Organization of Systems with Bussed Interconnections

by

Shlomo Kipnis

B.Sc. Mathematics and Physics
The Hebrew University of Jerusalem
(1983)
M.Sc. Computer Science
The Hebrew University of Jerusalem
(1985)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 1990

Signature of Author_____
Department of Electrical Engineering and Computer Science
August 29, 1990

Certified by_____
Charles E. Leiserson
Associate Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by_____
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Organization of Systems with Bussed Interconnections

by

Shlomo Kipnis

Submitted to the Department of Electrical Engineering and Computer Science
on August 29, 1990, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis investigates several aspects of the organization of digital systems that employ *bussed interconnections*. The thesis focuses on two application domains for busses: communication architectures and control mechanisms, and explores the capabilities of busses as interconnection media, computation devices, and transmission channels.

Chapter 1 discusses the significance of bussed interconnect in digital systems, provides some background on busses, and describes the problems addressed in this thesis.

In Chapter 2 we investigate the organization of permutation architectures that employ bussed interconnections. We explore the problem of efficiently permuting data stored in VLSI chips in accordance with a predetermined set of permutations. By connecting chips with shared bus interconnections, as opposed to point-to-point interconnections, we show that the number of pins per chip can often be reduced. For example, we exhibit permutation architectures with $\lceil \sqrt{n} \rceil$ pins per chip that can realize any of the n cyclic shifts on n chips in one clock tick. When the set of permutations forms a group with p elements, any permutation in the group can be realized in one clock tick by an architecture with $O(\sqrt{p \lg p})$ pins per chip. When the permutation group is abelian, we show that $O(\sqrt{p})$ pins suffice. These results are all derived from a mathematical characterization of *uniform permutation architectures* based on the combinatorial notion of a *difference cover*. We also consider uniform permutation architectures that realize permutations in several clock ticks, instead of one, and show that further savings in the number of pins per chip can be obtained.

Chapter 3 explores efficient utilization of busses for implementing arbitration mechanisms. We investigate priority arbitration schemes that use busses to arbitrate among n modules in a digital system. We focus on distributed mechanisms that employ m busses, for $\lg n \leq m \leq n$, and use asynchronous combinational arbitration logic. A widely used distributed asynchronous mechanism is the *binary arbitration* scheme, which with $m = \lg n$ busses arbitrates in $t = \lg n$ units of bus-settling time. We present a new asynchronous scheme — *binomial arbitration* — that by using $m = \lg n + 1$ busses reduces the arbitration time to $t = \frac{1}{2} \lg n$. Extending this result, we present the *generalized binomial arbitration* scheme that achieves a bus-time tradeoff of the form $m = O(tn^{1/t})$ between the number of arbitration busses m, and the arbitration time t (in units of bus-settling time), for values of $1 \leq t \leq \lg n$ and $\lg n \leq m \leq n$. Our schemes are based on a novel analysis of *data-dependent delays*. Most importantly, our schemes can be adopted with no changes to existing hardware and protocols; they merely involve selecting a *good* set of priority arbitration codewords.

In Chapter 4, we examine the performance of priority arbitration schemes presented in Chapter 3 under the *digital transmission line* bus model. This bus model accounts for the propagation time of signals along bus lines and assumes that the propagating signals are always valid digital signals. A widely held misconception is that in the digital transmission line model the arbitration time of the binary arbitration scheme is at most 4 units of bus-propagation delay. We formally disprove this conjecture by demonstrating that the arbitration time of the binary arbitration scheme is heavily dependent on the arrangement of the arbitrating modules in the system. We provide a general scenario of module arrangement on m busses, for which binary arbitration takes at least $m/2$ units of bus-propagation delay to stabilize. We also prove that for general arrangements of modules on m busses, binary arbitration settles in at most $m/2+2$ units of bus-propagation delay, while binomial arbitration settles in at most $m/4+2$ units of bus-propagation delay, thereby demonstrating the superiority of binomial arbitration for general arrangements of modules under the digital transmission line model. For linear arrangements of modules in increasing order of priorities and equal spacings between modules, we show that 3 units of bus-propagation delay are necessary for binary arbitration to settle, and we sketch an argument that 3 units of bus-propagation delay are also asymptotically sufficient.

Finally, Chapter 5 provides some concluding remarks and identifies directions for further research on systems with bussed interconnections.

Keywords: arbitration, arbitration protocol, asynchronous arbitration, binary arbitration, binomial arbitration, bus-propagation time, bus-settling time, bus-time tradeoff, bussed interconnections, busses, cyclic shifter, data-dependent delays, difference cover, digital transmission line, generalized binomial arbitration, linear arbitration, permutation architecture, permutation set, priority arbitration, signal propagation, uniform architecture, VLSI.

Thesis Supervisor: Charles E. Leiserson
Title: Associate Professor of Computer Science and Engineering

Contents

Acknowledgments

This thesis is a result of an effort that lasted five years. During those five years, I have learned the meaning of scholarship from many people at MIT and outside it. I have engaged myself in a variety of activities, trying to grasp as much as possible of computer science, of MIT, and of Boston. I would like to take this opportunity and thank all those who have educated, taught, contributed, supported, encouraged, and entertained me along this path.

First and foremost, I would like to express my gratitude to my advisor Charles Leiserson. Without his guidance, I would probably not have been attracted to the exciting world of parallel and VLSI computation, and could not have achieved all that I have achieved. Charles was a constant source of ideas and motivation, had educated me about organized reasoning and presentation, and offered his advice in many occasions. It was not easy keeping up with his high standards, but looking back, it was well worth the effort. Much of this thesis was done in collaboration with him or under his direct supervision and guidance.

Besides Charles Leiserson, my other two committee members, Tom Leighton and Steve Ward, were also very involved in my research. Tom was a constant flow of information and references about theoretical aspects of parallel and VLSI computation. Steve introduced me to some of the problems related to busses and supplied me with much insight and valuable references. Their comments on my research and thesis were especially valuable.

There were many individuals who have helped me shape my interests and become a more knowledgeable researcher. I shall remain grateful to them and would like to mention those who were especially influential on my development, including Ron Rivest of the Theory of Computation group at MIT, Alan Oppenheim of the Research Laboratory of Electronics at MIT, Danny Dolev of the Hebrew University of Jerusalem, Marc Snir of IBM T. J. Watson Research Center, and Bob Thomas of BBN.

Many individuals contributed to and commented on my research and ideas. These include Hagit Attiya, Baruch Awerbuch, Benny Chor, Lance Fortnow, Shafi Goldwasser, Michelangelo Grigni, Joe Kilian, Phil Klein, Tom Knight, James Park, Hershel Safer, John Wolfe, and Su-Ming Wu of MIT, Noga Alon of Tel-Aviv University, Chuck Fiduccia of the Supercomputing

Research Center, Nicholas Pippenger of the University of British Columbia, Andrew Odlyzko of AT&T Bell Laboratories. Mark Manasse, James Saxe, and Chuck Thacker of DEC SRC, Bernard Chazelle of Princeton University, Alok Aggarwal and Marc Snir of IBM T. J. Watson Research Center, Bob Thomas of BBN, and Guy Steele of Thinking Machines.

My fellow graduate students deserve special thanks for helping me navigate through the Ph.D. process at MIT. Their social and morale support were invaluable, as well as their willingness to listen, participate, and share research ideas with me. Special thanks go to Tom Cormen, Paul Feldman, Andrew Goldberg, Sally Goldman, Ron Greenberg, Joe Kilian, Dina Kravets, Bruce Maggs, Yishay Mansour, James Park, Cindy Phillips, Serge Plotkin, Hershel Safer, Rob Schapire, Eric Schwabe, Jeff Siskind, Cliff Stein, and Joel Wein. My present and past office mates, Javed Aslam, Margrit Betke, Aditi Dhagat, Phil Klein, Satish Rao, Phillip Rogaway, and Laura Yedwab helped in creating a special and warm office atmosphere. I would like to thank them all.

My work could not have been the same in any other environment. The Laboratory for Computer Science at MIT is indeed a unique place to study and work. I feel fortunate for having spent those five years in this special environment, and would like to express my sincere thanks to the support staff, including William Ang, Arline Benford, Sally Bemus, Be Hubbard, Denise Sergent, and Barbara Tilson. In addition, I would like to acknowledge the financial support I received during my Ph.D. studies, which was provided by the Defense Advanced Research Projects Agency under Contract N00014–87–K–825.

Most important, I would like to thank my family who have all done so much to help me conquer the Ph.D. mountain. My parents, Bracha and Joshua Kipnis, and my brother, Aviad, encouraged me in many occasions along the way and offered much morale and financial support. My wife's family have also done much to ease those difficult years and traveled overseas many times to see us. My children, Avital, Elisha, and the newly born, Atarah, have given me much joy, helped me overcome the most difficult times, and kept me sane during those long years. Last, but mostly, I would like to thank (if this is the word) my wife, Fabienne, for taking so much on herself during those years, for providing the necessary financial and morale support, and for being there whenever I needed her. She is an important partner in this thesis. Affectionately, I would like to dedicate this thesis to her.

Chapter 1

Introduction

This thesis investigates several aspects of the organization of systems with bussed interconnections. Busses are used in many electronic and computer systems for a variety of applications, including broadcasting information, realizing communication patterns, implementing system primitives, and performing computations. Busses come in all shapes and sizes and connect modules at various system levels. Busses are the backbone of many digital systems and play a vital role in numerous architectures.

Busses are desirable in many systems due to their simplicity, modularity, reliability, and monitoring capabilities. Busses constitute shared media to which connected modules can listen and onto which they can broadcast. Busses offer scalable-cost interconnect, standard module interface, and configuration flexibility. Bussed organizations are easy to control and monitor, and provide a high level of reliability at moderate cost.

Busses have been extensively researched in the electrical engineering and computer science literature (see references). Various aspects of busses have been investigated, including the physical and electrical characteristics of the media, interconnection topologies, communication protocols, and algorithmic techniques, among others. Bussed interconnections are still not fully understood, however, and their capabilities are not fully exploited. Due to the widespread use of busses for applications in electronic and computer systems, it is important to develop a better understanding of the organization and capabilities of systems with bussed interconnections. In this thesis, we investigate several organizational aspects of digital systems that employ bussed interconnections and demonstrate how to use busses more efficiently for implementing several

system functions. Although the results of this thesis are presented with computer systems and computer busses in mind, they are not limited to these settings and are applicable to general systems that employ communication over shared media.

This thesis is organized as follows. In this chapter, we discuss several issues of bussed interconnections that are relevant to our work and describe the problems addressed in this thesis. The body of the thesis focuses on two application domains for shared interconnect: communication architectures and control mechanisms, and examines the capabilities of busses as interconnection media, computation devices, and transmission channels. In Chapter 2, we investigate the organization of permutation architectures that employ bussed interconnections. Chapter 3 explores how to implement priority arbitration mechanisms efficiently on busses that exhibit fixed settling delay. In Chapter 4, we examine the performance of some priority arbitration schemes under the digital transmission line model. Finally, Chapter 5 presents some concluding remarks and directions for further research concerning systems with bussed interconnections.

1.1 Bussed interconnections

Busses are shared communication media. Many digital systems employ one or more busses to communicate among system modules. Busses enable several devices sharing the same interconnection medium to communicate, in contrast with point-to-point wires that establish communication only between pairs of devices.

