connectivity, and population are loaded

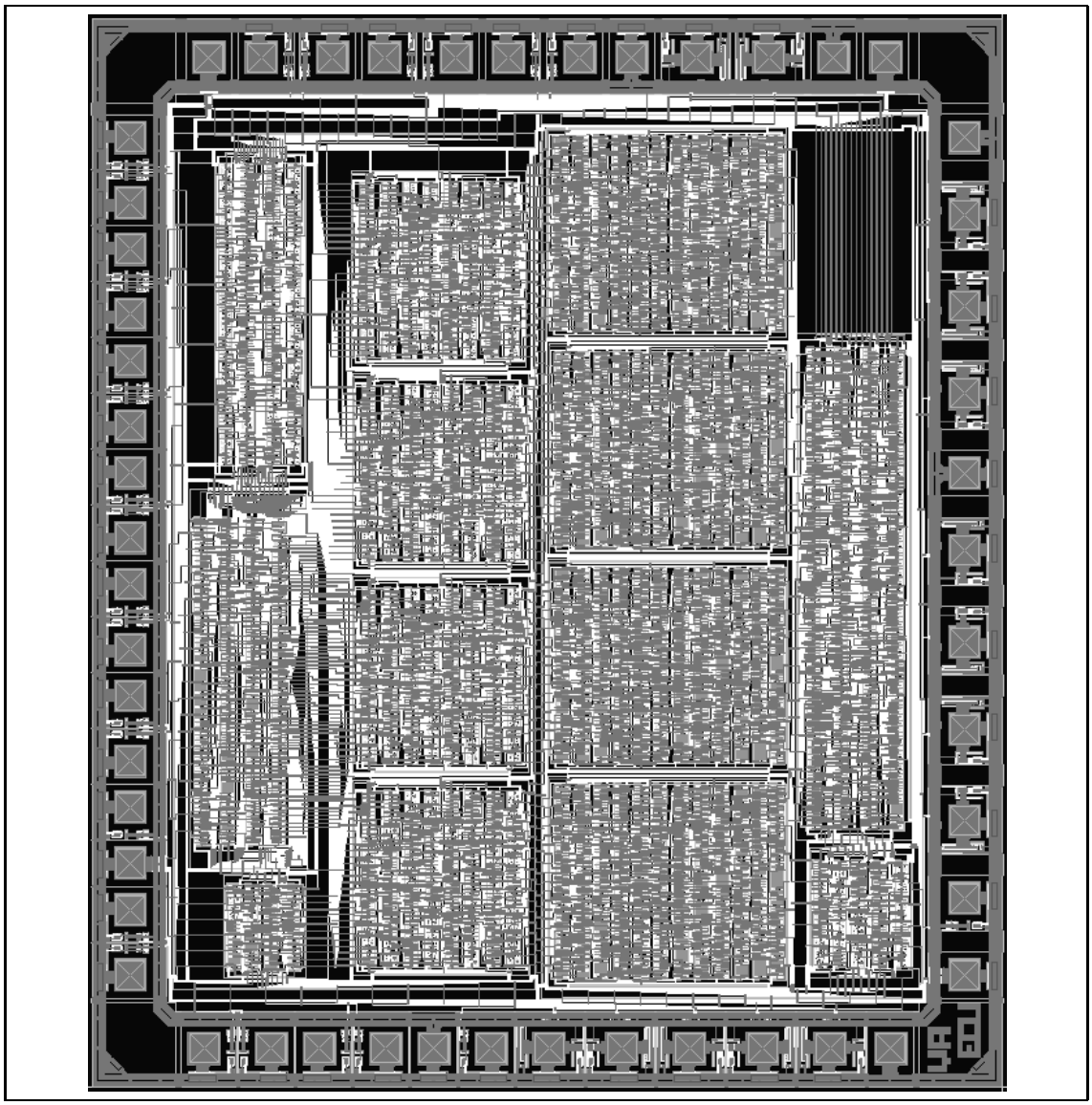


Figure 6.4 *Layout of the e-cube router chip designed in 1.2 μm CMOS technology*

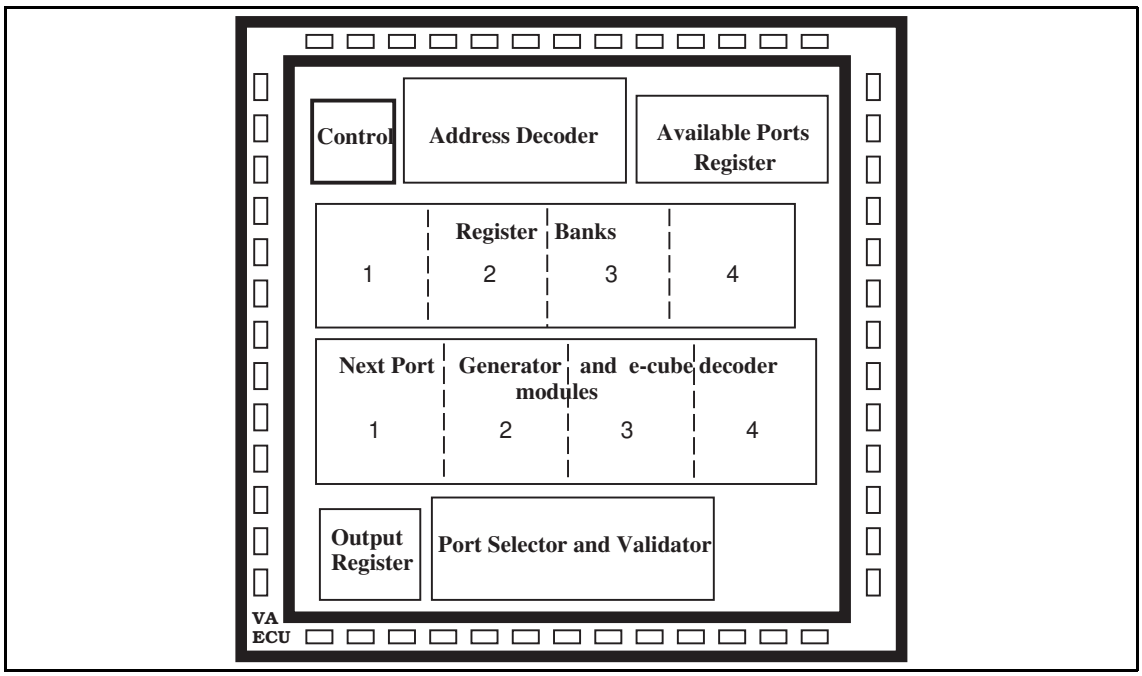


Figure 6.5 *Floorplan of the e-cube router chip*

Table 6.2 *Results of single stuck-at fault testing*

Fault Coverage - 97 % SSF		
No. of test sequences - 8747		
Chip No.	Comparison errors	Result
1	6568	Fail
2	0	Pass
3	0	Pass
4	0	Pass
5	0	Pass
6	0	Pass
7	1666	Fail
8	22	Fail
9	0	Pass
10	0	Pass

into the internal registers of the chip by setting the ADDRESS and DATAIN pin groups of the chip when EN is asserted. The propagation delay is defined as the time between the onset of the load phase and the time at which the computed result (PORT group) appears at the end of the execute phase when the RDY pin goes high. This delay was measured to be 78ns by programming the IMS tester with a $2 \times RZ$ format clock with a period of 10ns.

The output pads of the chip supplied by CMC as part of the CMOS4S [117] library incur a delay of approximately 16ns at a load capacitance of 120 pF. Since the outputs are latched, the load phase can be overlapped with the execute phase of the previous decision cycle with a commensurate decrease in latency to 62ns. The measured throughput of the router is therefore in excess of 16 Million decisions per second.

It is worth mentioning that timing simulation on the core circuitry (excluding the I/O pads) show that the chip could be operated on a 8ns clock period resulting in over 20 Million decisions per second. Due to hardware limitations in the IMS tester, the chip could not be tested near its rated speed.

6.3 Design of the Controller

In this section, we consider the next important component of the routing engine, namely, the controller module. The main function of the controller is to act as an interface between the network and the router/crossbar modules and deliberate the mechanism for establishing a path between a source to destination address. In other words, it schedules the various operations required for communication by importing, manipulating and exporting data from the routing engine.

For any routing policy, the controller must be able to import specific message types (i.e. message headers) extract pertinent information from these messages (e.g. the destination address), perform some computation, and elicit the appropriate "helper" modules (e.g. cross-bar, routers, encoders) as need arises by exporting to them specific message types that can be understood and utilized by these modules. The data manipulated by the controller is embedded in bit-streams of specific formats and structure. Since the structure of these data as well as the way that the controller imports the data (i.e. the protocol of interaction) is application specific, the objective of this work is to develop a general framework where the functionality of an arbitrary controller, which abides to the constraints outlined above, can be expressed and eventually embodied in hardware.

One approach to implement the controller is to use stored program microcontrollers but these are inefficient owing to their slow execution. Another solution is to implement the

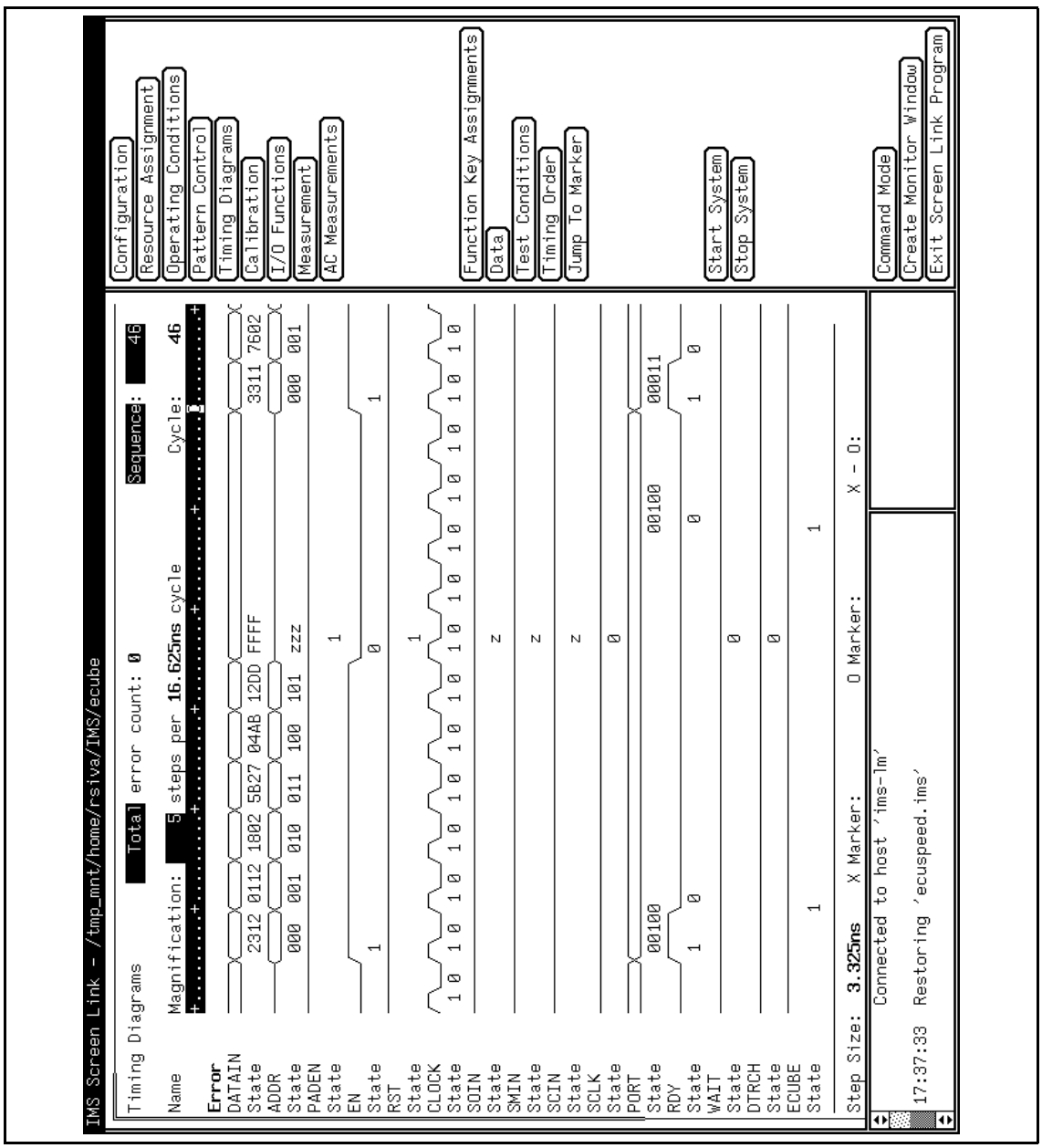


Figure 6.6 *Timing Diagram of the e-cube router chip*

controller in VLSI. This approach gives the best performance but is more expensive and has a large turn-around-time.

We take the view that the controller must be "programmable" in that it must be able to easily implement a new routing policy since the specifics are application dependent and avoid the costly custom one-of-a-kind VLSI design. Yet, it must perform better than the classical stored program controller.

Hence, the focus of this Chapter is to provide a framework where a routing engine can be specified from a high level description and targeted towards a Field Programmable Gate Array (FPGA) technology for implementation.

We first introduce a hardware description language specifically tailored to the problem of implementing routing engines. We then give an an example of how the language can be used to design a practical controller for the e-cube routing and discuss its performance.

6.3.1 Hardware Description Languages

In this section, we address the problem of creating a hardware description language for rapidly prototyping policy-specific controllers for interconnection networks. The use of Hardware Description Languages (HDL) for digital design has greatly increased in recent years. With chip densities and design complexities increasing rapidly, HDLs have created a vibrant impact in minimizing the turn-around-time and improving productivity of integrated circuits.

To this end, we have developed CoDeL (Controller Description Language) which is a hardware description language for designing sequential machines with special attention paid to the data abstraction and manipulation.

CoDeL is an algorithmic oriented language which can capture a high degree of abstraction and which can be compiled down to VHDL. The CoDeL compiler parses the source file and produces a synthesizable VHDL code which in turn can be directly mapped to any given technology such as Field Programmable Gate Arrays (FPGA) and Application Specific Integrated Circuits (ASIC). Using FPGAs as the medium of implementation, controllers can be reprogrammed for a gamut of routing policies as the application demands. Thus rapid prototyping is made possible along with chip use/reuse technology.

Hardware description languages, silicon compilation, and automatic synthesis techniques have been investigated and tools have been developed which can be used to specify/simulate and automatically synthesize a design. An HDL description uses a *formal language* to express the specification of a design.

Formal language specifications are soft compared to the the hard form of schematics. Text based descriptions are easy to read, debug and document. Apart from describing the system, HDLs are useful for simulation purposes and they can be synthesized into a gate level implementation with library components for a given technology. Hence, they are easily portable and rapid prototyping is possible. There are many factors involved in selecting and developing an HDL to suit a given purpose. Some of the earliest attempts in HDLs include A hardware Description Language (AHPL), Digital Systems Design Language (DDL) and Instruction Set Processor (ISP) [118]-[120]. AHPL is limited to simple boolean operations and does not have arithmetic operators. DDL assumes a Finite State Machine (FSM) model and converts a Register Transfer Level (RTL) code into boolean equations. It produces a technology dependent circuit which is 50%-60% larger than manually generated ones [119]. In ISP, the time sequence for the execution of the instructions is not implicitly derived and interfacing signals with the "outside world" is cumbersome [118]. Martin [121] and his group have developed techniques of translating concurrent programs into asynchronous VLSI chips. In 1987, VHDL [122] (Very high speed integrated circuits Hardware Description Language) became an IEEE standard and is considered as one of the premier languages for electronic design in industry to describe hardware from abstract to the concrete levels. Verilog [123] is yet another contemporary HDL competing with VHDL.

A controller in simple terms is a finite state machine that sequences through various states outputting specific signal values as determined by the control flow. In order to describe the actions of the routing controller in the case at hand, it is desirable that the specifications be at the algorithmic level requiring few structural details. Such an attempt would help a designer who conceives an algorithm to design a chip with little or no hardware expertise. In other words, our primary focus is towards expressing the controller's functionality and not deal with any architectural issues of the resulting design. As a result, we have developed CoDeL, an acronym for Controller Description Language. CoDeL is targeted towards the specification and design of controllers at the behavioral level.

CoDeL is a procedural language in which the statements are ordered implicitly representing the sequence of activities. It extracts the data and control flow from the program automatically, assigning the necessary hardware blocks and exploits inherent parallelism. It is simple and elegant for it provides basic program constructs in a succinct and readable form. It is quite similar to the C programming language and is therefore very easy to learn for a designer who is not familiar with the syntax and semantics of HDLs like VHDL and Verilog. CoDeL provides primitives, data structures and constructs for manipulating *ob-*

[*structure declarations*]

`module` *model_name*(*port declarations*)
{
  [*register declarations*]

  CoDeL *statements*

}

Figure 6.7 *Basic Structure of a* CoDeL *program*

jects at the behavioral/Register Transfer Level (RTL) and has been built with a library of I/O protocols that simplifies system interaction. Furthermore, it allows design changes to be easily made and allows for rapid silicon compilation to a given technology.

Though VHDL and Verilog are general-purpose languages that can address and manipulate objects at various levels of abstraction including behavioral, dataflow and structural descriptions, a synthesizable design must explicitly describe its architecture including the data and control paths. These architectural details are not of interest to a designer of an algorithm whose primary objective is to specify the controller's functionality at the highest level of abstraction.

In the following sections, we shall briefly present the salient features of CoDeL and present an example of a *e*-cube controller implemented in CoDeL.

6.3.2 CoDeL Syntax

The basic notation for describing the behaviour of a controller in CoDeL is illustrated in Fig 6.7. It comprises of the following parts which will be explained in detail.

- Structure declarations
- Module declaration
- Port declarations
- Register declarations
- CoDeL Statements

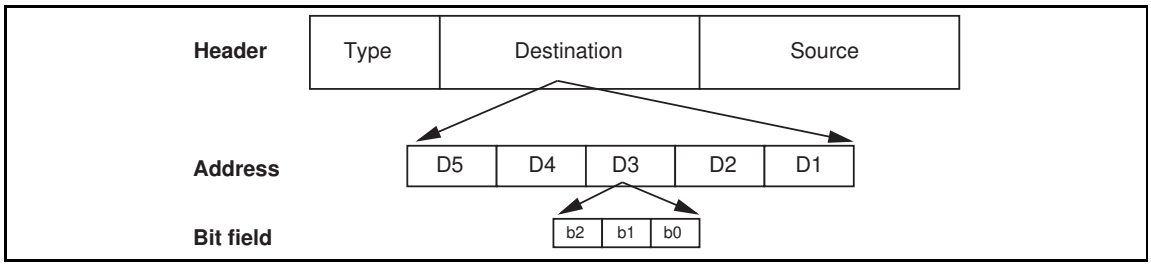


Figure 6.8 *A typical frame for a message header. Both destination and source addresses are expressed in mixed radix number formats*

6.3.2.1 Structure declarations

Analogous to the `struct` definition in C, we have introduced the `bitstruct` command in CoDeL for defining complex data structures. Data manipulated by the routing controller for instance can be organized into hierarchical structures called frames. Frames represent message headers or specific message types (e.g. connection-setup, connection-acknowledge etc). A frame is recursively defined to consist of frames or bitfields at the lowest level. An example of a header frame is given in Fig. 6.8. The example given in Fig. 6.9 represents a structure composed of 28 bits and corresponds to the header in Fig. 6.8.

6.3.2.2 Module declaration

Every CoDeL program should define the name of a module beginning with the keyword `module` and this will uniquely identify the top level circuit that is being synthesized. e.g. `module ecube_controller([port_declarations])`

6.3.2.3 Port declarations

A port is an abstract data type that permits I/O interaction between the module and the external components. Ports can be of type input or output[§]. A novel feature in the definition of ports is the introduction of protocols that can be associated with any arbitrary port. An example of a fully handshaking protocol is depicted in Fig 6.10. When the Data_Ready/ Strobe signal is asserted, it indicates that the data is valid and the recipient acknowledges after the data is latched by asserting the Acknowledge signal and the sender then negates its Data_Ready signal. The protocol is resolved with the negation of the Acknowledge signal by the receiver following which another cycle may commence.

[§]Bidirectional ports have not yet been introduced in the current implementation of CoDeL.

```
bitstruct field3
{
   (bits) b[3];
}
bitstruct field2
{
   (bits) b[2];
}
bitstruct address
{
   (field2) d1;
   (field2) d2;
   (field3) d3;
   (field3) d4;
   (field3) d5;
}
bitstruct header
{
   (address) source;
   (address) destination;
   (bits) type[2];
}
```

Figure 6.9 *An example of a frame and its associated bit structures*

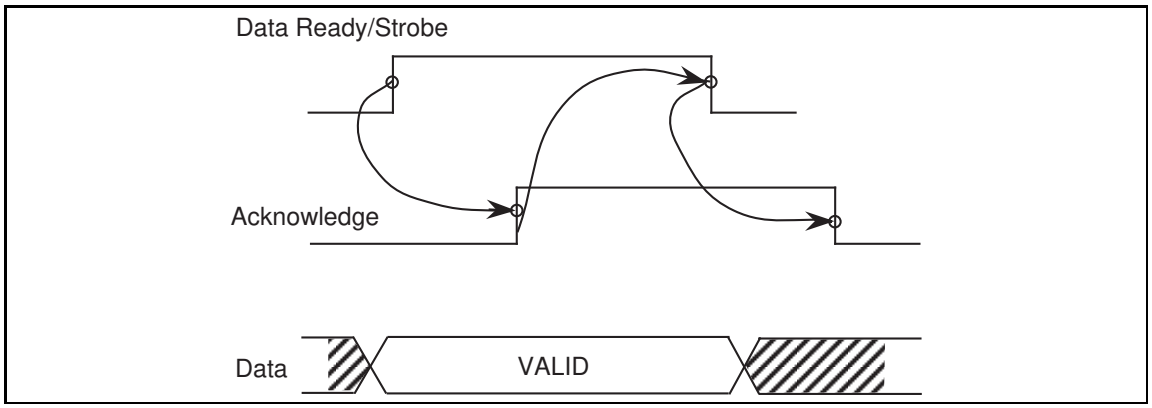


Figure 6.10 *A full handshaking protocol*

Consider the case of a routing controller where frames need to be imported from the external environment (e.g. the crossbar network, the "helper" modules or the host itself) to the controller. Similarly, frames need be exported. In both cases the controller needs to interact with external components that follow specific protocols like strobed, partial handshake, full handshake etc. Thus, an I/O operation is defined through a port and it specifies both the data path and the protocol to be followed. Once a port has been associated with a particular protocol, interactions around the port are completely transparent to the user. The syntax for port declarations in CoDeL is given below.

`in` (*bitstruct_name*) *port_name* [`with` *protocol_name*]
`out` (*bitstruct_name*) *port_name* [`with` *protocol_name*]

For an input port, the latch is transparent and the input signals become readily available to be polled at will. All registers associated with output ports are latched. Correspondingly, for the output port, the buffer is enabled continuously, and whatever is placed at the register appears at the output immediately.

Protocols are included in a standard protocol library with CoDeL. The general structure of a port is depicted in Fig. 6.11. The protocol engine is responsible for transferring the data from the environment to their respective latches or registers. It interacts with the control path of the controller through the standard ProtocolEngage and ProtocolTerminate signals. When protocols are used, the input and output ports are latched and this feature is hidden from the user. The respective Protocol engines are responsible, when engaged, to correctly latch the signals at the input port or enable the output buffers according to the protocol requirements. A NULL protocol (default) implies that there are no control signals and protocol engines present. The NULL protocol can be used by a designer who wishes

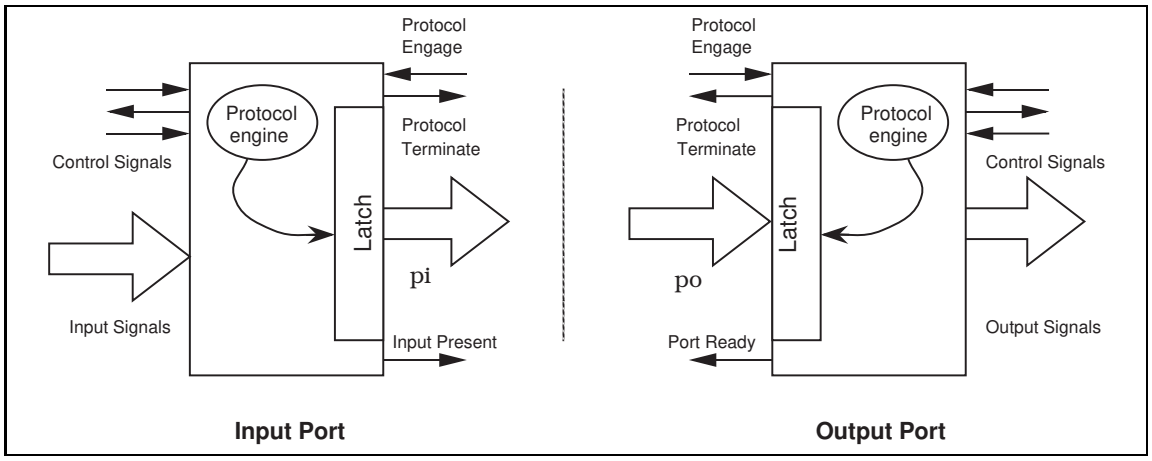


Figure 6.11 *Structure of an I/O protocol engine in* CoDeL

```
bitstruct mixed_radix_4          # Define a 16-bit address in 4-dim
{
(bits) field1[4];
(bits) field2[4];
(bits) field3[4];
(bits) field4[4];
}
bitstruct data_frame             # Define a 36-bit message header
{
(mixed_radix_4) source_address;
(mixed_radix_4) destn_address;
(bits) header[4];
}

in (data_frame) p1 with inp_hsk;
out (data_frame) p3 with out_hsk;
```

Figure 6.12 *Examples of port and protocol definitions*

to explicitly define the transfer mechanism. Protocol libraries can be developed easily by writing the specifications in CoDeL itself to account for the specific needs in the design. Examples of port definitions are given in Fig. 6.12.

Exporting/Importing data

Once a port is defined, it can be used in CoDeL expressions for not only exporting/importing data and also for checking whether the port which is of interest is available for use. To accomplish this objective, the following primitives have been provided in CoDeL

`input` (*port_name*)
`output` (*port_name*)
`isready` (*port_name*)

The `input`/`output`) primitives import/export data through the declared ports engaging the associated protocols, if any. The `isready` primitive is used to check (i) the availability of data in an input port or (ii) the status of an output port to accept data. We shall illustrate the above syntax and explain the significance of these functions through the following example using Fig. 6.12.

- `input(p1)` - engages the input protocol `inp_hsk` shown in Fig. 6.12 which latches valid data to the `p1` latch.
- `output(p3)` - the protocol `out_hsk` associated with output port `p3` enables the port's output buffer and data is exported.
- `isready(p1)` - used for testing for the existence of valid data at the specified port. This primitive can be used by the controller to determine whether there is data waiting at a port before the controller engages the protocol. It is also useful in avoiding unbounded delays in cases of asynchronous protocols and when data is not present.
- `isready(p3)` - tests whether the previous output operation has terminated. This is useful in asynchronous protocols for instance, to test whether the protocol engine has terminated the data transfer and is ready to accept the next set of data.