Several technologies of shared interconnect can be classified as busses, including broadcast radio channels, electrical wires, and optical fibers. The focus of this thesis is on electrical busses, which are used by most computer systems. Extensive surveys and tutorials on the characteristics of electrical busses appear in [16, 22, 40, 57, 82, 88]. Discussion of other shared communication media can be found, for example, in [12, 61, 78]. In this section, we briefly introduce and discuss several issues of electrical busses that are important for the development of this thesis and we comment on their relevance. We present these issues in a somewhat bottom-up manner.

Bus driving technologies. There are several standard technologies for driving digital signals onto an electrical bus. One common bus-driving technology is the tri-state driver, where

a device driver applies either a logic level of 0, a logic level of 1, or disables its output terminal and leaves it floating (see [22, 62, 88]). Tri-state drivers consume little power, but can only be used when it is guaranteed that at all times no more than one device drives the bus, while all other devices disable their drivers. This requirement must be met, since otherwise devices may fight each other, resulting in high-current spikes, intermediate voltage levels on the bus, and possible component failure. Another common bus-driving technology is the open-collector driver, where an external pullup drives the bus to a default logic level and device drivers can pull the bus down to express the nondefault logic value (see [22, 40, 88]). The open-collector technology allows the bus to implement a wired-OR logic function, since several devices can pull the bus down simultaneously, resulting in the OR of the logic values applied. (Another technology for implementing wired-OR is to charge and discharge a VLSI bus line that is treated as a large capacitor (see [62, 83]).) In this thesis, we explore both tri-state and open-collector drivers. The results of Chapter 2 can use either tri-state or open-collector busses, while Chapters 3 and 4 make use of open-collector busses.

Bus signal propagation. A bus, being a physical element, has several physical and electrical characteristics. The propagation of a signal on a bus takes time, which depends on the length, material, shape, temperature, and other physical properties of the bus and its environment. A high-speed bus is modeled as an analog transmission line with associated impedance that depends on the inductance, the capacity, and the length of the bus (see [5, 40]). Most computer systems, however, use the digital abstraction, which specifies certain discrete voltage levels for representing logic values. Digital signals driven onto a bus require time to propagate and to resolve various transient effects before the bus reaches a valid logic level. In designing digital bus primitives and protocols, careful attention must be given to modeling the bus appropriately and to allowing enough time for the bus to settle before the logic value that it carries can be reliably used. In this thesis we use the digital abstraction of busses. In Chapter 2, busses are used as interconnection media and we assume that sufficient time is allocated for signal propagation along a bus. In Chapters 3 and 4, busses may be driven by multiple modules and may carry transient signals. Chapter 3 assumes that the bus-settling time, denoted by T_{bus}, is accounted for, while Chapter 4 analyzes the effects of signal propagation along idealized digital transmission lines with bus-propagation time of T_{p}.

Number and functionality of bus lines. Bussed systems vary considerably in the number of bus lines they use and in their functionality. A single bus line can only implement one communication transaction at any given time and its performance, therefore, degrades when the number of modules connected to it increases; the latency of a bus with n modules is $\Theta(n)$ and its throughput is $\Theta(1/n)$. However, many bussed systems use a single bus line for serial communication when the cost associated with multiple lines is too high or when the functionality of the bus does not justify multiple lines (see [16, 22, 61, 88]). Most backplane bus systems, on the other hand, use a collection of bus lines to provide high bandwidth connections between system modules (see [16, 22, 40]). Such systems use parallel communication to transfer several bits concurrently, thereby reducing the time that the bus system is occupied by any given transaction. In addition, several multiplexing techniques enable multiple transactions over the same collection of bus lines by using time sharing or frequency sharing of the busses. Another common method for enhancing system connectivity and performance is the use of multiple busses to establish concurrent and independent communication channels among system modules or subsets of them (see [10, 13, 30, 54, 64, 69, 70, 73, 77]). In this thesis, we focus on multiple and parallel bus lines. Chapter 2 uses multiple busses to establish concurrent and independent communication channels among subsets of modules and Chapters 3 and 4 explore how to efficiently employ parallel bus lines that are shared among all system modules.

Bus timing disciplines. To control the behavior of a complex digital system, one of several timing disciplines is used (see [22, 62, 88]). There are two orthogonal dimensions to distinguish between timing disciplines: synchronous vs. asynchronous and global vs. local. In a synchronous system, there is a systemwide notion of time, generally established by using systemwide clock signals, that is used for timing and coordinating transactions. Bus transactions, in a synchronous system, start at some clock edge and finish at a subsequent clock edge, taking an integral multiple of clock cycles to complete. An asynchronous system, in contrast, does not time operations but rather coordinates them through the use of hand-shaking protocols. Bus transactions, in an asynchronous system, can start and finish at any time and their duration is self determined. In globally timed systems each operation takes a fixed and predetermined amount of time, while in locally timed systems modules can control the duration of different operations by using several control signals. These two orthogonal dimensions of classifying

timing disciplines give rise to four general classes of timing disciplines: Synchronous Globally Timed (SGT), Asynchronous Globally Timed (AGT), Synchronous Locally Timed (SLT), and Asynchronous Locally Timed (ALT). The choice between these timing disciplines depends on the purpose, performance, and cost of the designed system. In this thesis, we focus on the SGT, AGT, and ALT timing disciplines. The architectures of Chapter 2 use the synchronous globally timed discipline, while Chapters 3 and 4 explore asynchronous globally timed and asynchronous locally timed mechanisms.

Bus arbitration and mastership. Since a bus is shared among several system modules, situations may arise where the bus is simultaneously requested by more than one module. To allocate the bus to one module at a time, an arbitration/access mechanism is required that determines the mastership of the bus. Numerous arbitration/access mechanisms have been developed, including daisy chains, priority circuits, polling, token passing, and carrier sense multiple access protocols (see [12, 16, 22, 40, 57, 61, 78, 82, 88]). A distinction is often made between centralized arbitration/access mechanisms, where bus arbitration and access are determined by a central controller, and distributed arbitration/access mechanisms, where arbitration and access processes are carried out simultaneously by all system modules. Centralized controllers are generally simpler, operate fast, and are more flexible in their assignment procedures. Distributed controllers, on the other hand, are usually more reliable, require less dedicated wiring and communication, and are easier to monitor and expand. Many tightly coupled systems, such as SIMD parallel machines and high-performance architectures, use central control mechanisms, while more loosely coupled systems, such as multiprocessor systems and data communication networks, employ distributed arbitration/access mechanisms. In this thesis, both centralized and distributed control mechanisms are explored. The permutation architectures described in Chapter 2 use a centralized bus mastership procedure, while Chapters 3 and 4 investigate distributed arbitration mechanisms with busses.

Bus transactions. Busses can be used to implement several types of communication transactions that can be characterized by the sets of modules involved. The most common types of bus transactions are one-to-one, where a single module transmits data intended for a single receiver, and one-to-many (broadcast), where a single module sends information to multiple receivers. The receiver (receivers) of bus transactions are typically identified by their

address or through external control. Two other types of transactions, which are less frequently implemented on busses, are the many-to-one (converge) and many-to-many (multicast) communication patterns. In these transactions, several modules may try to transmit information concurrently over the same media, which requires some means of combining or selecting among the different requests. This thesis investigates some of these bus transactions. Chapter 2 deals with realizing permutations (one-to-one transactions) over bussed interconnections, while Chapters 3 and 4 use broadcast (one-to-many transactions) and multicast (many-to-many transactions) over wired-OR busses.

1.2 Focus and contribution of this thesis

Bussed interconnections are used for many applications in electronic and computer systems. This thesis focuses on two application domains for busses: communication architectures and control mechanisms, and examines the capabilities of busses as interconnection media, computation devices, and transmission channels. The following subsections describe the contribution of the thesis chapters and put the results of this thesis in perspective.

1.2.1 Communication architectures

The interconnection network of a digital system, which connects the system modules to each other, has a profound impact on the system's capabilities, performance, size, and cost. Several interconnection schemes have been heavily studied and are used in many systems, including point-to-point wires, multistage interconnection networks, and shared busses. Because of the costs associated with wiring and packaging, it is generally desirable to minimize the number of wires in a system and the number of connections per module.

Chapter 2 of this thesis investigates how busses (multiple-pin wires) can be employed to efficiently realize certain communication patterns among modules in a digital system. We concentrate on the problem of efficiently permuting data stored in VLSI chips (modules) in accordance with a predetermined set of permutations. We show that by connecting modules with shared bus interconnections, as opposed to point-to-point interconnections, the number of pins per module can often be significantly reduced.

Much research has focused on implementing permutations and various other communication patterns on different interconnection networks. By using point-to-point wires, for example, any communication pattern can be realized in one communication cycle. For rich and diverse communication patterns, however, full point-to-point interconnections tend to use many wires and many connections per module, since any two modules that need to communicate must share a wire. (See [60, 83] for VLSI costs of point-to-point interconnection schemes.) Multistage interconnection networks have also been heavily investigated for the purpose of realizing general communication patterns and more specifically for routing permutations (see [6, 7, 27, 32, 37, 52, 53, 55, 74, 75, 86]). Many multistage interconnection networks exhibit logarithmic number of stages and constant number of connections per module. However, the savings in the number of pins per module come at the expense of realizing permutations in logarithmic number of communication cycles and the use of a considerable amount of switching hardware. The use of busses as the interconnection infrastructure for realizing communication patterns has also been examined by several researchers (see [10, 13, 30, 64, 73, 77]). In this thesis we demonstrate that bussed interconnections can be employed for realizing general classes of permutations in one communication cycle, with considerably small number of pins per module, and with virtually no switching and controlling hardware.

In Chapter 2, we exhibit bussed permutation architectures for many classes of permutation sets. For example, we present permutation architectures that with $O(\sqrt{n})$ pins per module can realize any of the n cyclic shifts on n modules in one communication cycle. Our results are derived from a mathematical characterization of *uniform permutation architectures* based on the combinatorial notion of a *difference cover*. We extend our discussion to permutation groups and show that when the set of permutations forms a group with p elements, any permutation in the group can be realized in one communication cycle by a uniform architecture with $O(\sqrt{p \lg p})$ pins per module. Furthermore, when the permutation group is abelian, we show that $O(\sqrt{p})$ pins per module suffice. We also consider uniform permutation architectures that realize permutations in several communication cycles, instead of one, and show that further savings in the number of pins per module can be obtained. Finally, we identify many permutation networks that can benefit from our methodology of using difference covers for designing uniform architectures, including hypercubes, multidimensional meshes, and shuffle-exchange networks.

1.2.2 Control mechanisms

Large digital systems use control mechanisms for several functions, including establishing timing disciplines, triggering events, and sequencing transactions. The complexity of a large digital system generally calls for the separation of the control mechanisms from the communication and computation structures. Description of control mechanisms for digital systems appear in [20, 88], for bus systems in [16, 22, 40, 57, 82], and for communication networks in [12, 61, 78].

Chapter 3 of this thesis explores the problem of arbitrating among modules in a digital system. Many arbitration mechanisms have been developed that use daisy chains, centralized priority circuits, polling mechanisms, token passing schemes, and carrier sense multiple access protocols, among others (see [12, 16, 22, 40, 45, 46, 57, 61, 78, 82, 88]). We focus on distributed priority arbitration mechanisms, where contention is resolved using predetermined module priorities and arbitration processes are carried out in a distributed manner by system modules. Distributed priority arbitration mechanisms are used in many modern systems, including numerous multiprocessors and data communication networks. Specifically, we investigate arbitration mechanisms that employ dedicated arbitration busses and use asynchronous globally or locally timed combinational logic. Several other studies of bus-based arbitration mechanisms appear in [3, 22, 23, 24, 47, 71, 79, 80, 81].

In Chapter 3, we examine distributed asynchronous priority arbitration mechanisms that arbitrate among n modules using m arbitration busses, for $\lg n \leq m \leq n$. A widely used distributed asynchronous mechanism is the *binary arbitration* scheme [79], which with $m = \lg n$ busses arbitrates in $t = \lg n$ units of time. We present a new asynchronous scheme — *binomial arbitration* — that by using $m = \lg n + 1$ busses reduces the arbitration time to $t = \frac{1}{2}\lg n$. Extending this result, we present the *generalized binomial arbitration* scheme that achieves a bus-time tradeoff of the form $m = \Theta(tn^{1/t})$, between the number of arbitration busses m and the arbitration time t (in units of bus-settling delay), for values of $\lg n \leq m \leq n$ and $1 \leq t \leq \lg n$. Our schemes are based on a novel analysis of *data-dependent delays*. Most importantly, our schemes can be adopted with no changes to existing hardware and protocols; they merely involve selecting a *good* set of priority arbitration codewords. We also investigate the capabilities of general asynchronous priority arbitration schemes that employ busses and present some lower bound arguments that demonstrate the efficiency of our schemes.

1.2.3 Transmission lines

The speed of information transfer through a communication medium is bounded by several physical properties of the medium. Different media such as radio broadcast channels, electrical wires, and optical fibers have different propagation speeds, but they can all be modeled essentially in the same manner. In any communication system, the information sent by a module requires time to propagate and reach other modules. Communication protocols must, therefore, account for signal propagation by incorporating appropriate time intervals.

In Chapters 3 and 4, we investigate how propagation delays of digital signals on electrical busses can influence the design of communication protocols. The propagation of a signal on an electrical bus depends on the length, shape, and other properties of the bus. A high-speed bus is modeled as an analog transmission line with associated impedance that determines the propagation speed of signals along it (see [5, 40]). Most computer systems, however, use the digital abstraction, which specifies certain discrete voltage levels for representing logic values. When designing communication protocols for electrical busses, signal propagation delays must be accounted for, as done, for example, in Ethernet [63]. A common method of dealing with different and unpredictable propagation delays on a shared medium is to allow sufficient time for the propagation of signals from the furthest module in the system and for the settlement of the communication medium. This approach is explored in Chapter 3, where the time required by bus-based arbitration mechanisms to stabilize is measured in units of bus-settling delay. The unit of a bus-settling delay is an upper bound on the time that an electrical bus resolves various transient effects and reaches a valid logic value. In Chapter 4, on the other hand, we investigate a more elaborate model of a bus as a digital transmission line, which takes into account propagation of signals along a bus line but ignores the analog nature of the signals.

In Chapter 4, we examine the performance of priority arbitration schemes presented in Chapter 3 under the *digital transmission line* bus model. This bus model accounts for the propagation time of signals along bus lines and assumes that the propagating signals are always valid digital signals. A widely held misconception is that in the digital transmission line model the arbitration time of the binary arbitration scheme is at most 4 units of bus-propagation delay. We formally disprove this conjecture by demonstrating that the arbitration time of the binary arbitration scheme is heavily dependent on the arrangement of the arbitrating modules in the

system. We provide a general scenario of module arrangement on m busses, for which binary arbitration takes at least $m/2$ units of bus-propagation delay to stabilize. We also prove that for general arrangements of modules on m busses, binary arbitration settles in at most $m/2+2$ units of bus-propagation delay, while binomial arbitration settles in at most $m/4+2$ units of bus-propagation delay, thereby demonstrating the superiority of binomial arbitration for general arrangements of modules under the digital transmission line model. For linear arrangements of modules in increasing order of priorities and equal spacings between modules, we show that 3 units of bus-propagation delay are necessary for binary arbitration to settle, and we sketch an argument that 3 units of bus-propagation delay are also asymptotically sufficient.

Chapter 2

Bussed Permutation Architectures

This chapter explores the problem of efficiently permuting data stored in VLSI chips in accordance with a predetermined set of permutations. By connecting chips with bussed interconnections, as opposed to point-to-point interconnections, we show that the number of pins per chip can often be reduced. For example, for infinitely many n, we exhibit permutation architectures with $\lceil \sqrt{n} \rceil$ pins per chip that can realize any of the n cyclic shifts on n chips in one clock tick. When the set of permutations forms a group with p elements, any permutation in the group can be realized in one clock tick by an architecture with $O(\sqrt{p \ln p})$ pins per chip. When the permutation group is abelian, we show that $O(\sqrt{p})$ pins suffice. These results are all derived from a mathematical characterization of *uniform permutation architectures* based on the combinatorial notion of a *difference cover*. We investigate properties of difference covers and describe procedures for designing efficient difference covers for many classes of permutation sets. We also consider uniform permutation architectures that realize permutations in several clock ticks, instead of one, and show that further savings in the number of pins per chip can be obtained. Our methodology of using difference covers for designing efficient uniform architectures is applicable to a wide range of permutation networks, including hypercubes, multidimensional meshes, and shuffle-exchange networks.

This chapter describes joint research with Joe Kilian and Charles Leiserson [48] and [49].

2.1 Introduction

The organization of communication among chips is a major concern in the design of an electronic system. Because of the costs associated with wiring and packaging, it is generally desirable to minimize the number of wires and the number of pins per chip in an architecture. Much research has focused on point-to-point and multistage interconnections (see [6, 7, 27, 37, 75, 86]). In this chapter, we investigate how busses can be employed to efficiently implement various communication patterns among a set of chips. Other studies of bussed interconnection schemes for realizing communication patterns can be found in [10, 11, 13, 30, 54, 64, 77].

Perhaps the simplest example of the advantage of bussed interconnections is the use of a single shared bus to communicate between any pair of chips connected to the bus in one clock tick. Communicating between any pair of chips in one clock tick can be implemented with two-pin wires, but any such scheme requires $\binom{n}{2}$ wires and $n-1$ pins per chip, where n is the number of chips in the system.[1] Of course, a two-pin (point-to-point) interconnection scheme may be able to implement more communication patterns, but if we are only interested in communication between individual pairs, the additional power, which comes at a high cost, is wasted.

An example that better illustrates the ideas in this chapter comes from the problem of building a fast *cyclic shifter* (sometimes called a *barrel shifter*) on n chips. Initially, each chip c contains a one-bit value ϵ_c. The function of the shifter is to move each bit ϵ_c to chip $c+s \pmod n$ in one clock tick, where s can be any value between 0 and $n-1$.

Any cyclic shifter that uses only two-pin wires requires at least $\binom{n}{2}$ wires and $n-1$ pins per chip in order to shift in one clock tick because each chip must be able to communicate directly with each of the other $n-1$ chips. Using busses, however, we can do much better. Figure 2-1 gives an architecture for a cyclic shifter on 13 chips which uses 13 busses and only 4 pins per chip. To realize a shift by 8, for example, each chip writes its bit to pin 3 and reads from pin 1. The reader may verify that all other cyclic shifts among the chips are possible in one clock tick. (In Section 2.4, we give a general method for constructing such cyclic shifters based on finite projective planes.)

[1] Unless otherwise specified, we count only data pins in our analysis and omit consideration of the pins for control, clock, power, and ground since they are needed by all implementations.

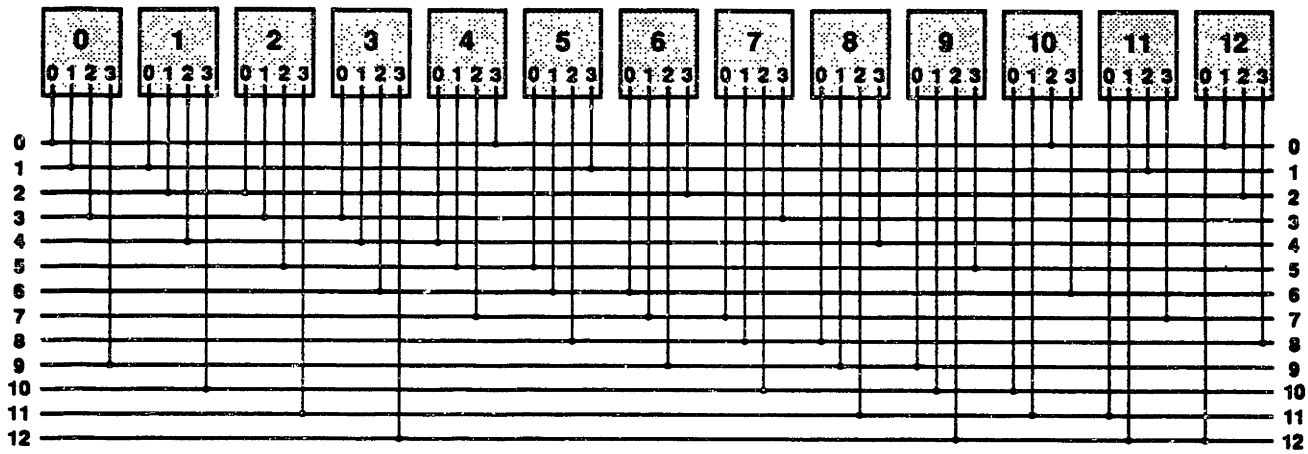

Figure 2-1: A cyclic shifter on 13 chips that uses 13 busses. Each chip has 4 pins, and each bus has 4 chips connected to it. This cyclic shifter is based on the difference cover $\{0, 1, 3, 9\}$ for $\mathbf{Z}_{13}$.

The cyclic shifter of Figure 2-1 has the advantage of uniformity. All chips have exactly the same number of pins, and to accomplish each of the 13 permutations specified by the problem, all chips write to (and read from) pins with identical labels. For all busses, the number of pins per bus is 4, which is the same as the number of pins per chip. Moreover, the connections between chips and busses follow a periodic pattern. The uniformity of the architecture leads to simplicity in the control of the system. Four control wires from a central controller are sufficient to determine each of the 13 shifts—two wires for specifying the number of the pin on which to write, and two for the pin to read—which is the minimum possible. Thus, our control scheme uses the minimum number of control pins, and the on-chip decoding logic is straightforward and identical for all the chips.

Cyclic shifters for general n can be constructed using an idea from combinatorial mathematics related to difference sets [43, p. 121]. (See also [14, 34, 38, 56, 66].)

Definition 1 A subset $D \subseteq \mathbf{Z}_n$ of the integers modulo n is a *difference cover for* $\mathbf{Z}_n$ if for all $s \in \mathbf{Z}_n$, there exist $d_i, d_j \in D$ such that $s = d_i - d_j \pmod{n}$.

That is, every integer in $\mathbf{Z}_n$ can be represented as the difference modulo n of two integers in D. For example, the set $D = \{0, 1, 3, 9\}$ is a difference cover for $\mathbf{Z}_{13}$, since

$$
\begin{aligned}
0 &= 0-0 \\
1 &= 1-0 \\
2 &= 3-1 \\
3 &= 3-0 \\
4 &= 0-9 \\
5 &= 1-9 \\
6 &= 9-3 \\
7 &= 3-9 \\
8 &= 9-1 \\
9 &= 9-0 \\
10 &= 0-3 \\
11 &= 1-3 \\
12 &= 0-1\,,
\end{aligned}
$$

where all subtractions are performed modulo 13.