The input/output primitives defined above is mandatory for all ports that have protocols associated with them except for NULL protocols.

Table 6.3 *List of* CoDeL *operators*

`( ,..., )`	concatenation	`+ - *`	arithmetic
`> >= < <=`	relational		
`==`	logical equality	`!=`	logical inequality
`!`	logical negation	`&&`	logical and
`||`	logical or		
`~`	bit-wise negation	`&`	bit-wise and
`|`	bit-wise or	`^`	bit-wise xor
`~|`	bit-wise nor	`~&`	bit-wise nand
`<<`	left shift	`>>`	right shift
`<@`	left rotate	`@>`	right rotate
`? :`	conditional	`( : )`	bit-selector

6.3.2.4 Register declaration

Registers are declared as having a particular structure through a declaration of the form

`register` [(`bitstruct_name`)] `register_name;`

The default structure is bits and the components of a register can be addressed at the `bitstruct` level or at the bit level. We use a dot notation to identify the components under the convention that a sequence of dot separated fields represents a node in the tree representing the structure of the object and it consists of all the bits at the leaf nodes of the sub-tree rooted at the node. Consider the example of the bit structure *header* defined in Fig. 6.9. If we declare

`register` (*header*) `r1;`

`register r2[4];`

then the following are legal names and represent the corresponding bitfields.

1. `r1(5:0)` - The bits starting from position 5 (most significant) to position 0 (least significant) are addressed
2. `r1.destination.d5` - Represents the bits from 25^{th} to 23^{th} position (corresponding to the most significant field of the destination) are addressed.
3. Register `r2` is declared to be of size 4 bits.

6.3.2.5 CoDeL Statements

Assignment statements

The assignment statement is one of the important constructs in CoDeL that places the results of a computation to a register or an output port or portions thereof as discussed in Sections 6.3.2.3 to 6.3.2.4. The syntax of the assignment statement is

`register_name [output_port_name] = computation_expression`

In general, assignments manifest as an RCR (Register Combinational logic Register) implementation where data is stored in registers, operations are effected by a combinational circuit or a sequence of combinational circuits and the results are stored back in a register.

A `computation_expression` can have either input ports or registers as its operands. It is formed using a number of standard C operators that include addition, subtraction, multiplication, shifts and rotates, logical and bitwise logical operations. These follow the usual C associativity and precedence rules and are given in Table 6.3.

Automatic parallelization of non-dependent statements has been implemented in the parser by performing dependency analysis through dependency analysis [124]. This feature schedules multiple register assignments simultaneously and hence improves the efficiency and speed of the synthesized hardware.

Control Statements

One of the premises in developing CoDeL was to avoid the explicit description of the control path. Rather, the control path is automatically synthesized from the algorithm itself. In a sequential environment, the control path is inherently described by the order of the operations in a program. This is exactly the view taken in CoDeL. The control path of the design is extracted based on the sequentiality of the algorithm and it includes states which explicitly clock the registers included in the design. The basic control structures in CoDeL include loop, conditional and wait primitives with definitions that are reminiscent of C. These structures are listed below for clarity.

Loop primitive:

```
while (condition)
{
   CoDeL statements
}
```

Branch primitive:

```
if (condition)
{
   CoDeL statements
}
[ else
{
   CoDeL statements
} ]
```

where the `else` statement is optional.

Wait statement:

```
wait (condition)
```

The wait primitive maintains the current state of the design until a certain condition is satisfied. It is used mainly for synchronizing with external signals.

6.3.3 CoDeL Compiler

The CoDeL compiler has been written in C and produces a synthesizable VHDL code as its output. The compiler consists of two main subprograms, namely, the *parser* and *VHDL builder*. The parser analyzes the CoDeL program and extracts pertinent information such as variables/tokens which is used by the VHDL builder for generating the output. The parser consisting of 3500 lines of C code uses the recursive descent technique with single token lookahead [125] and has been written by Mr. V. V. Dimakopoulos.

The VHDL builder part of the CoDeL compiler was coded by the author and roughly consists of 3070 lines of C code. The methodology used is the Moore's approach [126] for describing a synchronous finite state machine whose control flow is extracted by the parser from the user program. The VHDL [127] construct, PROCESS() is used extensively for describing the register assignments and state transitions in addition to the computation of arithmetic/logical operations. Other VHDL constructs used in the program include the BLOCK statement for initialization of signals and for performing I/O operations. The COMPONENT construct is used for instantiating the protocol libraries and integrating with the main program.

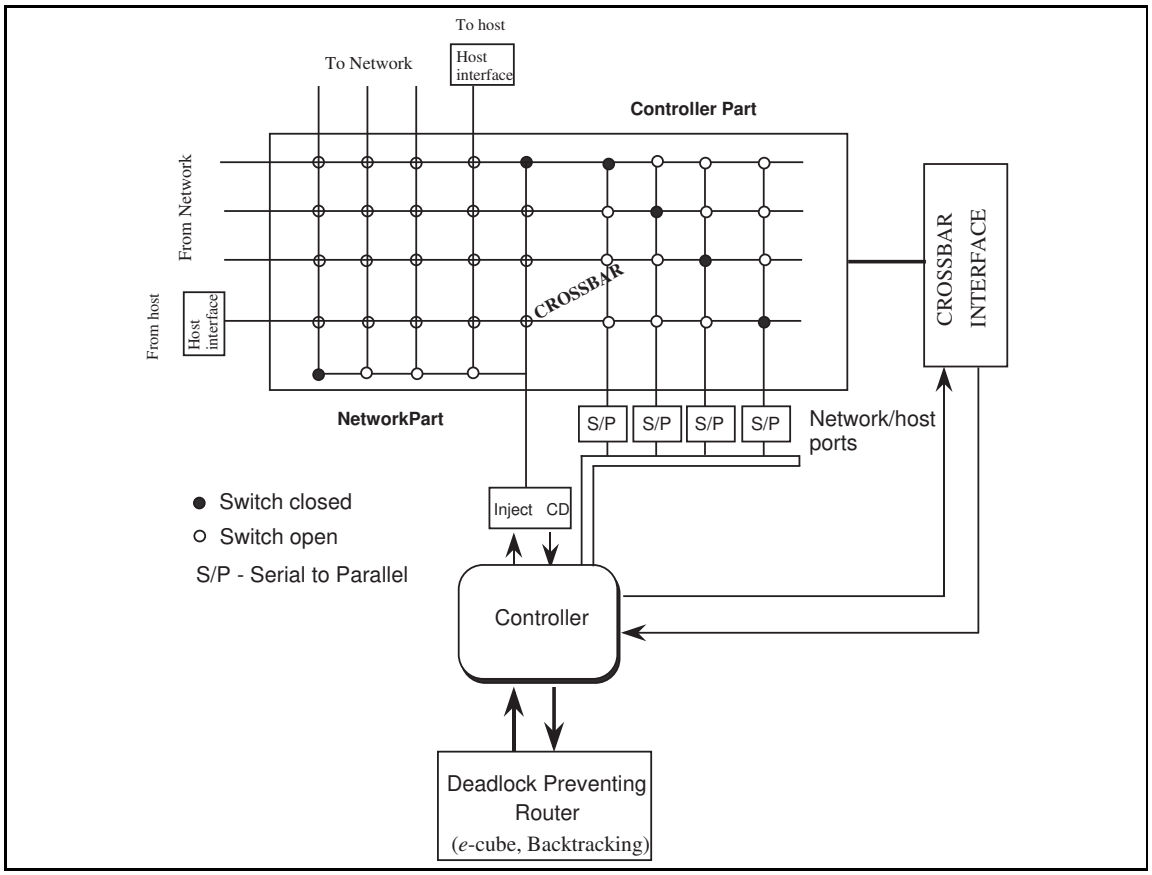


Figure 6.13 *Structure of a circuit switching routing node in a hypercycle network*

The project began on February 15^{th}, 1995 and the core of the CoDeL compiler was completed by March 31^{st}, 1995. The current version of CoDeL is v0.9 and is targeted towards the Mentor Graphics System 1076 VHDL language [128]. It should be mentioned that different Computer Aided Design (CAD) tool vendors (eg. Mentor, Synopsis, Compass) producing synthesis software packages have their own subset of synthesizable VHDL constructs/paradigms and hence the resulting VHDL code cannot be readily ported from one system to the other. However, our design of the CoDeL compiler is such that the parser and VHDL builder have been made independent of each other and hence by incorporating minimal changes to the VHDL builder program as necessitated, portability can be readily achieved.

6.4 A Routing Engine Controller Implementation

In this section, we present the design of an e-cube controller used in a circuit-switching environment node and discuss its implementation using a CoDeL program. The overall architecture of the node is presented in Fig. 6.13. It is assumed that the communication is bit-serial, and that there are bidirectional serial links between nodes. As the circuit is formed, both links are allocated. One is used for forwarding the message, the other is for acknowledgments. Furthermore, Serial/Parallel (S/P) interfaces are assumed at each link and these present message streams to the controller for a routing decision.

The crossbar is assumed to be able to switch the bit-serial streams and to be composed of intelligent switches which are capable of detecting a break on the bit-stream. The detection of the break causes the switch to open and disconnect. It is also assumed that the switches in the crossbar can be randomly accessed (tested and set) through the crossbar interface. The router component implements a generalized deadlock-free e-cube routing as discussed earlier in this chapter and returns a "network port" through which the circuit is to be continued towards the destination. It is the responsibility of the controller to ascertain that the required "network port" is available and continue the circuit. If the "network port" returned by the router is free, then the circuit can be extended. The controller uses the address of the "network port" and sets the appropriate switches in the crossbar so that it can *inject* the header of the message on both directions of the chosen link. At the same time, the collision detection hardware that is incorporated in the injector detects any concurrent attempt by the recipient node to use the same link. Upon detection of a collision, the controller implements a collision resolution strategy based on the lexicographical ordering of the nodes of the network. If no collision is detected, then the controller sets the crossbar switches so that the circuit is extended and marks the "network port" as being used.

The cycle continues with the controller sequentially testing all network or host ports for incoming traffic, and the status of the switches that implement the currently active circuits. If a closed switch has been opened because of a break (i.e. termination of the circuit) then the controller updates its internal state of the ports to reflect that the previously used port is now free and continues the cycle. It is assumed that the switches at the network part of the crossbar are paired with switches at the controller part so that once the circuit is dissolved, the incoming link is connected to its corresponding port while the outgoing link is disconnected. Any new messages just arriving, will be captured by the ports and initiate a new routing cycle by the controller. The above discussion is summarized in the form of a pseudo code given in Fig 6.14 whose implementation will be discussed in the next section.

```
• Configuration Phase:
    Initialize and generate appropriate control signals for crossbar and router.
• Normal Operation:
    Repeat for each input port i, i= 1 to maximum_available_ports
        if port i is ready then
            -Import message header.
            -Extract Destination address, Available ports and send to router.
            -Import forwarding port and status signals from router
            -Analyze status signal to determine if destination has been reached.
            if destination is not reached then
                -Configure cross bar and export message header to next node.
                -Wait until receiving node acknowledges or a collision signal
                    is generated.
                if no collision then
                    -Clear previous cross bar connection.
                    -Set crossbar to connect the previous node to the
                        succeeding node for data transmission.
                else
                    -Invoke collision resolution strategy.
                end if
            else
                -Send message header to local host if it is free since
                    destination has been reached.
            end if
        end if
    end repeat
```

Figure 6.14 *Pseudo code of the e-cube controller algorithm*

6.4.0.1 Controller Implementation

The control algorithm for the above e-cube routing policy described earlier has been coded in CoDeL for a 4-dimensional network with a maximum degree 16 and 15 nodes per dimension. The resultant VHDL code was synthesized using Mentor's Autologic [126] and targeted towards a XILINX 4010PG191 chip with 158 Input/Output (I/O) pins. The algorithm uses polling to test for availability of data at each of the S/P interfaces in a round-robin fashion. If it has valid data, the header is imported, the destination address is extracted and sent to the router "helper module", and then it inputs the resulting "network port" through which the route will be extended. We assumed full handshake protocols to interface with the crossbar module and S/P ports and that these helper modules operate faster than the controller. Additionally, it was assumed that the injection module/collision detection (CD) circuitry communicated only with the "network ports" connected to outgoing nodes for the sake of simplicity in the cross bar interface. It was also assumed that when a collision occurs, the messages at either end would wait till the polling mechanism services them again.

Functional simulation was performed to verify the correctness of the VHDL code and the underlying hardware. For this particular example, we assumed a message header consisting of a 4-bit message type and two 16-bit addresses for the source and destination addresses. Also, it was assumed that 17 serial-to-parallel ports (including the host) present messages to the controller for routing decisions.

In order to verify the actual performance in hardware, the algorithm was adapted for a smaller 2-dimensional network of degree 4 with a maximum of 3 nodes per dimension. This substantially reduced the I/O pin count and gate complexity making it amenable for downloading to a Xilinx 4010PC84 chip that was at our disposal. The source CoDeL program for this design is listed in Appendix E. The size of the source code is 156 lines which was translated into a 1665 line Mentor system 1076 VHDL code. The control path of the implementation of the main machine consists of 78 states of which it takes 25 clock cycles for the initialization, 20 cycles for interaction with the router module, 21 cycles for transmitting the message assuming there is no collision or a block and 12 cycles to reach the final destination. The protocol engines have a maximum of 9 states for their interaction with the main machine.

The code has been simulated for functional correctness using Mentor's Quicksim's tool and the design was synthesized and downloaded to a Xilinx 4010 PC84-5 chip[¶]. The synthesized design consists of 989 Xilinx primitive cells and some statistics of the area usage

[¶]The chip has 84 pins packed in a Plastic Leaded Chip Carrier (PLCC) with a speed grade of -5.

Table 6.4 *Statistics of e-cube controller implementation*

Xilinx 4010PC84-5 FPGA			
Note: CLB denotes Combinational Logic Block			
Resources	Used	Available	Utilization
Occupied CLBs	310	400	77%
Packed CLBs	187	400	46%
I/O Pins	60	61	98%
CLB Flip Flops	117	800	14%

is summarized in Table 6.4. The chip was tested using an HP16500A Logic Analyzer at a maximum clock frequency of 25 Mhz. This was the threshold of the Logic Analyzer's capability to generate test patterns. At this clock speed and assuming the crossbar and S/P devices have a 40ns delay in responding to I/O requests, the synthesized design provides commendable performance with a delay of approximately $3\mu s$ in routing a message from its arrival on an input port, setting up the crossbar and forwarding the message to the next node. The minimum clock period for 4010 chips is 9ns as given in the Xilinx manual [129]. Accounting for the set-up time, the delay of the interconnect, I/O pads and factoring in the combinational delay incurred between state transitions, a reasonable estimate is that the *e*-cube controller could be clocked at a speed of 30-40 Mhz achieving a sub $2\mu s$ delay in routing. At the time of writing this dissertation, Mentor's timing analyzer was not functioning properly due to setup and software problems. The availability of a more sophisticated automatic test equipment would substantially help to accurately characterize the controller.

6.5 Conclusion

The advent of FPGAs has paved the way for developing and testing hardware circuits *in-house*. Re-programmability is a major advantage of FPGAs in terms of reduced cost and less turn-around-time. In this Chapter, we have formulated a general purpose hardware description language that provides elementary operations, data structures and simplified I/O operations for describing controllers at the behavioral/RTL level. Rapid prototyping and easy programmability are the two main features of CoDeL. In the example at hand, the routing engine controller can be configured for a variety of routing policies to suit the application at hand by describing its behaviour in CoDeL.

CoDeL, in our opinion, is a compact language which, even though it does not offer the versatility and generality of VHDL, it allows us to abstract the higher level functional description of the controller's algorithm. It has a rich repertoire of operations for efficient data manipulation and its object-oriented features render structured protocols and I/O functions transparent to the user. It automatically extracts the necessary control paths from the algorithm relieving the user from structural and implementation details. CoDeL thus achieves fast top-down silicon compilation at a low cost.

A rigorous characterization and validation of the afore mentioned Xilinx implementations needs to be carried out in the future. In addition, optimization of the compiler, for instance increasing the parallelization of CoDeL statements, minimization of redundant states and registers, introduction of bidirectional ports, new operators, functions and procedures, loop indices and asynchronous clocking schemes, multi-module integration are some of the areas that need improvement and further work. It is our expectation that these features will enhance the specification of more complex routing polices and produce efficient circuits.

Chapter 7

Conclusion

The evolution of multicomputer systems is a by-product of the advances in computer technology that was brought about demands for improved system performance. One of the important considerations in the design of multicomputer systems is the interconnection network and the associated problem of finding and implementing efficient routing algorithms.

7.1 Synopsis

In this dissertation, we have focussed on hypercycles, a novel class of multidimensional graphs for modeling interconnection networks. These are generalizations of popular interconnection networks such as the ring, torus, hypercubes, generalized hypercubes and the complete graph. Hypercycles exhibit a regular, cyclic and with simple routing schemes and their wide topological flexibility makes them attractive for wide spectrum of applications. We have addressed some of the architectural and communication issues on hypercycles, namely topological optimization, graph embeddings and the design & implementation of deadlock-free routing algorithms.

In Chapter 1, the evolution of parallel computers is traced and the need for efficient interconnection networks for multiprocessing system was established. Based on this, the problems for investigation were defined. A brief overview of the basic terminologies of hypercycle graphs was also given.

In Chapter 2, topological optimization was conducted on hypercycles with respect to the average distance, diameter and product of degree*diameter. The main conclusion drawn from the optimization study is that balanced hypercycles are not necessarily optimal structures.

In Chapter 3, we presented an efficient algorithm for embedding complete binary trees with unit dilation, and unit load factor in a hypercycle network having an asymptotic expansion of 1.25. The optimal hypercycle is identified using numerical techniques and is shown to be cost-effective compared to unit dilation embeddings in hypercubes.

In Chapter 4, a generalized m-ary gray code scheme for generating a Hamiltonian cycle was proposed and expressions for the ranking and unranking of Hamiltonian cycles were derived. In addition, a short study on Hamiltonian decomposition was also made.

Chapter 5 considered the issue of designing deadlock-free routing algorithms for hypercycles. The deterministic e-cube routing algorithm is shown to be deadlock preventing for a subset of hypercycles. In addition, an algorithm which amalgamates backtracking and e-cube algorithms called the the PRIMER has also been proposed.

Chapter 6 addressed the system architecture and hardware implementation aspects of a routing engine consisting of a specialized router, cross bar and controller, is responsible for intercepting messages and establishing a connection between a source to destination. A router component that incorporates deadlock-free e-cube algorithm for was designed for a 4-dimensional network and fabricated in 1.2 μm CMOS technology. The chip has a measured throughput in excess of 15 MDPS. A generalized framework for designing controllers was evolved by the development of CoDeL, a novel hardware description language for designing synchronous sequential machines and a prototype e-cube controller was synthesized and tested in hardware. CoDeL provides basic computational operations and a higher level of data abstraction with powerful structured I/O interaction not to mention its simplicity, user-friendliness and similarity to C. It is hoped that CoDeL will open new vistas in mass prototyping of controllers along with use/reuse technology reducing design cycle time and costs significantly.

7.2 Suggestions for Further Research

Some of the open and unsolved problems that have surfaced from this research include the following.

- Obtaining graph theoretical bounds for hypercycle structures that can embed a complete binary tree with unit dilation. This will provide a quantitative measure to judge the structural efficiency of the hypercycle obtained through optimization in Chapter 3.
- Constructing algorithms for embedding arbitrary trees and other computational structures in hypercycles.

- Enumerating the number of Hamiltonian cycles in hypercycles.
- In-depth analysis of the Hamiltonian decomposition in hypercycles.

Apart from these recommendations, there are several pertinent issues on hypercycle networks that warrant further research and development.

Parallel Controller:

The present implementation of the controller for e-cube routing uses a simple, round-robin algorithm to sample the input ports and effect routing decisions one at a time. To speed-up scheduling, concurrent processing of input ports may be brought about by overlapping I/O operations that do not have dependencies. For instance, when the controller awaits the acknowledge signal from the inject module for a given message, it can instruct the router to issue a new forwarding port for a different message. These operations are complex and require extensive resource management, buffering and dependency analysis.

Assuming that such a parallel scheduler can be constructed and given a much faster FPGA technology, the throughput of the controller can be made to match the router module if not more. Efforts can then be directed to increase the performance of the router component as well. Ideas for speed-up in the router such as the introduction of multiple NPG units have already been incorporated. To achieve further speed, improvements to the architecture of the router can be brought about in the form of pipelining. Some work in this respect has been done by the author by decomposing the Next Port Generator as a 3 stage-pipeline and synthesizing the schematic from a VHDL description. However, one possible bottleneck that would have to be considered for the router is the number of *available ports* which gets updated dynamically in the Port Validator module during each decision and hence will affect successive entries in the pipeline. It remains to be seen how efficiently pipelining can be achieved in this case.

The CoDeL compiler also needs to be geared and tuned up for parallel scheduling. Providing parallel constructs in the language such as coprocess statements, fork and join primitives and other innovations will simplify the user's task. This aspect of the language along with the suggestions given at the end of Chapter 6 is open for further investigation and would be an interesting project to pursue.

The router component has been isolated from the controller and other parts of the hypercycle router engine in view of the fact that it can be used in conjunction with other routing schemes and from the point of view of speed/area considerations and programmability of the controller. One of the deadlock-free routing strategies proposed in this dissertation, namely the PRIMER algorithm which is a cross between backtracking and e-cube can be analyzed

in depth and its performance in terms of throughput and delay should be validated by simulation. Hardware implementation of the PRIMER algorithm is also a worthy proposition. Our primary attention has been focussed on developing and implementing circuit switching routing techniques for hypercycles. Worm-hole routing is another popular mechanism of contemporary interest and its adaptation to hypercycles will be an interesting exercise. Much of the remaining work will be directed towards placing the various components of the router engine under one ensemble (board level design) and making a prototype targeted for a specific application.

Broadcasting is an important communication protocol in message passing concurrent computers [130]. Some specific issues in broadcasting include

- All-to-All broadcasting: Every node sends the same message to all other nodes.
- Total exchange: Every node sends a unique message to every other nodes.

All-to-All broadcasting has been solved for 1-dimensional hypercycles [131] while an optimal solution for the total exchange in linear arrays is given in [132]. However, for multi-dimensional hypercycles, finding optimal algorithms for the above is open for investigation.

Fault tolerance, load balancing and task allocation in hypercycles are other broad areas for further research.

In fine, it is our conclusion that this research work will contribute significantly to the field of parallel & distributed computing and a practical realization of a hypercycle based multi-computer system can be envisioned for the future.

Bibliography

[1] Digital Equipment Corporation, *Alpha: A technology foundation for the 21st century*, DEC News Release, Order No 01754-2571, 1995.

[2] K. Hwang and A. Briggs, *Computer Architecture and Parallel Processing*, McGraw Hill series, 1987.

[3] X. Lin, P. K. McKinley, and L. M. Ni, "Deadlock-Free Multicast Wormhole routing in 2-D Mesh Multicomputers", *IEEE Trans. Comput.*, vol. 5, No. 8, pp. 793-804, Aug. 1994.

[4] A. L. Leiner, W. A. Notz et al., "PILOT - A new multiple computer system", *Journal of ACM*, vol. 6. No. 3, pp. 313-335, July 1959.

[5] G. H. Barnes et al., "The ILLIAC IV computer", *IEEE Trans. Comput.*, vol. C-17, pp. 144-151, Aug. 1968.

[6] D. P. Siewiorek, V. Kini, H. Mashburn, S. R. McConnel and M. M. Tsao, "A Case study of C.mmp, C.m* and C.vmp - Part I - Experiences with fault-tolerance in multiprocessor systems", *Proc. IEEE*, vol. 66, No. 10, Oct. 1978, pp. 1178-1199.

[7] K. Hwang and D. Degroots, *Parallel Processing for Supercomputers and Artificial Intelligence*, McGraw Hill series, 1989.

[8] Digital Equipment Corporation, *High Performance Computing*, Technical Report, DEC, Order No 01754-2571, 1995.

[9] J. P. Hayes, T. N. Mudge, Q. F. Stout et al., "Architecture of a Hypercube Supercomputer", *Proc. 1986 International Conference on Parallel Processing*, 1986, pp. 653-660.

[10] J. Rattner, "Concurrent Processing: A New Direction in Scientific Computing", *Proc. 1985 National Computer Conference*, AFIPS Press, vol. 54, 1985, pp. 764-771.

[11] J. O. Tuazon, J. C. Patterson et al.,"Caltech/JPL MARK II Hypercube Concurrent Processor", *Proc. 1985 International Conference on Parallel Processing*, Aug. 1985, pp. 666-673.

[12] J. C. Patterson, J. O. Tuazon, D. Leiberman et al.," The Mark III Hypercube Ensemble Concurrent computer", *Proc. 1985 International Conference on Parallel Processing*, Aug. 1985, pp. 71-73.

[13] R. D. Rasmussen, G. S. Bolotin, N. J. Dimopoulos, B. F. Lewis, and R. F. Manning, "Advanced General Purpose Multicomputer for Space Applications," *Proc. 1987 International Conference on Parallel Processing*, vol. 1, Aug. 1987, pp. 54-57.

[14] Thinking Machines Corporation, *The Connection Machine CM-5 Technical Summary*, MA, 1992.

[15] W. J. Dally, J. A. Stuart Fiske, J. S. Keen, R. A. Lethin, M. D. Noakes, P. R. Nuth, R. E. Davison, and G. Fyler, "The Message-Driven Processor: A Multicomputer Processing Node with Efficient Mechanisms", *IEEE Micro*, pp. 23-40, April 1992.

[16] T. Blank, "The MASPAR MP-1 Architecture", *Proc. COMPCON 90*, CA, 1990, pp. 20-24.

[17] Digital Equipment Corporation, "DECmpp 12000 Architecture Specification", Part No. AA-PMASA-TE, 1992.

[18] Intel Corp., *Paragon System - Technical Documentation Guide*, Order No 312820-002, June 1994.

[19] D. A. Reed and D. C. Grunwald, "The Performance of Multicomputer Interconnection Networks," *IEEE Comp.*, vol. 20, No. 6, pp. 63-73, June 1987.

[20] T. Feng, "A Survey of Interconnection Networks," *IEEE Computer,* pp. 12-27, Dec. 1981.

[21] A. Varma and C. S. Raghavendra, *Interconnection Networks for multiprocessors and multicomputers*, IEEE Computer Society Press, CA, 1994.

[22] T. Lang and H. S. Stone, "A Shuffle exchange network with simplified control", *IEEE Trans. Comput.*, vol. C-25, pp. 55-56, Jan. 1976.

[23] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing - Design and Analysis of Algorithms*, The Benjamin/Cummins Publication Co., CA, 1994.