Given a difference cover for $\mathbf{Z}_n$ with k elements, a cyclic shifter on n chips with n busses and k pins per chip can be constructed. Suppose $D = \{d_0, d_1, \ldots, d_{k-1}\}$ is a difference cover for $\mathbf{Z}_n$. In the cyclic shifter, chip c connects via its pin i to bus $c + d_i \pmod n$, for all $c = 0, 1, \ldots, n-1$ and $i = 0, 1, \ldots, k-1$. To see that any cyclic shift on the n chips can be uniformly realized, consider a cyclic shift by s. Since D is a difference cover for $\mathbf{Z}_n$, there exist $d_i, d_j \in D$ such that $s = d_i - d_j \pmod n$. To realize the shift by s, each chip writes to pin i and reads from pin j. Chip c therefore writes onto bus $c + d_i$, and bus $c + d_i$ is read by chip $(c + d_i) - d_j = c + s$. No collisions occur because each bus has exactly one pin labeled i and one pin labeled j connected to it, as can be verified.

The remainder of this chapter explores permutation architectures, the properties of multiple-pin interconnections, and related combinatorial mathematics. In Section 2.2, we define a permutation architecture, introduce the notion of uniformity, and prove some basic properties of architectures that employ busses to realize arbitrary sets of permutations. Section 2.3 defines

the notion of a difference cover for a set of permutations, relates it to the notion of a uniform permutation architecture, and proves some properties of difference covers. In Section 2.4, we show how to build cyclic shifters that are provably efficient. Section 2.5 investigates how to design small difference covers for any set of permutations that forms a finite group. In Section 2.6, we extend the discussion to uniform architectures that realize permutations in more than one clock tick. Several applications and extensions of bussed permutation architectures are discussed in Section 2.7, as well as further research and some questions left open by our research.

2.2 Permutation architectures

In this section we formally define the notion of a permutation architecture, and we make precise the notion of uniformity. We also prove some basic properties of permutation architectures that realize arbitrary sets of permutations. The definitions in this section are somewhat intricate and tedious, and are indicative of the difficulties faced in the design of efficient permutation architectures. In the next section, however, we use these definitions to show that reasoning about uniform permutation architectures is essentially equivalent to reasoning about difference covers, a simpler and more elegant mathematical notion. The remainder of this chapter then uses the simpler notion.

For convenience, we adopt a few notational conventions. We use multiplicative notation to denote composition of permutations. The inverse of a permutation π is denoted by π^{-1}. Composition of functions is performed in right-to-left order, so that $\pi_1\pi_2$ is defined by $\pi_1\pi_2 x = \pi_1(\pi_2(x))$. The identity permutation on n elements is denoted by I_n, or by I if the number of elements is unimportant. For a permutation set Φ, we denote by Φ^{-1} the set of all the inverses of the permutations of Φ, i.e., $\Phi^{-1} = \{\phi^{-1} : \phi \in \Phi\}$. For two permutation sets Φ and Ψ, the notation $\Phi\Psi$ is used to denote the permutation set $\{\phi\psi : \phi \in \Phi \text{ and } \psi \in \Psi\}$. We use the notation $[n]$ to denote the set of n integers $\{0, 1, \ldots, n-1\}$.

2.2.1 What is a permutation architecture?

We begin by formally defining the notion of a permutation architecture.

Definition 2 A *permutation architecture* is a 6-tuple $\mathcal{A} = \langle C, B, P, \text{CHIP}, \text{BUS}, \text{LABEL} \rangle$ as follows.

1. C is a set of *chips*;
2. B is a set of *busses*;
3. P is a set of *pins*;
4. CHIP is a function $\text{CHIP} : P \to C$;
5. BUS is a function $\text{BUS} : P \to B$;
6. LABEL is a function $\text{LABEL} : P \to \mathbf{N}$, where if $x, y \in P$, $x \neq y$, and $\text{CHIP}(x) = \text{CHIP}(y)$, then $\text{LABEL}(x) \neq \text{LABEL}(y)$.

The set C contains all the chips in the architecture, and the set B contains all the busses. Which chips are connected to which busses is determined by the pins they have in common; the set P contains all the pins. The function CHIP determines which pins belong to which chips. Similarly, the function BUS determines which pins are interconnected by which bus. The function LABEL names the pins on the chips by natural numbers such that all pins on a given chip have distinct labels, which we shall sometimes call pin numbers.

Our formal definition of a permutation architecture omits several subsystems that technically should be included, but whose inclusion is not germane to our study. These subsystems include a control network that specifies what permutation is to be performed and clocking circuitry for synchronization. Our focus is on the structure of the bussed interconnections for permuting the data, and thus our definition encompasses only this aspect of the architecture.

We now define what it means for a permutation architecture to realize a permutation.

Definition 3 A permutation architecture $\mathcal{A} = \langle C, B, P, \text{CHIP}, \text{BUS}, \text{LABEL} \rangle$ *realizes* a permutation $\pi : C \to C$ if there exist two functions $\text{WRITE}_\pi : C \to P$ and $\text{READ}_\pi : C \to P$, such that for any chips $c, c_1, c_2 \in C$, we have:

1. $\text{CHIP}(\text{READ}_\pi(c)) = \text{CHIP}(\text{WRITE}_\pi(c)) = c$;
2. $\text{BUS}(\text{WRITE}_\pi(c)) = \text{BUS}(\text{READ}_\pi(\pi(c)))$;

3. $c_1 \neq c_2$ implies $\text{BUS}(\text{WRITE}_\pi(c_1)) \neq \text{BUS}(\text{WRITE}_\pi(c_2))$.

The architecture *uniformly realizes* π if, in addition:

4. $\text{LABEL}(\text{WRITE}_\pi(c_1)) = \text{LABEL}(\text{WRITE}_\pi(c_2))$;

5. $\text{LABEL}(\text{READ}_\pi(c_1)) = \text{LABEL}(\text{READ}_\pi(c_2))$.

We say that a permutation architecture *realizes* a set Π of permutations if it realizes every permutation in Π. We say that it *uniformly realizes* Π if it uniformly realizes every permutation in Π.

Intuitively, for a permutation π, the functions WRITE_π and READ_π identify the *write pin* and the *read pin* for each chip. Condition 1 makes sure that each chip writes and reads pins that are connected to it. Condition 2 ensures that the bus to which chip c writes is read by chip $\pi(c)$. Condition 3 guarantees that no collisions occur, that is, no two data transfers use the same bus. The architecture uniformly realizes a permutation (Conditions 4 and 5) if all chips write to pins with the same pin number and read from pins with the same pin number, as in the cyclic shifter from Figure 2-1.

Our definition of a permutation architecture implies that "complete" permutations are to be realized, that is, every chip sends exactly one datum and receives exactly one datum. Moreover, an interconnection is required even when a chip sends a datum to itself. Since no collisions occur, the number of busses in the architecture must be at least the number of chips. This observation leads directly to the following theorem.

Theorem 1 *In any permutation architecture that realizes some nonempty permutation set* Π, *the average number of pins per bus is at most the average number of pins per chip.*

Proof. Let $\mathcal{A} = \langle C, B, P, \text{CHIP}, \text{BUS}, \text{LABEL}\rangle$ be a permutation architecture for Π. The average number of pins per chip is $|P| / |C|$, and the average number of pins per bus is $|P| / |B|$. Condition 3 of Definition 3 says that for any permutation $\pi \in \Pi$, any two distinct chips are mapped to distinct busses. Consequently, we get that $|B| \geq |C|$, which proves the theorem. ■

Under the assumption that no interconnection is needed for a chip to send data to itself, Theorem 1 is no longer applicable. A similar theorem can be proved for this model, however,

which involves the number of fixed points in the permutations realized by the architecture. Specifically, suppose the architecture realizes a set Π of permutations. Define the *rank* of a permutation $\pi \in \Pi$ as $\text{RANK}(\pi) = |\{c \in C : \pi(c) \neq c\}|$, and define the rank of the permutation set Π as $\text{RANK}(\Pi) = \max_{\pi \in \Pi} \text{RANK}(\pi)$. The analogue to Theorem 1 states that the ratio between the average number of pins per bus and the average number of pins per chip is at most $|C| / \text{RANK}(\Pi)$.

2.2.2 Uniform permutation architectures

In any architecture $\mathcal{A}$ that uniformly realizes a permutation set Π, the number of pins that are actually used to uniformly realize Π is the same for all chips, and additional pins on a chip are unused. Furthermore, the number of busses used in realizing any permutation $\pi \in \Pi$ is equal to the number of chips. These observations lead to the following definition of a uniform architecture.

Definition 4 A *uniform permutation architecture* for a permutation set Π is a permutation architecture $\mathcal{A} = \langle C, B, P, \text{CHIP}, \text{BUS}, \text{LABEL} \rangle$ such that:

1. $\mathcal{A}$ uniformly realizes Π;
2. $|\{x \in P : \text{CHIP}(x) = c_1\}| = |\{x \in P : \text{CHIP}(x) = c_2\}|$ for any two chips $c_1, c_2 \in C$;
3. $|B| = |C|$;
4. if $x \neq y$ and $\text{LABEL}(x) = \text{LABEL}(y)$, then $\text{BUS}(x) \neq \text{BUS}(y)$.

Thus, all the chips in a uniform permutation architecture have the same number of pins (Condition 2), the number of busses is equal to the number of chips (Condition 3), and the labels of the pins on any bus are distinct (Condition 4).

The following theorem demonstrates that any permutation architecture that uniformly realizes some permutation set Π can be made into a uniform architecture for Π.

Theorem 2 *Let $\mathcal{A} = \langle C, B, P, \text{CHIP}, \text{BUS}, \text{LABEL} \rangle$ be a permutation architecture that uniformly realizes the permutation set Π, and let k be the smallest number of pins on any chip in C. Then there is a uniform architecture $\mathcal{A}' = \langle C', B', P', \text{CHIP}', \text{BUS}', \text{LABEL}' \rangle$ for Π with at most k pins per chip.*

Proof. We construct the uniform architecture $\mathcal{A}'$ from the permutation architecture $\mathcal{A}$ in two steps. First, we construct an intermediate permutation architecture $\mathcal{A}'' = \langle C'', B'', P'', \text{CHIP}'', \text{BUS}'', \text{LABEL}'' \rangle$ by removing extraneous pins from chips in $\mathcal{A}$ such that all chips end up with the same number of pins per chip and such that each pin plays a role in uniformly realizing Π. Then, the busses of $\mathcal{A}''$ are reorganized to produce the architecture $\mathcal{A}'$ in such a way that the number of busses in $\mathcal{A}'$ is equal to the number of chips. We assume that the permutation set Π is nonempty, since otherwise the theorem is trivial.

In the first step, we remove pins that are unused in uniformly realizing Π. Since $\mathcal{A}$ uniformly realizes Π, each permutation $\pi \in \Pi$ can be associated with a distinct pair (i, j) of pin labels corresponding to the labels that all chips write to and read from in order to realize π. A pin is unused if its label does not appear in any of these $|\Pi|$ pairs. Removing the unused pins results in the architecture $\mathcal{A}''$ in which all chips have the same number of pins, since each chip has exactly one pin for each label used in uniformly realizing Π. The permutation architecture $\mathcal{A}''$ uniformly realizes Π, and furthermore, each pin is used in uniformly realizing some $\pi \in \Pi$. If we let s denote the number of pins per chip in $\mathcal{A}''$, then we have $s \leq k$, since originally at least one chip had k pins and no pins were added.

In the second step, we reorganize the busses of $\mathcal{A}''$ to produce the uniform architecture $\mathcal{A}'$ in which the number of busses is equal to the number of chips. For any permutation architecture that realizes a nonempty permutation set, the number of busses is never smaller that the number of chips. Assume without loss of generality that $C'' = [n]$, $B'' = [m]$, and $\text{range}(\text{LABEL}'') = [s]$. The theorem is proved if the architecture $\mathcal{A}''$ uses only $n = |C''|$ busses, but in general, the architecture might use $m > n$ busses.