[24] R. D. Rasmussen, N. J. Dimopoulos, G. S. Bolotin, R. F. Manning and B. F. Lewis, "MAX: Advanced General Purpose Real-Time Multicomputer for Space Applications", *Proc. IEEE Real Time Systems Applications*, Dec. 1987, pp. 70-78.

[25] L. N. Bhuyan and D. P. Agrawal, "Generalized Hypercube and Hyperbus Structures for a Computer Network", *IEEE Trans. Comput.*, vol. C-33, No. 4, pp. 323-333, April 1984.

[26] L. N. Bhuyan and D. P. Agrawal, "Design and Performance of Generalized Interconnection Networks", *IEEE Trans. Comput.*, vol. C-32, No. 12, pp. 1081-1090, Dec. 1983.

[27] S. B. Akers and B. Krishnamurthy, "A Group-Theoretic Model for Symmetric Interconnection Networks", *IEEE Trans. Comput.*, vol. 38, No. 4, pp. 555-566, April 1989.

[28] N. J. Dimopoulos, R. D. Rasmussen, G. S. Bolotin, B. F. Lewis, and R. F. Manning, "Hypercycles, Interconnection Networks with simple Routing Strategies", *Proc. Canadian Conference on Electrical & Computer Engineering*, Nov. 1988, pp. 577-580.

[29] W. J. Dally, "Performance Analysis of k-ary n-cube Interconnection networks", *IEEE Trans. Comput.*, vol. 39, No. 6, pp. 775-785, June 1990.

[30] O. M. Lubeck, "Supercomputer Performance: The Theory, Practice and Results" in Marshall C. Yovits (ed.), *Advances in COMPUTERS*, vol. 27, Academic Press, pp. 315, 1988.

[31] N. Biggs, *Algebraic Graph Theory*, Cambridge University Press, 1979.

[32] Don Radvan, "Performance Evaluation and Router Design for the Backtrack-to-the-origin-and-retry-routing-scheme of the Hypercycle based interconnection networks", M.A.Sc. Thesis, University of Victoria, July 1990.

[33] N. J. Dimopoulos, D. Radvan, K. F. Li, " Backtrack-to-the-Origin-and-Retry-routing for Hypercycle based interconnection networks", *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, June 1989, pp. 173-177.

[34] G. Sabidussi, "Graph Multiplication", *Math. Zeitschr.* vol. 72, pp. 446-457, 1960.

[35] N. J. Dimopoulos, D. Radvan and K. F. Li, "Performance Evaluation of the Backtrack to the Origin and Retry Routing for Hypercycle based Interconnection Networks," *Proc. Tenth International Conference on Distributed Systems*, Paris, France, June 1990, pp. 278-284.

[36] A. Berge, *Principles of Combinatorics*, Prentice Hall, New York, 1987.

[37] M. Imase, T. Soneoko, "Connectivity of Regular Directed Graphs with Small Diameters", *IEEE Trans. Comput.*, vol. C-34, No. 3, pp. 267-273, March 1985.

[38] K. B. Irani and K. Chen, "Minimization of Interprocessor Communication for Parallel computation," *IEEE Trans. Comput.*, vol. C-31, No. 11, pp. 1067-1075, Nov. 1982.

[39] A. Kershenbaum, P. Kermani, and G. A. Grover, "MENTOR: An Algorithm for mesh Network topological optimization and routing", *IEEE Trans. Commun.*, vol. 39, No. 4, pp. 503-513, April 1991.

[40] V. Chaudhary and J. K. Aggarwal, "A Generalized scheme for mapping parallel algorithms," *IEEE Trans. Par. and Dist. Sys*, vol 4. No. 3, pp. 328-346, March 1993.

[41] A. Sengupta, P. D. Joshi, and S. Bandhyopadhyay, "A Synthesis Approach to design optimally Fault-tolerant network Architecture", *IEEE Trans. Comput.*, vol 40. No. 1, pp. 94-99, Jan. 1991.

[42] D. Sarkar, "Cost and Time-cost effectiveness of Multiprocessing," *IEEE Trans. Par. and Dist. Sys.*, vol. 4, No. 6, pp. 704-712, June 1993.

[43] C. I. Chen, T. H. Lai, Y-N. Lien, "A note on "Generalized hypercube and hyperbus structures for a computer network", *IEEE Trans. Comput.*, vol. 40, pp. 667-668, May 1991.

[44] A. L. Rosenberg, "Issues in the study of graph embedding", *Graph Theoretic concepts in Computer Science, Lecture Notes in Computer Science*, Springer-Verlag, vol. 100, New York, pp. 150-176, 1981.

[45] J. W. Hong, K. Mehlhorn and A. L. Rosenberg, "Cost Trade-offs in graph embeddings with applications", *Journal of ACM*, vol. 30, pp. 709-728, 1983.

[46] E. L. Leiss, "On Efficient Embeddings of Communication Structures", *Congress Numerantium*, vol. 59, pp. 185-194, 1987.

[47] B. Monien and H. Sudborough, "Comparing Interconnection Networks", Technical Report No. 55, Fachbereich Mathematik, Universitat Paderborn, Germany, Oct 1988.

[48] T. F. Chan and Y. Saad, "Multigrid algorithms on the Hypercubes multiprocessors", *IEEE Trans. Comput.*, vol. C-35, pp. 969-977, Nov. 1986.

[49] M. Y. Chan and F. Y. L. Chin, "On embedding of rectangular grids in Hypercubes", *IEEE Trans. Comput.*, vol. 37, pp. 1285-1288, Oct. 1988.

[50] C. T. Ho and S.L. Johnsson, "On embedding of arbitrary Meshes in Boolean cubes with expansion two dilation two", *Proc. 1987 International conference on Parallel Processing*, 1987, pp. 188-191.

[51] E. Horowitz and A. Zorat, "The binary tree as an interconnection network: Applications to multiprocessor systems and VLSI," *IEEE Trans. Comput.*, vol. C-30, No. 2, pp. 247-253, Feb. 1981.

[52] J. L. Bentley and H. T. Kung, " A Tree machine for searching problems", *Proc. 1979 International Conf. Parallel Proc.*, Detroit, Aug. 1979, pp. 257-266.

[53] J. D. Ullman, *Computational Aspects of VLSI*, Rockville, MD: Computer Science Press, 1984.

[54] S. N. Bhatt and I. F. Ipsen, "How to embed trees in Hypercubes", *Technical Report*, YALEU/DCS/RR-443, Yale University, Dec. 1985.

[55] H. Y. Youn and A. D. Singh, "Near optimal embedding of binary architectures in VLSI", *Int. Conf. Comp. Design*, 1988, pp. 86-93.

[56] F. Dehne and N. Santoro, "Optimal VLSI dictionary machines on meshes", *Proc. 1987 Int. Conf. Par. Processing*, 1987, pp. 86-93.

[57] A. Wagner, "Embedding trees in a hypercube", Ph.D. Thesis, Department of Computer Science, University of Toronto, 1987.

[58] I. Havel and P. Leibl, "Embedding the polytomic tree into the n-cube", *Casopis Pest. Mat.*, vol. 98, pp. 307-314, 1973.

[59] E. L. Leiss and H. N. Reddy, "Embedding complete binary trees into hypercubes", *Info. Proc. Lett.*, vol. 38, pp. 197-199, 1991.

[60] A. Y. Wu, "Embedding of Tree Networks into Hypercubes", *J. Par. and Dist. Comp.*, vol. 2, pp. 238-249, 1985.

[61] L. Nebesky, "On cubes and dichotomic trees", *Casopis Pest. Mat.*, vol. 9, pp. 164-167, 1974.

[62] A. Wagner, "Embedding arbitrary binary trees in a hypercube", *J. Par. and Dist. Comp.*, vol. 7, pp. 503-520, 1989.

[63] Garey, Graham, Johnson, Knuth, "Complexity results for bandwidth minimization", *SIAM J. Appl. Math.* vol. 34, pp. 477-495, 1978.

[64] P. J. Chin, J. Chvátalová, A. K. Dewdney and N. E. Gibbs, "The Bandwidth Problem for Graphs: A Collection of Recent Results", *Journal of Graph Theory*, No. 6, pp. 223-254, 1982.

[65] C. H. Papadimitriou, "The NP-completeness of the Bandwidth minimization problem", *Computing*, vol. 16, pp. 263-270, 1976,

[66] J. Chvátalová, "On the Bandwidth Problem for Graphs", Ph.D. Dissertation, University of Waterloo, 1980.

[67] M-J. Chung, F. Makedon, I. H. Sudborough, and J. Turner, "Polynomial time algorithms for the Min-cut problem on degree restricted trees", *SIAM J. Comp.* vol. 14, pp. 158-177, Feb. 1985.

[68] F. R. K. Chung, "Some Problems and results in Labellings of graphs", *Proc. Fourth Int. Graph Theory Conference*, 1980, pp. 255-263.

[69] D. Dolev, and H. Trickey, "Embedding a tree on a line", *IBM Technical Report*, RJ3368, 1982.

[70] M. S. Patterson, W. L. Ruzzo and L. Snyder, "Bounds on Minimax Edge length for Complete Binary Tree", *Proc. ACM Sympos. on Theory of Computing*, 1981, pp. 293-299.

[71] D. Gordon, "Efficient embeddings of Binary trees in VLSI Arrays", *IEEE Trans. Comput.*, vol. C-36, No. 9, pp. 1009-1018, Sept. 1987.

[72] L. G. Valiant, "Universality considerations in VLSI Circuits", *IEEE Trans. Comput.*, vol. C-30, No. 2, pp. 135-140, Feb. 1981.

[73] J. B. Saxe, "Dynamic Programming Algorithms for recognizing small bandwidth graphs in polynomial time", *SIAM. J. Alg. Disc. Methods I*, pp. 363-369, 1980.

[74] J. Chvátalová, A. K. Dewdney, P. J. Chin, and N. E. Gibbs, "The Bandwidth Problem for Graphs: A Collection of Recent Results", Technical Report No. 23, Dept. of Comp. Sc., University of Western Ontario, No. 6, 1975.

[75] R. Heckmann, R. Klasing, B. Monien and W. Unger, "Optimal embedding of complete binary trees into lines and grids", *Proc.* 17^{th} *Graph-Theoretic Concepts in Computer Science, Lecture Notes in Comp. Sc.*, vol. 570, 1991, pp. 25-35.

[76] J. Z. Cha and R. W. Mayne, "Optimization with Discrete Variables via Recursive Quadratic Programming: Part 2 - Algorithm and Results," *Journal of Mechanisms, Transmissions, and Automation in Design*, ASME, vol. 111, pp. 130-136, March 1989.

[77] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures - Arrays, Trees and Hypercubes*, M. Kaufmann Publishers, 1992.

[78] J. S. Hwo, S. Lakshmivarahan, S. K. Dhall, "Embedding of cycles and grids in Star graphs", *Journal of Circuits, Systems, and Computers*, vol. 1, pp. 43-74, 1991.

[79] H.J. Broersma, "A note on Hamiltonian graphs", *Ann. Discrete Math.*, vol. 41, pp. 71-74, 1989.

[80] R. J. Gould, "Updating the Hamiltonian Problem - A Survey", *Journal of Graph Theory*, vol. 15, pp. 121-158, June 1991.

[81] M.R. Garey and D. S. Johnson, *Computer and Interactability - A Guide to the theory of NP-completeness*, Freeman, 1979.

[82] E. Dixon and S. Goodman, "On the number of Hamiltonian circuits in the n-cube", *Proc. Am. Math. Soc.*, vol. 50, pp. 500-504, 1975.

[83] R. J. Douglas, "Bounds on the number of Hamiltonian circuits in the n-cube", *Disc. Math*, vol. 17, pp. 143-146, 1977.

[84] E. N. Gilbert, "Gray codes and Paths on the n-cube", *Bell Systems Tech. Journal*, vol. 37, pp. 815-826, May 1989.

[85] D. H. Smith, "Hamiltonian circuits on the n-cube", *Canadian Math. Bulletin*, vol. 17, pp. 759-761, 1974.

[86] B. Alspach, "Hamiltonian Cycles in Vertex-transitive graphs of order $2p$", *Proc.* 10^{th} *Southeastern Conf. on Combinatorics, Graph Theory and Computing*, Congressus Numerantium, vol. 1, April 1979, pp. 131-139.

[87] B. Alspach, "Hamiltonian cycles in metacirculant graphs with prime power cardinal blocks", *Ann. Discrete Math.*, vol. 41, pp. 7-16, 1989.

[88] B. Alspach, S. C. Locke and D. Witte, "The Hamiltonian Spaces of Cayley Graphs on Abelian Groups", *Disc. Math*, vol. 82, pp. 113-126, 1990.

[89] I. A. Peckham, "The Hamiltonian product of Graphs", *Lecture Notes in Mathematics*, Springer-Verlag, pp. 89-92, 1974.

[90] B. Alspach and T. D. Parsons, "On Hamiltonian cycles in metacirculant graphs", *Ann. Discrete Math.*, vol. 15, pp. 1-7, 1982.

[91] D. Witte and Gallian, "A survey - Hamiltonian cycles in Cayley graphs", *Disc. Math.*, vol. 51, pp. 293-304, 1984.

[92] H. P. Yap, "Symmetries in Graphs" in *Some Topics in Graph Theory*, London Mathematical Society, Lecture Notes Series 108, Camb. Press, London, 1986.

[93] D. Marušič, "Hamiltonian circuits in Cayley graphs", *Disc. Math.*, vol. 46, pp. 49-54, 1983.

[94] M. C. Er, "On Generating the N-ary Reflected Gray code", *IEEE Trans. Comput.*, vol. C-33, No. 8, pp. 739-741, Aug. 1984.

[95] B. Arazi, "An Approach for generating different types of Gray codes", *Informations and Control*, vol. 63, pp. 1-10, 1984.

[96] B. D. Sharma and R. K. Khanna, " On m-ary Gray codes", *Informations Sciences*, vol. 15, pp. 31-43, 1978.

[97] D. Stanton and D. White, *Constructive Combinatorics*, Springer-Verlag, New York, 1986.

[98] B. Alspach, J. C. Bermond and D. Sotteau, "Decomposition into Cycles I: Hamilton Decompositions" in *Cycles and Rays*, G. Hahn (editor), Kluwer Academic, pp. 9-18, 1990.

[99] J. C. Bermond, O. Favaron and M. Maheo, "Hamiltonian Decomposition of Cayley graphs of degree 4", *Journal of Combinatorial Theory*, Series B 46, pp. 142-153, 1989.

[100] P. T. Gaughan and S. Yalamanchilli, "Adaptive Routing Protocols for Hypercube Interconnection Networks", *IEEE Comput.*, vol. 26, No. 5, pp. 12-22, May 1993.

[101] S. A. Felperin, L. Gravano, G. D. Pifarré, and J.L. C. Sanz, "Routing techniques for Massively parallel Communication", *Proc. IEEE*, vol. 79, No. 4, pp. 488-503, April 1991.

[102] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication and switching technique", *Computer Networks*, vol. 3, pp. 267-286, 1979.

[103] Z. Zhang, "System Effects of Interprocessor Communication Latency in Multicomputers, *IEEE Micro*, pp. 12-15; 52-55, 1991.

[104] M. S. Chen and K. G. Shin, "Adaptive Fault-tolerant Routing in Hypercube Multicomputers", *IEEE Trans. Comput.*, vol. 39, No. 12, pp. 1406-1416, Dec. 1990.

[105] N. J. Dimopoulos, M. Chowdhury, V. Dimakopoulos, and R. Sivakumar, "Routing and Broadcasting in Hypercycles, Deadlock Free and Backtracking Strategies", *Proc. Conference on Parallel Architectures and Languages Europe (PARLE 92)*, Paris, July 1992.

[106] W. J. Dally, "Virtual Channel Flow Control", *IEEE Trans. Par. and Dist. Sys.*, vol. 3, No. 2, pp. 194-205, March 1992.

[107] J. Duato, "A New Theory of deadlock-free Routing in Wormhole networks", *IEEE Trans. Par. and Dist. Sys.*, vol. 4, No. 12, pp. 1320-1331, Dec. 1993.

[108] L. G. Valiant and G. J. Brebner, "Universal schemes for parallel communication", *Proc. 13th ACM Symp. on Theory of Computing*, 1981, pp. 263-277.

[109] C. Germain. J Bĕchennec, D. Etiemble, and J-P. Sansonnet, "A Communication Architecture for a Massively Parallel Message-Passing Multicomputer", *Journal of Parallel and Distributed Computing*, vol. 19, pp. 1386-1397, 1993.

[110] S. Konstandinidou and L. Snyder, "The Chaos Router," *IEEE Trans. Comput.*, vol. 43, No. 12, pp. 1386-1397, Dec. 1994.

[111] W. J. Dally and Charles L. Seitz, "Deadlock-Free Message Routing in multiprocessor Interconnection networks", *IEEE Trans. Comput.*, vol. C-36, No. 5, pp. 547-553, May 1987.

[112] N. J. Dimopoulos and R. Sivakumar, "Deadlock-preventing routing in Hypercycles", *Canadian Journal of Electrical and Computer Engineering*, No. 4, vol. 19, pp. 193-199, Oct. 1994.

[113] G. D. Pifarré, L. Gravano, G. Denicolay, and J. L. C. Sanz, "Adaptive Deadlock- and Livelock-free routing in the Hypercube Network", *IEEE Trans. Comput.*, vol. 5, No. 11, pp. 1121-1139, Nov. 1994.

[114] S. Radhakrishnan, "VLSI implementation of the Backtrack-to-the-origin-and-retry-routing in Hypercycle based Interconnection Networks", M.A.Sc. Thesis, University of Victoria, June 1991.

[115] M. Abramovici, M. A. Breuer, A. D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.

[116] *CATPAG User's Manual*, version 2.0.1, Report ICI-025R0, Canadian Microelectronics Corporation, Queen's University, 1990.

[117] *The CMOS4S Standard Cell library*, VLSI Implementation Centre report, CMC, March 1990.

[118] D. D. Gajski, *Silicon Compilation*, Addison-Wesley, Reading-Mass., 1988.

[119] S. G. Shiva,"Automatic Hardware Synthesis", *Proc. IEEE*, vol. 71, No. 1, Jan. 1983.

[120] R. Sivakumar, S. Mayil Vahanan, P. Murugavel, and P. R. Venkatasubramaniam,"A special purpose silicon compiler for Synthesis and analysis of Supercomputing systems on PACUBE VLSI Arrays", *Project Report*, Sri Venkateswara College of Eng., Madras, India, April 1989.

[121] A. Martin, "Translating Concurrent Programs into VLSI chips" in *Lecture Notes in Computer Science*, D. Etiemble J.C. Syre (Eds), No. 605, Springer Verlag, Berlin Heidelberg, 1992.

[122] *IEEE Standard VHDL Language Reference Manual*, IEEE Std. 1076-1987, The Institute of Electrical and Electronics Engineers, New York, NY 1988.

[123] E. Sternheim, R. Singh, and Y. Trivedi, *Digital Design with Verilog HDL*, Automata Publishing Company, CA, 1990.

[124] R. Sivakumar, V. V. Dimakopoulos, N. J. Dimopoulos, "Design of a programmable Controller for Hypercycle based Interconnection Networks", *The 3rd Canadian Workshop on Field programmable Devices, Technology, Tools and Applications*, Montreal, May 1995, pp. 212-217.

[125] P. B. Hansen, *On PASCAL compilers*, Englewood Cliffs, N.J.: Prentice-Hall, 1985.

[126] *Autologic VHDL Synthesis Guide*, v8.2, Mentor Graphics Corporation, CA, 1993.

[127] D. L. Perry, *VHDL*, Second Edition, McGraw-Hill, New York, 1989

[128] *Mentor Graphics VHDL Reference Manual*, v8.2, Mentor Graphics Corporation, Feb. 1993.

[129] *The Programmable Data Book*, Xilinx Corporation, CA, 1994.

[130] S. L. Johnson and C. T. Ho, "Optimum Broadcasting and Personalized Communication in Hypercubes", *IEEE Trans. Comput.*, vol. 38, No. 9, pp. 1249-1268, Sept. 1989.

[131] R. Sivakumar and N. J. Dimopoulos, "All-to-All Broadcasting in 1-D Hypercycles", *Technical Report*, ECE-94-2, University of Victoria, April 1994.

[132] V. V. Dimakopoulos and N. J. Dimopoulos, "Optimal Total Exchange in Linear Arrays and Rings", *Technical Report*, ECE-94-5, University of Victoria, April 1994.

Appendix A

A.1 Proof of Lemma 2.1

An upper bound of δ can be expressed as

$$\hat{\delta} = \frac{\sum_{j=1}^{r} \hat{S}_j}{M-1}$$

$$\hat{S}_j = \prod_{k=1,k\neq j}^{r} m_k \sum_{\beta_j=0}^{m_j-1} \hat{\Psi}(\beta_j, 0)$$

$$\hat{\Psi}(\beta_j, 0) = \begin{cases} \frac{\beta_j}{\rho_j} + 1 & \text{if } \beta_j \leq (m_j - \beta_j) \\ \\ \frac{m_j - \beta_j}{\rho_j} + 1 & \text{otherwise} \end{cases}$$

We partition $\hat{\delta}$ as the sum of the distances in dimension 1 and the remaining $r-1$ dimensions of $\mathcal{G}_{\vec{m}}^{\vec{\rho}}$ such that

$$\begin{aligned}
(M-1)\hat{\delta} &= \prod_{h=2}^{r} m_h \left(\sum_{i=\lceil \frac{m_1}{2} \rceil}^{m_1-1} \left(\frac{m_1 - i}{\rho_1} + 1 \right) + \sum_{i=0}^{\lceil \frac{m_1}{2} \rceil - 1} \left(\frac{i}{\rho_1} + 1 \right) \right) \\
&\quad + \sum_{j=2}^{r} \prod_{k=1,k\neq j}^{r} m_k \left(\sum_{\beta_j=\lceil \frac{m_j}{2} \rceil}^{m_j-1} \left(\frac{m_j - \beta_j}{\rho_j} + 1 \right) + \sum_{\beta_j=0}^{\lceil \frac{m_j}{2} \rceil - 1} \left(\frac{\beta_j}{\rho_j} + 1 \right) \right) \\
&= \prod_{h=2}^{r} m_h \left(m_1 + \frac{1}{\rho_1} \left(\sum_{i=\lceil \frac{m_1}{2} \rceil}^{m_1-1} m_1 - i + \sum_{i=0}^{\lceil \frac{m_1}{2} \rceil - 1} i \right) \right) \\
&\quad + \sum_{j=2}^{r} \prod_{k=1,k\neq j}^{r} m_k \left(m_j + \frac{1}{\rho_j} \left(\sum_{\beta_j=\lceil \frac{m_j}{2} \rceil}^{m_j-1} m_j - \beta_j + \sum_{\beta_j=0}^{\lceil \frac{m_j}{2} \rceil - 1} \beta_j \right) \right) \\
&= \prod_{h=2}^{r} m_h \left(m_1 + \frac{1}{\rho_1} \left\lfloor \frac{m_1^2}{4} \right\rfloor \right) + \sum_{j=2}^{r} \prod_{k=1,k\neq j}^{r} m_k \left(m_j + \frac{1}{\rho_j} \left\lfloor \frac{m_j^2}{4} \right\rfloor \right)
\end{aligned}$$

$$\begin{aligned}
&\leq \prod_{h=2}^{r} m_h \left(m_1 + \frac{m_1^2}{4\rho_1}\right) + \sum_{j=2}^{r} \prod_{k=1,k\neq j}^{r} m_k \left(m_j + \frac{m_j^2}{4\rho_j}\right) \\
&= \prod_{h=1}^{r} m_h \left(1 + \frac{m_1}{4\rho_1}\right) + \sum_{j=2}^{r} \prod_{k=1,}^{r} m_k \left(1 + \frac{m_j}{4\rho_j}\right) \\
&= M\left(\frac{m_1}{4\rho_1} + \sum_{j=2}^{r} \frac{m_j}{4\rho_j} + r\right)
\end{aligned}$$

Similarly the lower bound of δ is given by $\check{\delta} = \frac{\sum_{j=1}^{r} \check{S}_j}{M-1}$ where

$$\begin{aligned}
\check{S}_j &= \prod_{k=1,k\neq j}^{r} m_k \sum_{\beta_j=0}^{m_j-1} \check{\Psi}(\beta_j, 0) \\
\check{\Psi}(\beta_j, 0) &= \begin{cases} \frac{\beta_j}{\rho_j} & \text{if } \beta_j \leq (m_j - \beta_j) \\ \\ \frac{m_j - \beta_j}{\rho_j} & \text{otherwise} \end{cases}
\end{aligned}$$

and we obtain

$$\begin{aligned}
(M-1)\check{\delta} &= \prod_{h=2}^{r} m_h \left(\sum_{i=\left\lceil \frac{m_1}{2} \right\rceil}^{m_1-1} \left(\frac{m_1 - i}{\rho_1}\right) + \sum_{i=0}^{\left\lceil \frac{m_1}{2} \right\rceil - 1} \left(\frac{i}{\rho_1}\right) \right) \\
&\quad + \sum_{j=2}^{r} \prod_{k=1,k\neq j}^{r} m_k \left(\sum_{\beta_j=\left\lceil \frac{m_j}{2} \right\rceil}^{m_j-1} \left(\frac{m_j - \beta_j}{\rho_j}\right) + \sum_{\beta_j=0}^{\left\lceil \frac{m_j}{2} \right\rceil - 1} \left(\frac{\beta_j}{\rho_j}\right) \right) \\
&= \prod_{h=2}^{r} m_h \left(\frac{1}{\rho_1} \left\lfloor \frac{m_1^2}{4} \right\rfloor\right) + \sum_{j=2}^{r} \prod_{k=1,k\neq j}^{r} m_k \left(\frac{1}{\rho_j} \left\lfloor \frac{m_j^2}{4} \right\rfloor\right)
\end{aligned}$$

Observe that when m_t is odd, $1 \leq t \leq r$, $\left\lfloor \frac{m_t^2}{4} \right\rfloor = \frac{m_t^2 - 1}{4}$ and when m_t is even, $\left\lfloor \frac{m_t^2}{4} \right\rfloor \geq \frac{m_t^2 - 1}{4}$. Hence

$$\begin{aligned}
(M-1)\check{\delta} &\geq \prod_{h=2}^{r} m_h \left(\frac{m_1^2 - 1}{4\rho_1}\right) + \sum_{j=2}^{r} \prod_{k=1,k\neq j}^{r} m_k \left(\frac{m_j^2 - 1}{4\rho_j}\right) \\
&= \prod_{h=1}^{r} m_h \left(\frac{m_1}{4\rho_1} - \frac{1}{4}\right) + \sum_{j=2}^{r} \prod_{k=1,}^{r} m_k \left(\frac{m_j}{4\rho_j} - \frac{r-1}{4}\right) \\
&= M\left(\frac{m_1}{4\rho_1} + \sum_{j=2}^{r} \frac{m_j}{4\rho_j} - \frac{r}{4}\right)
\end{aligned}$$