We define a collection of mappings $\Psi = \{\psi_0, \psi_1, \ldots, \psi_{s-1}\}$, where for each $0 \leq i \leq s-1$, the mapping $\psi_i : [n] \to [m]$ is defined to be $\psi_i(c) = b$ if and only if chip $c \in C''$ is connected via its pin number i to bus $b \in B''$. The elements of Ψ are indeed mappings since each chip has a pin numbered i for each $0 \leq i \leq s-1$. The mappings are injective (one-to-one), since otherwise two pins with the same pin number would be connected to the same bus, and both pins could not be used to uniformly realize permutations, thereby violating the construction of $\mathcal{A}''$ in the first step. The collection Ψ is a multiset, since it may be that two different pin numbers $i \neq j$ define the same mapping (i.e., $\psi_i = \psi_j$). The key idea is that any permutation

is implemented by each chip writing to pin i and reading from pin j, thereby employing the mapping ψ_i to write data from the n chips to n distinct busses, and the inverse of the mapping ψ_j to read data from the same n busses back to the n chips.

We now show how to reorganize the busses of $\mathcal{A}''$ in order to construct a uniform architecture $\mathcal{A}'$. We partition Ψ into l equivalence classes $\Psi_0 \cup \Psi_1 \cup \cdots \cup \Psi_{l-1}$ such that ψ_i and ψ_j are in the same equivalence class Ψ_r, if and only if range(ψ_i) = range(ψ_j). This partitioning has the property that if $\pi \in \Pi$, then there exists an r such that $\pi = \psi_j^{-1}\psi_i$ where $\psi_i, \psi_j \in \Psi_r$. (Recall that the inverse of an injective mapping $\psi : [n] \rightarrow [m]$ is defined as the mapping ψ^{-1} : range(ψ) $\rightarrow [n]$ such that if $\psi(c) = b$, then $\psi^{-1}(b) = c$.) For each $0 \leq r \leq l-1$, pick a bijection (one-to-one, onto) f_r : range(ψ) $\rightarrow [n]$, where ψ is any mapping in Ψ_r. (We can pick a bijection, since ψ is injective, which implies $|\text{range}(\psi)| = n$.) We define the architecture $\mathcal{A}'$ by $C' = C''$, $B' = [n]$, $P' = P''$, CHIP$'$ = CHIP$''$, LABEL$'$ = LABEL$''$, and for any pin $x \in P'$ such that $\psi_{\text{LABEL}'(x)} \in \Psi_r$, we define BUS$'(x) = f_r(\text{BUS}''(x))$.

The architecture $\mathcal{A}'$ has exactly s pins per chip and satisfies $|B'| = |C'| = n$, thereby satisfying Conditions 2 and 3 of Definition 4. We show Condition 4 holds by considering any two pins x and y with LABEL$'(x)$ = LABEL$'(y) = i$. We have BUS$'(x) = f_r(\text{BUS}''(x))$ and BUS$'(y) = f_r(\text{BUS}''(y))$ for some f_r as defined in the previous paragraph. Since f_r is an injective mapping and because Condition 4 of Definition 4 holds for $\mathcal{A}''$, we then have $x \neq y$ implies BUS$'(x) \neq$ BUS$'(y)$.

It remains to show that Condition 1 of Definition 4 holds, that is, that $\mathcal{A}'$ uniformly realizes Π. Consider any permutation $\pi \in \Pi$. Since $\mathcal{A}''$ uniformly realizes Π, there exists a pair of pin labels (i, j) such that π is realized in $\mathcal{A}''$ by each chip writing to its pin numbered i and reading from its pin numbered j. We use the same pin labels (i, j) to realize the permutation π in $\mathcal{A}'$. Conditions 1, 4, and 5 of Definition 3 are immediately satisfied. To verify Conditions 2 and 3 we use the following observation. In architecture $\mathcal{A}''$ chip c is connected via its pin labeled h to bus $\psi_h(c)$, while in architecture $\mathcal{A}'$ it is connected to bus $f_r(\psi_h(c))$, where $\psi_h \in \Psi_r$. Condition 2 now holds since $\pi = \psi_j^{-1}\psi_i = (f_r\psi_j)^{-1}(f_r\psi_i)$. Condition 3 holds since $f_r\psi_i$ is a permutation on $[n]$. We therefore conclude that $\mathcal{A}''$ is a uniform architecture for Π with at most k pins per chip. ■

2.2.3 Some properties of uniform architectures

From the definition of uniform permutation architectures one can derive several structural properties of these architectures. The next theorem provides a lower bound on the number of pins per chip in any uniform architecture for a permutation set Π.

Theorem 3 *Let $\mathcal{A} = \langle C, B, P, \text{CHIP}, \text{BUS}, \text{LABEL}\rangle$ be a uniform permutation architecture for a permutation set Π. Then the number of pins per chip in $\mathcal{A}$ is at least $\sqrt{|\Pi|}$.*

Proof. Because architecture $\mathcal{A}$ realizes Π uniformly, we can associate each $\pi \in \Pi$ with a pair (i, j) of pin numbers such that π is realized by each chip writing to its pin labeled i and reading from its pin labeled j. Since $\mathcal{A}$ is uniform, each chip has exactly $|P| / |C|$ pins, and the number of such pairs is $(|P| / |C|)^2$. No two permutations can be associated with the same pair, and thus, we have $(|P| / |C|)^2 \geq |\Pi|$ or $|P| / |C| \geq \sqrt{|\Pi|}$. ■

Another observation made by Fiduccia [28] involves the maximal number of chips reachable in one clock tick from any given chip in a uniform architecture. (See also [48, p. 308].)

Theorem 4 *Any uniform permutation architecture with k pins per chip has exactly k pins per bus, and each chip is connected to at most $k(k-1)$ other chips.*

Proof. If there is a bus with more than k pins, then two pins on the bus must have the same label, contradicting Condition 4 of Definition 4. Now, since for uniform architectures the number of busses is equal to the number of chips, each bus must have exactly k pins. Moreover, since any chip is connected to at most k different busses (via its k pins), each of which is connected to no more than $k-1$ other chips, the number of neighbors of a chip is at most $k(k-1)$. ■

A permutation architecture can often nonuniformly realize many more permutations than the square of the number of pins per chip. As an example, consider a "crossbar" architecture of n chips and n busses where each chip is connected to each bus. This architecture can nonuniformly realize all $n!$ permutations, which is much greater than n^2, the square of the number of pins per chip. In Section 2.7.3 we discuss some of the capabilities of nonuniform permutation architectures.

2.3 Difference covers

In this section, we present our main theorems which establish the relationship between difference covers for permutation sets and uniform permutation architectures. We also prove some theorems concerning the design of general difference covers and difference covers for Cartesian products of permutation sets. Finally, we present an alternative representation for difference covers called substring covers based on similar notions in the literature of difference sets.

2.3.1 Difference covers and uniform architectures

We first provide a generalization of Definition 1 to arbitrary sets of permutations.

Definition 5 A *difference cover* for a permutation set Π is a set $\Phi = \{\phi_0, \phi_1, \ldots, \phi_{k-1}\}$ of permutations such that for each $\pi \in \Pi$ there exist $\phi_i, \phi_j \in \Phi$ such that $\pi = \phi_j^{-1}\phi_i$.

Equivalently, we can use our product-of-sets notation to say that Φ is a difference cover for Π if $\Phi^{-1}\Phi \supseteq \Pi$.

The following theorems show how difference covers and uniform architectures are related. Theorem 5 describes how to design a uniform architecture for a permutation set Π when a difference cover for Π is given. Theorem 6 presents a construction of a difference cover for a permutation set Π from a uniform architecture for Π.

Theorem 5 *Let Π be a permutation set, and let Φ be a difference cover for Π such that $|\Phi| = k$. Then there exists a uniform architecture for Π with k pins per chip.*

Proof. Let $\Phi = \{\phi_0, \phi_1, \ldots, \phi_{k-1}\}$, and assume that Π is a set of permutations on n objects. We construct a permutation architecture for Π with n busses and k pins per chip. We name the chips and busses of the architecture by natural numbers, and the pins by pairs of natural numbers. The architecture $\mathcal{A} = \langle C, B, P, \text{CHIP}, \text{BUS}, \text{LABEL}\rangle$ is defined as $C = [n]$, $B = [n]$, $P = [n] \times [k]$, $\text{CHIP}(c, i) = c$, $\text{LABEL}(c, i) = i$, and $\text{BUS}(c, i) = \phi_{\text{LABEL}(c,i)}(\text{CHIP}(c, i)) = \phi_i(c)$. That is, chip c is connected via its pin number i to bus $\phi_i(c)$.

To see formally that this architecture uniformly realizes Π, let $\pi \in \Pi$ be a permutation, and let $\phi_i, \phi_j \in \Phi$ be elements of the difference cover for Π such that $\pi = \phi_j^{-1}\phi_i$. Define the write function for π as $\text{WRITE}_\pi(c) = (c, i)$ and define the read function for π as $\text{READ}_\pi(c) = (c, j)$.

(Note that i and j are always in the range 0 through $k-1$.) We now verify that the five Conditions of Definition 3 are satisfied. Condition 1 holds since for any chip $c \in C$ we have $\text{CHIP}(\text{WRITE}_\pi(c)) = \text{CHIP}(c,i) = c$, and $\text{CHIP}(\text{READ}_\pi(c)) = \text{CHIP}(c,j) = c$. Condition 2 is satisfied since for any chip $c \in C$ we have

$$\begin{aligned}
\text{BUS}(\text{WRITE}_\pi(c)) &= \text{BUS}(c,i) \\
&= \phi_i(c) \\
&= \phi_j\phi_j^{-1}\phi_i(c) \\
&= \phi_j(\pi(c)) \\
&= \text{BUS}(\pi(c),j) \\
&= \text{BUS}(\text{READ}_\pi(\pi(c))).
\end{aligned}$$

Condition 3 holds because if $\text{BUS}(\text{WRITE}_\pi(c_1)) = \text{BUS}(\text{WRITE}_\pi(c_2))$ for any two chips $c_1, c_2 \in C$, then we have $\phi_i(c_1) = \phi_i(c_2)$, which implies that $c_1 = c_2$, since ϕ_i is invertible. Conditions 4 and 5 both hold since $\text{LABEL}(\text{WRITE}_\pi(c)) = i$ and $\text{LABEL}(\text{READ}_\pi(c)) = j$ for all chips $c \in C$. We therefore conclude that the architecture $\mathcal{A}$ uniformly realizes Π. The architecture is uniform, but Theorem 2 obviates the need to show this fact. ■

Given a difference cover of small cardinality, Theorem 5 says we can construct a uniform architecture with few pins per chip. In fact, the reverse is true as well, as the following theorem shows.

Theorem 6 *Let Π be a permutation set, and let $\mathcal{A}$ be a uniform architecture for Π with k pins per chip. Then Π has a difference cover Φ such that $|\Phi| \leq k$.*

Proof. Given a uniform architecture $\mathcal{A} = \langle C, B, P, \text{CHIP}, \text{BUS}, \text{LABEL}\rangle$ for the permutation set Π, where k is the number of pins on each chip, we construct a difference cover Φ for Π as follows. Assume without loss of generality that $C = B = [n]$ and $\text{range}(\text{LABEL}) = [k]$. For each pin number i, where $i = 0, 1, \ldots, k-1$, we define ϕ_i by $\phi_i(c) = b$ if and only if chip c is connected via its pin number i to bus b. We now define the difference cover Φ to be the set $\Phi = \{\phi_0, \phi_1, \ldots, \phi_{k-1}\}$. (The set Φ may have less than k elements, since some permutations may be repeated among the ϕ_i's.)