■

Appendix B

B.1 Proof of Lemma 2.6

Rewriting Eq. (2.3), we obtain

$$f(m,\rho) = \sum_{i=0}^{\lceil \frac{m}{2} \rceil - 1} \left\lceil \frac{i}{\rho} \right\rceil + \sum_{i=\lceil \frac{m}{2} \rceil}^{m-1} \left\lceil \frac{m-i}{\rho} \right\rceil$$

Consider the following cases:

1. m **- odd**

$$\begin{aligned} f(m,\rho) &= \sum_{i=1}^{\frac{m-1}{2}} \left\lceil \frac{i}{\rho} \right\rceil + \sum_{i=\frac{m+1}{2}}^{m-1} \left\lceil \frac{m-i}{\rho} \right\rceil \\ &= \sum_{i=1}^{\frac{m-1}{2}} \left\lceil \frac{i}{\rho} \right\rceil + \sum_{j=\frac{m-1}{2}}^{1} \left\lceil \frac{j}{\rho} \right\rceil \quad \text{(substituting } i = m - j \text{ in the II term of the above Equation)} \\ &= 2 \sum_{i=1}^{\frac{m-1}{2}} \left\lceil \frac{i}{\rho} \right\rceil \end{aligned}$$

2. m **even**

$$\begin{aligned} f(m,\rho) &= \sum_{i=0}^{\frac{m}{2}-1} \left\lceil \frac{i}{\rho} \right\rceil + \sum_{i=\frac{m}{2}}^{m-1} \left\lceil \frac{m-i}{\rho} \right\rceil \\ &= \sum_{i=1}^{\frac{m}{2}-1} \left\lceil \frac{i}{\rho} \right\rceil + \left\lceil \frac{m}{2\rho} \right\rceil + \sum_{i=\frac{m+2}{2}}^{m-1} \left\lceil \frac{m-i}{\rho} \right\rceil \\ &= \left(\sum_{i=1}^{\frac{m}{2}-1} \left\lceil \frac{i}{\rho} \right\rceil + \left\lceil \frac{m}{2\rho} \right\rceil \right) + \left(\sum_{i=\frac{m+2}{2}}^{m-1} \left\lceil \frac{m-i}{\rho} \right\rceil + \left\lceil \frac{m}{2\rho} \right\rceil \right) - \left\lceil \frac{m}{2\rho} \right\rceil \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{\frac{m}{2}} \left\lceil \frac{i}{\rho} \right\rceil + \sum_{i=\frac{m}{2}}^{m-1} \left\lceil \frac{m-i}{\rho} \right\rceil - \left\lceil \frac{m}{2\rho} \right\rceil \\
&= \sum_{i=1}^{\frac{m}{2}} \left\lceil \frac{i}{\rho} \right\rceil + \sum_{j=\frac{m}{2}}^{1} \left\lceil \frac{j}{\rho} \right\rceil - \left\lceil \frac{m}{2\rho} \right\rceil \qquad \text{(substituting } i = m - j \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{the II term of the above)} \\
&= 2\sum_{i=1}^{\frac{m}{2}} \left\lceil \frac{i}{\rho} \right\rceil - \left\lceil \frac{m}{2\rho} \right\rceil
\end{aligned}
$$

■

B.2 Proof of Lemma 2.7

In this section, we first derive a closed formula for computing $\sum_{i=1}^{n} \left\lceil \frac{i}{\rho} \right\rceil$. For given integers ρ and $i \geq 1$, we write

$$i = \rho x_i + r_i \tag{B.1}$$

where x_i and r_i are integers such that $x_i \geq 0$ and $1 \leq r_i \leq \rho$. Note that such a representation of i is unique. It follows that

$$\left\lceil \frac{i}{\rho} \right\rceil = x_i + 1 \tag{B.2}$$

$$\left\lfloor \frac{i}{\rho} \right\rfloor = \begin{cases} x_i & \text{if } 1 \leq r_i \leq \rho - 1 \\ x_i + 1 & \text{if } r_i = \rho \end{cases} \tag{B.3}$$

Therefore,

$$
\begin{aligned}
\sum_{i=1}^{n} \left\lceil \frac{i}{\rho} \right\rceil &= \sum_{i=1}^{n} (x_i + 1) \\
&= n + \sum_{i=1}^{n} \frac{i - r_i}{\rho} \\
&= n + \frac{n(n+1)}{2\rho} - \frac{1}{\rho} \sum_{i=1}^{n} r_i
\end{aligned} \tag{B.4}
$$

In order to evaluate the term $\frac{1}{\rho} \sum_{i=1}^{n} r_i$, we consider the following cases:

Case 1: n/ρ is an integer

It can be observed that for $1 \leq i \leq \rho$, we have $x_i = 0$ and $r_i = i$. Hence $\sum_{i=1}^{\rho} r_i = \frac{\rho(\rho+1)}{2}$. Similarly, for $(\rho + 1) \leq i \leq 2\rho$, we have $x_i = 1$ and $r_i = i$. So $\sum_{i=\rho+1}^{2\rho} r_i =$

$\frac{\rho(\rho+1)}{2}$. In general, we obtain $\sum_{i=\rho(j-1)+1}^{\rho j} r_i = \frac{\rho(\rho+1)}{2}$ for $j \in \left\{1, 2, \ldots, \frac{n}{\rho}\right\}$. Therefore

$$\begin{aligned}
\frac{1}{\rho}\sum_{i=1}^{n} r_i &= \frac{1}{\rho}\underbrace{\left[\sum_{i=1}^{\rho} r_i + \sum_{i=\rho+1}^{2\rho} r_i + \ldots + \sum_{i=n-2\rho+1}^{n-\rho} r_i + \sum_{i=n-\rho+1}^{n} r_i\right]}_{n/\rho \quad \text{terms}} \\
&= \left(\frac{n}{\rho}\right)\frac{(\rho+1)}{2} \\
&= \left\lfloor\frac{n}{\rho}\right\rfloor\frac{(\rho+1)}{2} \qquad \text{(B.5)}
\end{aligned}$$

Case 2: n/ρ **is not an integer**

$$\begin{aligned}
\frac{1}{\rho}\sum_{i=1}^{n} r_i &= \frac{1}{\rho}\sum_{i=1}^{\left\lfloor\frac{n}{\rho}\right\rfloor\rho} r_i + \frac{1}{\rho}\sum_{i=\left\lfloor\frac{n}{\rho}\right\rfloor\rho+1}^{n} r_i \\
&= \left\lfloor\frac{n}{\rho}\right\rfloor\frac{(\rho+1)}{2} + \frac{1}{\rho}\sum_{i=\left\lfloor\frac{n}{\rho}\right\rfloor\rho+1}^{n} r_i \qquad \text{(B.6)}
\end{aligned}$$

Let $i = \rho\left\lfloor\frac{n}{\rho}\right\rfloor + j$ where $j \in \left\{1, 2, \ldots, n - \left\lfloor\frac{n}{\rho}\right\rfloor\rho\right\}$. Since $n - \left\lfloor\frac{n}{\rho}\right\rfloor\rho = (n \bmod \rho) < \rho$, we have $1 \le j < \rho$. Furthermore, since $i = \rho x_i + r_i = \rho\left\lfloor\frac{n}{\rho}\right\rfloor + j$, the uniqueness of the representation of i as given by (B.1) implies that

$$x_i = \left\lfloor\frac{n}{\rho}\right\rfloor \quad \text{and} \quad r_i = j$$

Therefore,

$$\begin{aligned}
\frac{1}{\rho}\sum_{i=\left\lfloor\frac{n}{\rho}\right\rfloor\rho+1}^{n} r_i &= \frac{1}{\rho}\sum_{j=1}^{n-\left\lfloor\frac{n}{\rho}\right\rfloor\rho} j \\
&= \frac{\left(n - \left\lfloor\frac{n}{\rho}\right\rfloor\rho\right)\left(n+1-\left\lfloor\frac{n}{\rho}\right\rfloor\rho\right)}{2\rho} \qquad \text{(B.7)}
\end{aligned}$$

Substituting (B.7) into (B.6), we obtain

$$\frac{1}{\rho}\sum_{i=1}^{n} r_i = \left\lfloor\frac{n}{\rho}\right\rfloor\frac{(\rho+1)}{2} + \frac{\left(n - \left\lfloor\frac{n}{\rho}\right\rfloor\rho\right)\left(n+1-\left\lfloor\frac{n}{\rho}\right\rfloor\rho\right)}{2\rho} \qquad \text{(B.8)}$$

Note that the second term on the right hand side of (B.8) vanishes when n/ρ is an integer and (B.8) in such a case becomes (B.5). Hence (B.8) can be used to evaluate $\frac{1}{\rho}\sum_{i=1}^{n} r_i$ for the two discussed cases. Substituting (B.8) into (B.4), we obtain

$$\begin{aligned}
\sum_{i=1}^{n}\left\lceil\frac{i}{\rho}\right\rceil &= n + \frac{n(n+1)}{2\rho} - \left\lfloor\frac{n}{\rho}\right\rfloor\left(\frac{\rho+1}{2}\right) - \frac{1}{2\rho}\left[\left(n - \left\lfloor\frac{n}{\rho}\right\rfloor\rho\right)\left(n+1-\left\lfloor\frac{n}{\rho}\right\rfloor\rho\right)\right] \\
&= n + \left(n - \frac{\rho}{2}\right)\left\lfloor\frac{n}{\rho}\right\rfloor - \frac{\rho}{2}\left\lfloor\frac{n}{\rho}\right\rfloor^2 \qquad \text{(B.9)}
\end{aligned}$$

If $X = n \bmod \rho$, then $\left\lfloor \frac{n}{\rho} \right\rfloor = \frac{n-X}{\rho}$. Simplifying Eq. (B.9), we obtain

$$\begin{aligned}
\sum_{i=1}^{n} \left\lceil \frac{i}{\rho} \right\rceil &= n + \frac{n-X}{\rho}\left(n - \frac{\rho}{2} - \frac{n-X}{2}\right) \\
&= n + \frac{n-X}{\rho}\left(\frac{n}{2} - \frac{\rho - X}{2}\right) \\
&= n + \frac{n^2}{2\rho} - \frac{nX}{2\rho} - \frac{n}{2} + \frac{nX}{2\rho} + \frac{X(\rho - X)}{2\rho} \\
&= \frac{n}{2} + \frac{n^2}{2\rho} + \frac{X(\rho - X)}{2\rho} \qquad \text{(B.10)}
\end{aligned}$$

■

B.3 Proof of Lemma 2.8

Consider the following cases of m:

1. m - odd

$$f(m,\rho) = 2\sum_{i=1}^{\frac{m-1}{2}} \left\lceil \frac{i}{\rho} \right\rceil$$

Substituting $n = \frac{m-1}{2}$ in Eq. (2.14), we get

$$\begin{aligned}
f(m,\rho) &= 2\left(\frac{m-1}{4} + \frac{(m-1)^2}{8\rho} + \frac{X(\rho - X)}{2\rho}\right) \quad \text{where } X = \tfrac{m-1}{2} \bmod \rho \\
&= \frac{m-1}{2} + \frac{(m-1)^2}{4\rho} + \frac{X(\rho - X)}{\rho} \qquad \text{(B.11)}
\end{aligned}$$

Since $X = 0$, we have

$$f(m,\rho) = \frac{m-1}{2} + \frac{(m-1)^2}{4\rho} \qquad \text{(B.12)}$$

2. m- even

$$f(m,\rho) = 2\sum_{i=1}^{\frac{m}{2}} \left\lceil \frac{i}{\rho} \right\rceil - \left\lceil \frac{m}{2\rho} \right\rceil$$

Substituting $n = \frac{m}{2}$ in Eq. (2.14)

$$\begin{aligned}
f(m,\rho) &= 2\left(\frac{m}{4} + \frac{m^2}{8\rho} + \frac{X(\rho - X)}{2\rho}\right) - \frac{m}{2\rho} \quad \text{where } X = \tfrac{m}{2} \bmod \rho \\
&= \frac{m}{2} + \frac{m^2}{4\rho} + \frac{X(\rho - X)}{\rho} - \left\lceil \frac{m}{2\rho} \right\rceil \qquad \text{(B.13)}
\end{aligned}$$

As $X = 0$, Eq. (B.13) reduces to

$$\begin{aligned}
f(m,\rho) &= \frac{m}{2} + \frac{m^2}{4\rho} - \frac{m}{2\rho} \\
&= \frac{m}{2} + \frac{m(m-2)}{4\rho} \qquad \text{(B.14)}
\end{aligned}$$

■

Appendix C

Proof of Lemma 2.9:

From Eq. (B.13), we know that $X = m/2 \bmod \rho$. Substituing the value of $m = \rho(\rho-1)$, we get

$X = (\rho(\rho-1)/2) \bmod \rho = ((\rho^2/2) \bmod \rho - (\rho/2) \bmod \rho) \mod \rho = (0 - (\frac{\rho}{2}) \bmod \rho) \bmod \rho = \rho/2$. Hence substituting X and m in Eq. (B.13), we have

$$\begin{aligned} f(m,\rho) &= \frac{\rho(\rho-1)}{2} + \frac{\rho^2(\rho-1)^2}{4\rho} + \frac{\frac{\rho}{2}(\rho-\frac{\rho}{2})}{\rho} - \left\lceil \frac{\rho(\rho-1)}{2\rho} \right\rceil \\ &= \frac{\rho(\rho-1)}{2} + \frac{\rho(\rho-1)^2}{4} + \frac{\rho}{4} - \frac{\rho}{2} \\ &= \frac{\rho(\rho-1)}{2}\left(\frac{\rho+1}{2}\right) - \frac{\rho}{4} \\ &= \frac{\rho}{4}\left(\rho^2-2\right) \end{aligned} \tag{C.1}$$

■

Proof of Lemma 2.10:

Note that $X = (m+\upsilon)/2 \bmod \rho = \rho^2/2 \bmod \rho = 0$ since q is even. Hence Eq. (B.13) reduces to the following

$$\begin{aligned} f(m,\rho) &= \frac{\rho^2}{2} + \frac{\rho^2(\rho^2-2)}{4\rho} \\ &= \frac{\rho}{4}\left(\rho^2+2\rho-2\right) \end{aligned} \tag{C.2}$$

■

Proof of Lemma 2.11:

Observe that $m - \lambda = (\rho-1)^2$ is odd and hence $X = \lfloor (m-\lambda)/2 \rfloor \bmod \rho$. This can be simplified as $X = (\rho-1)^2 - 1)/2 \bmod \rho = (\rho^2 - 2\rho)/2 \bmod \rho = 0$ since ρ is even. Hence Eq. (B.12) reduces to

$$f(m,\rho) = \frac{m-\lambda-1}{2} + \frac{(m-\lambda-1)^2}{4\rho}$$

$$\begin{aligned} &= \frac{(\rho-1)^2-1}{2} + \frac{((\rho-1)^2-1)^2}{4\rho} \\ &= \frac{\rho(\rho-2)}{2} + \frac{\rho(\rho-2)^2}{4} \\ &= \frac{\rho^2(\rho-2)}{4} \end{aligned} \tag{C.3}$$

■

Proof of Lemma 2.12:

Evaluating $X = m/2 \bmod \rho$, we obtain $X = ((\rho^2 - 3\rho + 2)/2) \bmod \rho = (\rho+2)/2$. Substituting the values of X and m in Eq. (B.13), we get

$$\begin{aligned} f(m,\rho) &= \frac{(\rho-1)(\rho-2)}{2} + \frac{(\rho-1)^2(\rho-2)^2}{4\rho} + \frac{\frac{(\rho+2)}{2}\left(\rho - \frac{(\rho+2)}{2}\right)}{\rho} - \left\lceil \frac{(\rho-1)(\rho-2)}{2\rho} \right\rceil \\ &= \frac{\rho^2-3\rho+2}{2} + \frac{(\rho^2-2\rho+1)(\rho^2-4\rho+1)}{4\rho} + \frac{\rho^2-4}{4\rho} - \frac{(\rho-2)}{2} \\ &= \frac{\rho^3-4\rho^2+6\rho-4}{4} \\ &= \frac{(\rho-2)\,(\rho^2-2\rho+2)}{4} \end{aligned} \tag{C.4}$$

■

Proof of Lemma 2.13:

Since $m - \lambda = (\rho-2)^2$ is even, we have $X = ((\rho^2 - 4\rho + 4)/2) \bmod \rho$. In other words, $X = (\rho^2/2 - 2\rho + 2) \bmod \rho = 2$. Substituting $X = 2$ and $m - \lambda = (\rho-2)^2$ in Eq. (B.13), we obtain

$$\begin{aligned} f(m-\lambda,\rho) &= \frac{(\rho-2)^2}{2} + \frac{(\rho-2)^4}{4\rho} + \frac{2(\rho-2)}{\rho} - \left\lceil \frac{(\rho-2)^2}{2\rho} \right\rceil \\ &= \frac{\rho^2-4\rho+4}{2} + \frac{(\rho^2-4\rho+4)^2}{4\rho} + \frac{2(\rho-2)}{\rho} - \frac{(\rho-2)}{2} \\ &= \frac{\rho^3-6\rho^2+14\rho-12}{4} \\ &= \frac{(\rho-2)(\rho^2-4\rho+6)}{4} \end{aligned} \tag{C.5}$$

■

Proof of Lemma 2.14:

Evaluating $X = m/2 \bmod \rho$, we obtain $X = ((\rho^2+\rho)/2) \bmod \rho = \rho/2$. Substituting X and m in Eq. (B.13), we get

$$\begin{aligned} f(m,\rho) &= \frac{\rho(\rho+1)}{2} + \frac{\rho^2(\rho+1)^2}{4\rho} + \frac{\frac{\rho}{2}\left(\rho - \frac{\rho}{2}\right)}{\rho} - \left\lceil \frac{\rho(\rho+1)}{2\rho} \right\rceil \\ &= \frac{\rho(\rho+1)}{2} + \frac{\rho(\rho^2+2\rho+1)}{4} + \frac{\rho}{4} - \frac{(\rho+2)}{2} \end{aligned}$$

$$\begin{aligned} &= \frac{\rho^3 + 4\rho^2 + 2\rho - 4}{4} \\ &= \frac{(\rho+2)\,(\rho^2 + 2\rho - 2)}{4} \end{aligned} \tag{C.6}$$

■

Proof of Lemma 2.15:

Here $X = \lfloor (\rho+1)^2/2 \rfloor \bmod \rho = (((\rho+1)^2 - 1)/2) \bmod \rho = (\rho^2 + 2\rho)/2 \bmod \rho = 0$ since q is even. Hence Eq. (B.12) reduces to

$$\begin{aligned} f(m,\rho) &= \frac{(\rho+1)^2 - 1}{2} + \frac{((\rho+1)^2 - 1)^2}{4\rho} \\ &= \frac{\rho^2 + 2\rho}{2} + \frac{(\rho^2 + 2\rho)^2}{4\rho} \\ &= \frac{\rho^3 + 6\rho^2 + 8\rho}{4} \\ &= \frac{\rho(\rho+2)(\rho+4)}{4} \end{aligned} \tag{C.7}$$

■

Appendix D

D.1 Proof of Lemma 4.1

By construction, let $(S_1, *)$ be an abelian group defined as $S_1 = \{0, 1, 2, \ldots, m_1 - 1\}$ and the binary operator '*' denote *addition modulo* m_1. Let $I = \{0\}$ be a set containing the identity element of $(S_1, *)$. Define

$$T_1 = \cup\{x, m_1 - x\} \quad \forall x \in \{1, \ldots, \rho_1\} \tag{D.1}$$

where x and $m_1 - x$ are inverses to each other with respect to the operator '*'. Then $C_1 = G(S_1, T_1)$ is a connected Cayley graph and T_1 constitutes a generator set for S_1. We shall now show that the Cayley graph $C_1 = G(S_1, T_1)$ is isomorphic to $\mathcal{G}_{m_1}^{\rho_1}$. Observe from the definition of S_1 that $|V(C_1)| = |V(\mathcal{G}_{m_1}^{\rho_1})|$. To establish the correspondence of edges between C_1 and $\mathcal{G}_{m_1}^{\rho_1}$, we need to prove the following

$\Rightarrow$ Let $(a, b) \in E(C_1)$, then $a^{-1} * b \in T_1$. From the definition of T_1 given by Eq. (D.1), the following cases arise.

Case 1: Let $a^{-1} * b = x$. Then $b = a * x = (a + x) \bmod m_1 \Rightarrow (a, b) \in E(\mathcal{G}_{m_1}^{\rho_1})$ $\forall x \in \{1, \ldots, \rho_1\}$ (from Eq. 1.3).

Case 2: Let $a^{-1} * b = m_1 - x$. Then $b = a * x = (a + m_1 - x) \bmod m_1 = (a - x) \bmod m_1 \Rightarrow (a, b) \in E(\mathcal{G}_{m_1}^{\rho_1})$.

$\Leftarrow$ Let $(a, b) \in E(\mathcal{G}_{m_1}^{\rho_1})$. Then, from the edge definition for $\mathcal{G}_{m_1}^{\rho_1}$ given by Eq. (1.3), we have the following cases.

Case 1: Let $b = (a + \xi_1) \bmod m_1$, for $1 \leq \xi_1 \leq \rho_1$. If $(a, b) \in E(C_1)$, then $a^{-1} * b = (a^{-1} + a + \xi_1) \bmod m_1 = \xi_1 \in T_1$.

Case 2: Let $b = (a - \xi_1) \bmod m_1$. Similarly if $(a, b) \in E(C_1)$ then $a^{-1} * b = (a^{-1} + a - \xi_1) \bmod m_1 = -\xi_1 \bmod m_1 = m_1 - \xi_1 \in T_1$.

■

Appendix E

CoDeL program listing of e-cube controller

```
#######################################################################
# This program describes the action of a controller for the e-cube    #
# routing policy for 2-D hypercycle interconnection networks          #
#######################################################################

bitstruct mixed_radix_2 # Mixed radix definition
{
  (bits) field1[2];  # address in dimension 1
  (bits) field2[2];  # address in dimension 2
}

bitstruct data_frame      # A 5-bit frame (header, destn address)
{
  (mixed_radix_2) destn_address; # destination address (4-bits)
  (bits)          header;        # message header (1-bit)
}

#######################################################################
# We assume the cross bar to be 6 x 10                                #
# 6 inputs consist of (4 ports + 1 host + 1 for inject module)        #
# 10 outputs consist of 5 outports (4 + 1 for host)                   #
# while the rest 5 outports are connected to controller               #
#######################################################################
```

```
bitstruct cross_bar_frame
{
  (bits) m_loc[6];     # 6 bit address for 60 locations
  (bits) m_data;       # matrix switch ( 0 - open; 1 - connect)
}

bitstruct npg_port_frame
{
  (bits) port_address[5];   # Port address from e-cube router
                            #  (Max. 5 physical ports)
         dtr;               # destination has been reached
         wait_at_node;      # indicates a block has occurred
         is_ecube;          # in routing asserts if network
                            # parameters conform to e-cube routing
}

#######################################################################
# Main module of ecube controller                                     #
# Protocols used for various ports                                    #
#   p1 - full handshake (input) for importing frame                   #
#   p2 - ecube protocol (input) to get computed port from router      #
#   p4 - ecube protocol (output) to send control and network          #
#          data (eg. destination address) to e-cube router            #
#   p3 - full handshake (output) to communicate with cross bar and    #
#          injection module                                           #
#                                                                     #
#######################################################################

module ecube_controller_new (
  out bus_enable[6],
  in  (data_frame)      p1 with inp_hsk,
  in  (npg_port_frame)  p2 with inp_ecube,
  out (data_frame)      p3 with out_hsk,
```

```
  out                    p4[11] with out_ecube,
  out (cross_bar_frame) cf with out_hsk,
  out cb_rst,
  in  config,
  in  data_sent,
  in  collision,
  in  available_ports[5]    # available_ports[0] signifies line
                            # connected to host
)
{
  register j[3], shift_value[6];
  register i[3];

  if (config)      # At power up, "config" input is held high to
  {                # program the  router with necessary data
    p4 = (0,0,0,0X11);      # connectivity vector
    output(p4);
    p4 = (0,0,1,0X22);      # current address of node
    output(p4);
    p4 = (0,3,0X33);        # Population of the network
    output(p4);
    p4 = (4,0X02);          # Port offset
    output(p4);
    cb_rst = 0;             # clear cross_bar memory

    j = 0;
    while (j <= 4)
    {
      cf.m_loc = 11*j + 5;
      cf.m_data = 1;        # configure cross bar switch
      output(cf);
      j = j+1;
    }
  }
```

```
else
{
  cb_rst = 1;                    # Enable cross bar
  i = 0;
  shift_value = 1;               # Initialize shift_value to 1
  while (i <= 4)
  {
    shift_value = shift_value << 1;   # Left Shift bit
    bus_enable  = shift_value;   # Enable one of S/P devices (1 to 5)
    if (isready(p1))
    {                            # test each port for availability of
      input(p1);                 # obtain data if available
      p4 = (5,available_ports[4:1],0X0);  # concatenate address 5 and
      output(p4);                # send avail. ports to router
      p4 = (0,2,0,0,p1.destn_address.field2,
                0,0,p1.destn_address.field1);
                          # concatenate 2 along with the extracted
                          # destination address and send the result
                          # to router

      output(p4);         # send avail. ports to router
      input(p2);          # Receive computed port and
                          # control signals (from NPG) in port p2
      if (p2.dtr == 0)
      {
        if (p2.wait_at_node == 0)
        {
          cf.m_loc = 49 + p2.port_address;
          cf.m_data = 1;  # set cross_bar switch on
          output(cf);     # the computed port
          p3 = p1;        # send data_frame to inject module
          output(p3);
          wait(data_sent || collision);
          # wait for either acknowledgement or collision to be
```

```
            # detected by injector  module
            # clear cross bar location
            cf.m_loc = 49 + p2.port_address;
            cf.m_data = 0;  # Clear location after data is sent or
            output(cf);     # a collision occurs

            if (!collision)    # no collision then
            {
              cf.m_loc = i*10 + (p2.port_address - 1)(3:0);
              cf.m_data = 1;   # configure
              output(cf);      # cross bar for connecting to source
            }
          }   # if there is a collision, ignore the current port
        }     # and service the next available port
        else
        {
            if (available_ports[0])
            {                       # if host is free then
               cf.m_loc = 54;       # Destination has been reached
               cf.m_data = 1;       # set cross_bar switch on the
                                    # computed port
               output(cf);          # to send data to host (destn)
               p3 = p1;             # Connection set-up
               output(p3);
            }
        }
      }
      i = i+1;
    } ### end while
  } ### end else (of config)
}
#####################################################################
# End of e-cube controller program                                  #
#####################################################################
```

PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my dissertation to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this dissertation for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this dissertation for financial gain shall not be allowed without my written permission.

Title of Dissertation: TOPOLOGICAL OPTIMIZATION AND DEADLOCK-FREE ROUTING IN INTERCONNECTION NETWORKS BASED ON HYPERCYCLES.

Author: _____

SIVAKUMAR RADHAKRISHNAN

November 10^{th}, 1995