To see that Φ is a difference cover for Π, consider any permutation $\pi \in \Pi$. Since $\mathcal{A}$ uniformly realizes π, there exists a pair of pin labels (i,j) such that π is realized by each chip writing to its pin numbered i and reading from its pin numbered j. The labels i and j satisfy $i = \text{LABEL}(\text{WRITE}_\pi(c))$ and $j = \text{LABEL}(\text{READ}_\pi(c))$ for all chips $c \in C$, as follows from Conditions 4 and 5 of Definition 3. Conditions 1 and 3 of Definition 3 imply that ϕ_i and ϕ_j are both permutations, and therefore there are $\phi_h, \phi_l \in \Phi$ such that $\phi_h = \phi_i$ and $\phi_l = \phi_j$. Finally, Condition 2 of Definition 3 implies that $\pi = \phi_j^{-1}\phi_i = \phi_l^{-1}\phi_h$, which proves that Φ is indeed a difference cover for Π. ■

2.3.2 Designing difference covers

Theorems 5 and 6 show that uniform architectures and difference covers are very closely related. Thus, when designing a uniform permutation architecture for a set of permutations, it suffices to focus on the problem of constructing a good difference cover for that set.

We first present a simple theorem that demonstrates that any arbitrary permutation set Π has a difference cover of size at most $|\Pi| + 1$.

Theorem 7 *Let Π be an arbitrary permutation set on n elements. Then Π has a difference cover of size at most $|\Pi| + 1$.*

Proof. Define $\Phi = \Pi \cup \{I_n\}$. For any $\pi \in \Pi$, we have $\pi = I_n^{-1}\pi$, where $\pi, I_n \in \Phi$. Therefore, Φ is a difference cover for Π, and $|\Phi| \leq |\Pi| + 1$. ■

Theorem 7 presents a naive construction of a difference cover for an arbitrary permutation set Π. In general, the bound of Theorem 7 cannot be improved without specific knowledge about the structure of the permutation set involved. In [30], Fiduccia describes how to construct a permutation set Π of arbitrary size, for which no difference cover of cardinality $|\Pi|$ exists. This shows that the construction of Theorem 7 is optimal for general permutation sets.

Specific knowledge about the structure of a permutation set can indeed be helpful in obtaining a small difference cover for it. In Sections 2.4 and 2.5, we investigate the construction of difference covers for cyclic groups of permutations and for groups in general. Here, we examine permutation sets formed by Cartesian products.

Definition 6 Let Π_1 be a set of permutations from X_1 to X_1, and let Π_2 be a set of permutations from X_2 to X_2. The *Cartesian product* $\Pi = \Pi_1 \times \Pi_2$ is the set of permutations from $X_1 \times X_2$ to $X_1 \times X_2$ defined as $\Pi = \{(\pi_1, \pi_2) : \pi_1 \in \Pi_1, \pi_2 \in \Pi_2\}$. Operations on the elements of Π are performed componentwise.

The Cartesian product $\Pi_1 \times \Pi_2$ is isomorphic to the Cartesian product $\Pi_2 \times \Pi_1$. The Cartesian product $\Pi = \Pi_1 \times \Pi_2$ is an abelian permutation set if and only if both Π_1 and Π_2 are abelian permutation sets.

The next two lemmas provide bounds on the size of difference covers for Cartesian products of permutation sets.

Lemma 8 *Let Π_1 be a permutation set on n_1 objects, and let Π_2 be a permutation set on n_2 objects. Then the Cartesian product $\Pi = \Pi_1 \times \Pi_2$, which is a permutation set on $n_1 \cdot n_2$ objects, has a difference cover of size $|\Pi_1| + |\Pi_2|$.*

Proof. Let Φ be the union of $\left\{(\pi_1^{-1}, I_{n_2}) : \pi_1 \in \Pi_1\right\}$ and $\{(I_{n_1}, \pi_2) : \pi_2 \in \Pi_2\}$. Each permutation $\pi = (\pi_1, \pi_2) \in \Pi$, can be represented as $(\pi_1, \pi_2) = (\pi_1^{-1}, I_{n_2})^{-1} \cdot (I_{n_1}, \pi_2)$, where both (π_1^{-1}, I_{n_2}) and (I_{n_1}, π_2) are in Φ. Thus Φ is a difference cover for Π, and the size of Φ is exactly $|\Pi_1| + |\Pi_2|$. ■

Lemma 9 *Let Π_1 be a permutation set on n_1 objects with a difference cover Φ_1, and let Π_2 be a permutation set on n_2 objects with a difference cover Φ_2. Then the Cartesian product $\Phi = \Phi_1 \times \Phi_2$ is a difference cover for $\Pi = \Pi_1 \times \Pi_2$.*

Proof. For each $\pi = (\pi_1, \pi_2) \in \Pi$, there exist $\phi_{i_1}, \phi_{j_1} \in \Phi_1$ such that $\pi_1 = \phi_{j_1}^{-1}\phi_{i_1}$, and there exist $\phi_{i_2}, \phi_{j_2} \in \Phi_2$ such that $\pi_2 = \phi_{j_2}^{-1}\phi_{i_2}$. We then have $(\pi_1, \pi_2) = (\phi_{j_1}^{-1}\phi_{i_1}, \phi_{j_2}^{-1}\phi_{i_2}) = (\phi_{j_1}, \phi_{j_2})^{-1}(\phi_{i_1}, \phi_{i_2})$, where both (ϕ_{i_1}, ϕ_{i_2}) and (ϕ_{j_1}, ϕ_{j_2}) are in $\Phi = \Phi_1 \times \Phi_2$, and hence Φ is a difference cover for Π. ■

To demonstrate both the use of difference covers and of Lemma 9, we present in Figure 2-2 a uniform permutation architecture due to Fiduccia [28] for realizing shifts in a two-dimensional array. The architecture uniformly realizes the permutation set $\Pi = \{\mathrm{I}, \mathrm{N}, \mathrm{E}, \mathrm{S}, \mathrm{W}, \mathrm{NE}, \mathrm{SE}, \mathrm{NW}, \mathrm{SW}\}$ of eight compass directions plus the identity I. We introduce

two permutation sets $\Pi_1 = \{I, N, S\}$, $\Pi_2 = \{I, E, W\}$, and corresponding difference covers $\Phi_1 = \{I, N\}$ and $\Phi_2 = \{I, E\}$. The Cartesian product $\Pi_1 \times \Pi_2$ is Π, and the set of permutations $\Phi = \Phi_1 \times \Phi_2 = \{I, E, NE, N\}$ is a difference cover for Π.

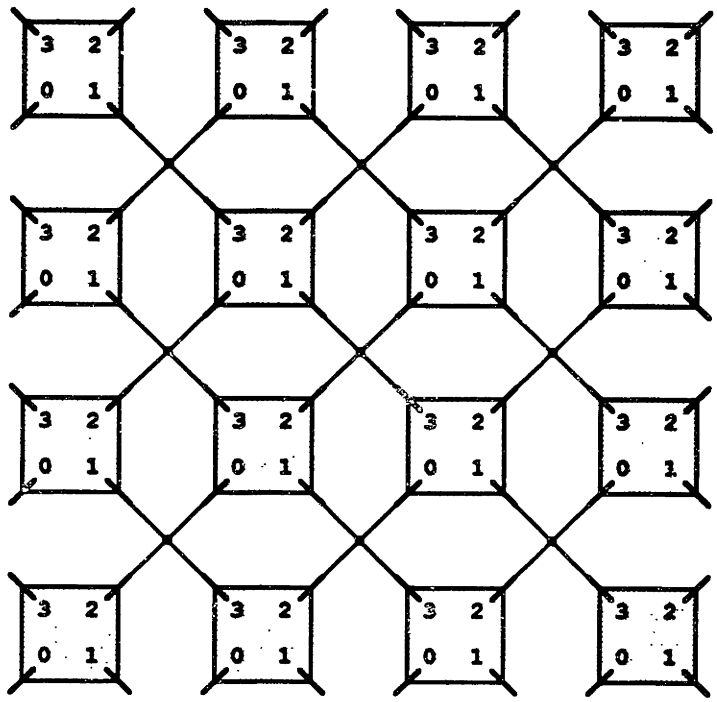

Figure 2-2: A uniform architecture due to Fiduccia [28] based on the difference cover $\{I, E, NE, N\}$ for the permutation set $\Pi = \{I, N, E, S, W, NE, SE, NW, SW\}$.

2.3.3 Substring covers: an alternative notation

We conclude this section by defining the notion of a substring cover for a permutation set Π, which is equivalent to the notion of a difference cover. (A similar notion for difference sets is well known in the literature [14, 66].)

Definition 7 An ordered list $\Sigma = \langle \sigma_0, \sigma_1, \ldots, \sigma_{k-1} \rangle$ of permutations is a *substring cover* for a permutation set Π if

1. $\sigma_0 \sigma_1 \cdots \sigma_{k-1} = I$, and

2. for all $\pi \in \Pi$, there exist $0 \leq i, j \leq k-1$ such that $\pi = \sigma_i \sigma_{i+1} \cdots \sigma_j$, where the arithmetic in the indices is performed modulo k.

The substring cover Σ is a list of permutations such that all the permutations in Π can be represented as a composition of a substring of permutations of Σ. The following two theorems show that the notions of a substring cover and a difference cover are equivalent.

Theorem 10 *Let Π be a permutation set on n elements, and let Σ be a k-element substring cover for Π. Then Π has a difference cover Φ with at most k elements.*

Proof. Given a k-element substring cover $\Sigma = \langle \sigma_0, \sigma_1, \ldots, \sigma_{k-1} \rangle$ for Π, a difference cover Φ with at most k elements can be constructed. For each $0 \leq i \leq k-1$ we define $\phi_i = \sigma_0 \sigma_1 \cdots \sigma_i$. If a permutation π can be represented as $\pi = \sigma_i \sigma_{i+1} \cdots \sigma_j$, then $\pi = \phi_{i-1}^{-1} \phi_j$. By construction, the difference cover Φ has at most k elements. ■

Theorem 11 *Let Π be a permutation set on n elements, and let Φ be a k-element difference cover for Π. Then Π has a substring cover Σ with k elements.*

Proof. Given a k-element difference cover $\Phi = \{\phi_0, \phi_1, \ldots, \phi_{k-1}\}$ for Π, we build a substring cover Σ for Π by defining $\sigma_i = \phi_{i-1}^{-1} \phi_i$ for all $0 \leq i \leq k-1$. The product $\sigma_0 \sigma_1 \cdots \sigma_{k-1}$ yields the identity permutation. For each $\pi \in \Pi$, if $\pi = \phi_i^{-1} \phi_j$, then $\pi = \sigma_{i+1} \sigma_{i+2} \cdots \sigma_j$. Therefore Σ is a substring cover for Π with k elements. ■

Referring back to the example of the eight compass directions, we present a substring cover for the permutation set $\Pi = \{\mathrm{I}, \mathrm{N}, \mathrm{E}, \mathrm{S}, \mathrm{W}, \mathrm{NE}, \mathrm{SE}, \mathrm{NW}, \mathrm{SW}\}$. The substring cover $\Sigma = \langle \mathrm{S}, \mathrm{E}, \mathrm{N}, \mathrm{W} \rangle$ is constructed from the difference cover $\Phi = \{\mathrm{I}, \mathrm{E}, \mathrm{NE}, \mathrm{N}\}$ that was used in the architecture of Figure 2-2. Each of the eight compass directions can be realized as a substring of the list $\Sigma = \langle \mathrm{S}, \mathrm{E}, \mathrm{N}, \mathrm{W} \rangle$.

As another example, consider the permutation set $\Pi = \{\mathrm{I}, \mathrm{N}, \mathrm{E}, \mathrm{S}, \mathrm{W}\}$ of the shifts in a 2-dimensional array corresponding to the four compass directions. This permutation set has a difference cover $\Phi = \{\mathrm{I}, \mathrm{SE}, \mathrm{S}\}$ and a corresponding substring cover $\Sigma = \langle \mathrm{N}, \mathrm{SE}, \mathrm{W} \rangle$. Consequently, there is a uniform architecture for realizing the four compass directions with three pins per chip, as has been observed by Feynman [36, pp. 437–438]. Figure 2-3 presents a uniform architecture based on the difference cover $\Phi = \{\mathrm{I}, \mathrm{SE}, \mathrm{S}\}$ for the permutation set $\Pi = \{\mathrm{I}, \mathrm{N}, \mathrm{E}, \mathrm{S}, \mathrm{W}\}$.

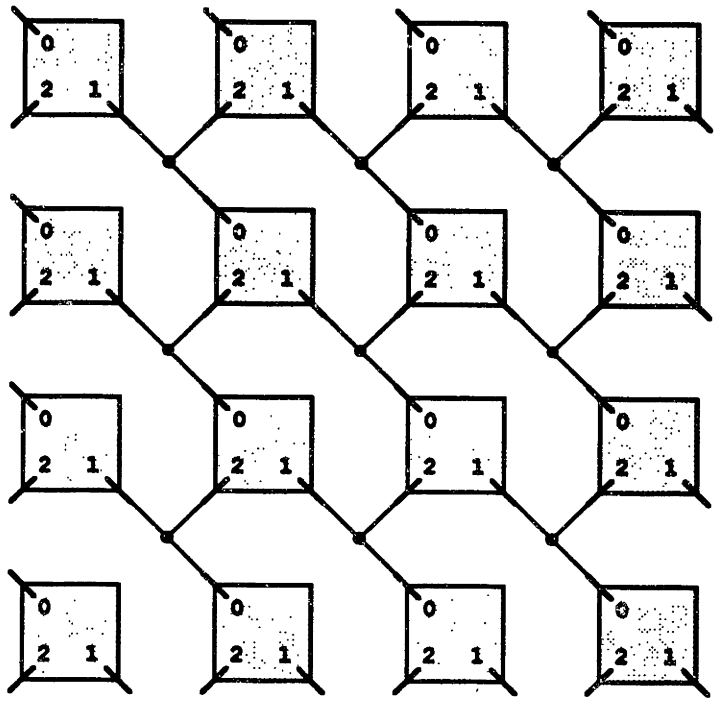

Figure 2-3: A uniform architecture due to Feynman [36] based on the difference cover $\{I, SE, S\}$ for the permutations set $\Pi = \{I, N, E, S, W\}$.

2.4 Cyclic shifters

This section describes uniform architectures for realizing cyclic shifts among n chips in one clock tick. We first present a difference cover of size $O(\sqrt{n})$ for the set of all n cyclic shifts on n elements, and we give an area-efficient layout for the corresponding permutation architecture suitable for implementation as a printed-circuit board. When n can be expressed as $n = q^2 + q + 1$, where q is a power of a prime, we improve the bound on the size of a difference cover for all cyclic shifts on n elements to the optimal value of $\lceil \sqrt{n} \rceil$. Finally, we prove that for any cyclic shifter that operates in one clock tick (even a nonuniform one), the average number of pins per chip is at least $\lceil \sqrt{n} \rceil$.

2.4.1 General difference covers for cyclic shifts

The first permutation architecture for cyclic shifters that we present is based on the construction in the following theorem.

Theorem 12 *The set of n cyclic shifts on n elements has a difference cover of size at most $2\lceil\sqrt{n}\rceil - 1$.*

Proof. Since the set of n cyclic shifts on n elements forms a group, and since this group is isomorphic to the group $\mathbf{Z}_n$, we shall construct a difference cover D for $\mathbf{Z}_n$. For convenience, let $m = \lceil\sqrt{n}\rceil$. Define two sets $A = \{0, 1, \ldots, m-1\}$ and $B = \{0, m, 2m, \ldots, (m-1)m\}$, and let the difference cover D be defined by $D = A \cup B$. Each element $s \in \mathbf{Z}_n$ can be realized as $s = b - a \pmod{n}$, where $a \in A$ and $b \in B$ by taking $a = m - (s \bmod m)$ and $b = \lceil s/m \rceil \cdot m$, as can be verified. The size of the difference cover D is $2m - 1 = 2\lceil\sqrt{n}\rceil - 1$, since the element 0 occurs in both A and B. ∎

The difference cover constructed in the proof of Theorem 12 corresponds to an architecture with a regular, area-efficient layout, as shown in Figure 2-4. The n chips of the architecture are laid out in an array consisting of $m = \sqrt{n}$ rows, each containing $\sqrt{n}$ chips. (For simplicity, we assume that n is a square.) Each chip has pins $0, 1, \ldots, m-1$ on the top side, and pins $m, m+1, \ldots, 2m-1$ on the left side. Each bus consists of one vertical segment and one or two horizontal segments. Each wiring channel consists of $m = \sqrt{n}$ tracks, where each track is used to lay out segments of busses. When n is not a square, a cyclic shifter on n chips can be laid out in a similar fashion, with each wiring channel having at most $2\lceil\sqrt{n}\rceil$ tracks. The side of the layout is therefore $O(n)$, since there are $\lceil\sqrt{n}\rceil$ chips and $\lceil\sqrt{n}\rceil$ wiring channels along the side. The area of the layout is $O(n^2)$, which is asymptotically optimal since any architecture that can realize any of the cyclic-shift permutations in one clock tick requires area $\Omega(n^2)$ [83, p. 56].

Remark. The bound of $2\lceil\sqrt{n}\rceil - 1$ pins per chip can be improved to $(\sqrt{2} + o(1))\sqrt{n}$, as was observed by Mills and Wiedemann [68]. See Section 2.7.4.

Occasionally, it is desirable to implement a subset of the cyclic shifts on n elements. The following corollary to Theorem 12 shows that when the shift amounts form an arithmetic sequence, a small difference cover exists.

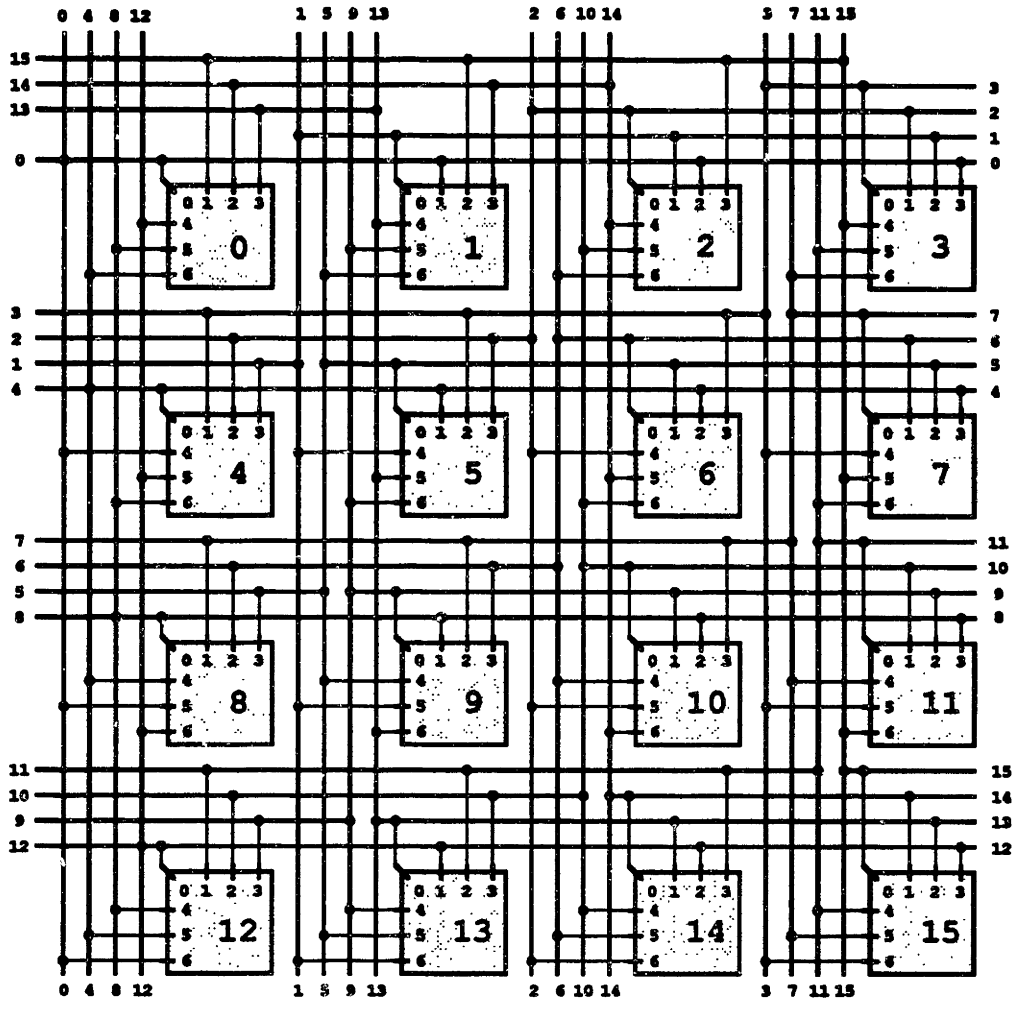

Figure 2-4: A layout for a cyclic shifter with $n = 16$ chips. Each chip and each bus has 7 pins. Each bus is constructed of one vertical segment and either one or two horizontal segments.

Corollary 13 *Let a, b, and p be integers modulo n. For each $r \in [p]$, define π_r to be the permutation on $[n]$ that maps each $c \in [n]$ to $c + a + rb \pmod n$. Then the permutation set $\{\pi_r : r \in [p]\}$ has a difference cover of size $2\lceil\sqrt{p}\,\rceil$.*

Proof. As in the proof of Theorem 12, we construct two sets A and B whose union is the desired difference cover. The sets are $A = \{0, b, 2b, \ldots, (m-1)b\}$ and $B = \{a, a + mb, a + 2mb, \ldots, a + (m-1)mb\}$, where $m = \lceil\sqrt{p}\,\rceil$. ∎

2.4.2 Optimal difference covers for cyclic shifts

Returning to the problem of implementing all n cyclic shifts on n elements, the following theorem demonstrates that for certain values of n, the optimal $\lceil\sqrt{n}\rceil$ bound can be obtained.

Theorem 14 *The set of n cyclic shifts on n elements has a difference cover of size $\lceil\sqrt{n}\rceil$ if $n = q^2 + q + 1$, where q is a power of a prime.*

Proof. As in the proof of Theorem 12, the problem is equivalent to that of constructing a difference cover D for $\mathbf{Z}_n$. When n is the size of a projective plane ($n = q^2 + q + 1$, where q is a power of a prime), this problem is equivalent to the problem of constructing a difference set. The difference set we give is due to Singer; a proof of its correctness is given in Hall [43, p. 129]. Let x be a primitive root of the Galois field $\mathrm{GF}(q^3)$, and let $F(y)$ be any irreducible cubic polynomial over the Galois field $\mathrm{GF}(q)$. We construct a difference cover D for $\mathbf{Z}_n$ from the set $[n]$ by choosing those $i \in [n]$ such that the power x^i can be written in the form $x^i = ax + b \pmod{F(x)}$ for some $a, b \in \mathrm{GF}(q)$. ∎

The construction of a uniform architecture based on a projective plane can be interpreted as follows. The n points of the projective plane correspond to the n chips, and the n lines of the projective plane correspond to the n busses. Each line contains $q + 1$ points, which means that each bus is connected to $q + 1$ chips. Each point is incident on $q + 1$ lines, which means that each chip is connected to $q + 1$ different busses through its $q + 1$ pins. For example, Figure 2-1 demonstrates a uniform architecture based on the projective plane of size 13.

Theorems similar to Theorem 12 (but without application to architecture) appear in the combinatorics literature: see, for example, [56]. Bus connection networks based on projective planes have also been studied by Bermond, Bond, and Scalé [11] and by Mickunas [64], who observed that projective planes can be used to construct hypergraphs of diameter one.

2.4.3 Lower bound for cyclic shifters

Uniform architectures for cyclic shifters based on projective planes achieve the minimal number of pins per chip among all uniform cyclic shifters. We now prove a lower bound of $\lceil\sqrt{n}\rceil$ on the average number of pins per chip for any permutation architecture that realizes all the cyclic shifts. This lower bound applies to all permutation architectures, including nonuniform ones,

and shows that uniform cyclic shifters based on projective planes are optimal among all cyclic shifters that operate in a single clock tick.

Theorem 15 *Let* $\mathcal{A} = \langle C, B, P, \text{CHIP}, \text{BUS}, \text{LABEL}\rangle$ *be a permutation architecture for the* n *cyclic shifts on* n *chips. Then the average number of pins per chip is at at least* $\lceil\sqrt{n}\rceil$.

Proof. The average number of pins per chip is $|P|/n$. We shall prove that $|P| \geq n\lceil\sqrt{n}\rceil$ which implies the theorem. We adopt the following conventions for notational convenience:

1. The set of busses is $B = \{b_0, b_1, \ldots, b_{m-1}\}$. We denote by k_i the number of pins connected to bus b_i, that is, $k_i = |\{x \in P : \text{BUS}(x) = b_i\}|$.

2. The busses that have at least $\lceil\sqrt{n}\rceil$ pins each are indexed first, that is, if there are r busses with at least $\lceil\sqrt{n}\rceil$ pins each, then $k_i \geq \lceil\sqrt{n}\rceil$ for $i = 0, \ldots, r-1$ and $k_i < \lceil\sqrt{n}\rceil$ for $i = r, \ldots, m-1$.

The thrust of the proof is to count the number of distinct data transfers when the architecture realizes each of the $n-1$ nontrivial shifts in turn. (The identity permutation is a trivial shift.) Each chip can be mapped to each other chip by one of the cyclic shifts, i.e., the cyclic shifts form a transitive group of permutations. Considering only the $n-1$ nontrivial shifts, there are exactly $n(n-1)$ distinct data transfers that must be implemented through interconnections in the architecture.

We compute an upper bound on the number of distinct data transfers that the busses can implement. Each of the first r busses $b_0, \ldots, b_{r-1}$ can be employed to realize at most one distinct data transfer in each of the $n-1$ nontrivial shifts. Thus, at most $r(n-1)$ distinct data transfers can be carried out by the first r busses. Any other bus b_i, where $r \leq i \leq m-1$, can realize at most $k_i(k_i-1)$ distinct nontrivial data transfers, since it has only k_i pins connected to it. Thus, the total number of distinct data transfers that the busses can realize is

$$r(n-1) + \sum_{i=r}^{m-1} k_i(k_i-1)\,,$$

which must be larger than $n(n-1)$ if all nontrivial shifts are to be realized. Hence, we have

$$\sum_{i=r}^{m-1} k_i(k_i-1) \geq (n-r)(n-1)\,.$$

We can use this inequality to bound the number of pins on all busses with fewer than $\lceil\sqrt{n}\rceil$ pins. We have $k_i - 1 \leq \lceil\sqrt{n}\rceil - 2$ for $i = r, \ldots, m-1$, and thus

$$\begin{aligned}\sum_{i=r}^{m-1} k_i &\geq \frac{1}{\lceil\sqrt{n}\rceil - 2} \sum_{i=r}^{m-1} k_i(k_i - 1) \\ &\geq \frac{(n-r)(n-1)}{\lceil\sqrt{n}\rceil - 2} \\ &\geq (n-r)\lceil\sqrt{n}\rceil \ .\end{aligned}$$

We now bound the total number of pins in the architecture from below. We have

$$\begin{aligned}|P| &= \sum_{i=0}^{m-1} k_i \\ &= \sum_{i=0}^{r-1} k_i + \sum_{i=r}^{m-1} k_i \\ &\geq r\lceil\sqrt{n}\rceil + (n-r)\lceil\sqrt{n}\rceil \\ &= n\lceil\sqrt{n}\rceil ,\end{aligned}$$

which proves the theorem. ■

2.5 Difference covers for groups

In this section we show that small difference covers for abelian and nonabelian permutation groups exist. Specifically, for any abelian permutation group Π with p elements, we apply the decomposition theorem for finite abelian groups and the results for cyclic shifters in Section 2.4, and we show the existence of a difference cover of size $O(\sqrt{p})$, which is optimal to within a constant factor. For a general permutation group Π with p elements, we give a greedy construction of a difference cover with $O(\sqrt{p \lg p})$ elements. Finkelstein, Kleitman, and Leighton [31] have recently improved our result for general groups to $O(\sqrt{p})$.

2.5.1 Abelian groups

We first show that if a permutation set forms an abelian group with p permutations, then a difference cover of size $O(\sqrt{p})$ can be constructed.

Theorem 16 *For any abelian group Π with p elements, there exists a difference cover Φ of size at most $3\sqrt{p}$.*

Proof. Assume without loss of generality that $p > 1$. By the decomposition theorem for finite abelian groups [58, p. 133], any abelian group Π is isomorphic to a cross product of cyclic groups

$$\Pi \approx \mathbf{Z}_{p_1} \times \mathbf{Z}_{p_2} \times \cdots \times \mathbf{Z}_{p_k},$$

where $p_1 p_2 \cdots p_k = p$, and each $p_j \geq 2$. Let i be the unique index such that $p_1 p_2 \cdots p_{i-1} \leq \sqrt{p}$ and $p_{i+1} p_{i+2} \cdots p_k < \sqrt{p}$, and let $m = \lceil \sqrt{p} / p_1 p_2 \cdots p_{i-1} \rceil$. Using the argument of Theorem 12, we first construct a difference cover for $\mathbf{Z}_{p_i}$ from the union of two sets A_i and B_i, where $|A_i| \leq m$ and $|B_i| \leq \lfloor p_i / m \rfloor$, such that each element of $\mathbf{Z}_{p_i}$ can be expressed in the form $b - a \pmod{p_i}$ or $a - b \pmod{p_i}$, where $a \in A_i$ and $b \in B_i$.

We now construct a difference cover for $\Pi \approx \mathbf{Z}_{p_1} \times \mathbf{Z}_{p_2} \times \cdots \times \mathbf{Z}_{p_k}$ from the union of two sets A and B, where

$$A \approx \mathbf{Z}_{p_1} \times \mathbf{Z}_{p_2} \times \cdots \times \mathbf{Z}_{p_{i-1}} \times A_i,$$

and

$$B \approx B_i \times \mathbf{Z}_{p_{i+1}} \times \mathbf{Z}_{p_{i+2}} \times \cdots \times \mathbf{Z}_{p_k}.$$

That $A \cup B$ is a difference cover for Π follows from essentially the same argument as is used in Lemma 9. The size of the difference cover $A \cup B$ is $|A| + |B|$. The size of A is

$$\begin{aligned} |A| &= p_1 p_2 \cdots p_{i-1} |A_i| \\ &\leq p_1 p_2 \cdots p_{i-1} m \\ &\leq p_1 p_2 \cdots p_{i-1} \lceil \sqrt{p} / p_1 p_2 \cdots p_{i-1} \rceil \\ &\leq \sqrt{p} + p_1 p_2 \cdots p_{i-1} \\ &\leq 2\sqrt{p}. \end{aligned}$$

Similarly, the size of B is

$$\begin{aligned} |B| &= |B_i| \, p_{i+1} p_{i+2} \cdots p_k \\ &\leq \lfloor p_i / m \rfloor \, p_{i+1} p_{i+2} \cdots p_k \\ &\leq (p_i / \lceil \sqrt{p} / p_1 p_2 \cdots p_{i-1} \rceil) \, p_{i+1} p_{i+2} \cdots p_k \\ &\leq (p_1 p_2 \cdots p_i / \sqrt{p}) \, p_{i+1} p_{i+2} \cdots p_k \\ &= \sqrt{p}. \end{aligned}$$

Consequently, the size of the difference cover for Π is at most $3\sqrt{p}$. ∎

2.5.2 General groups

The next theorem gives a method for constructing small difference covers for general groups of permutations.

Theorem 17 *Let Π be an arbitrary group with p elements. Then Π has a difference cover Φ of size at most $\sqrt{2p\ln p} + 1$.*

Proof. We construct a difference cover incrementally starting with a partial difference cover $\Phi_1 = \{I\}$. At each step of the construction, we select an element $\phi_{i+1} \in \Pi$ such that $\left|\Phi_i^{-1}(\Phi_i \cup \{\phi_{i+1}\})\right|$ maximizes $\left|\Phi_i^{-1}(\Phi_i \cup \{\pi\})\right|$ over all $\pi \in \Pi$. We then define the new partial difference cover as $\Phi_{i+1} = \Phi_i \cup \{\phi_{i+1}\}$.

The analysis of this construction is in three parts. We first determine a lower bound on the number of elements of Π that are not covered by the partial difference cover Φ_i but are covered by Φ_{i+1}. We then develop a recurrence to upper bound the number of elements of the group Π that are not covered at the ith step. Finally, we solve the recurrence to determine that the number k of iterations needed to cover all elements in Π is at most $\sqrt{2p\ln p} + 1$.

We first determine how many new elements of Π are covered when Φ_i is augmented with ϕ_{i+1} to produce Φ_{i+1}, for $i \geq 1$. Let the set Δ_i be the set of elements that are not covered by the partial difference cover Φ_i, which can be defined as $\Delta_i = \Pi - \Phi_i^{-1}\Phi_i$. Consider triples of the form $\langle \phi, \delta, \pi \rangle$ such that $\phi \in \Phi_i$, $\delta \in \Delta_i$, $\pi \in \Pi$, and $\phi\delta = \pi$. Observe that for any fixed $\pi \in \Pi$ and $\delta \in \Delta_i$, there is at most one triple of the form $\langle \phi, \delta, \pi \rangle$ in the set of triples, namely $\langle \pi\delta^{-1}, \delta, \pi \rangle$ when $\pi\delta^{-1} \in \Phi_i$. For a fixed π, the number of triples $\langle \phi, \delta, \pi \rangle$ in the set of triples is a lower bound on the number of elements covered by $\Phi_i \cup \{\pi\}$ but not by Φ_i, since we have $\delta = \phi^{-1}\pi$ and $\delta \in \Delta_i = \Pi - \Phi_i^{-1}\Phi_i$. For each $\phi \in \Phi_i$ and $\delta \in \Delta_i$, there is exactly one triple in the set of triples, and thus there are exactly $|\Phi_i| \cdot |\Delta_i|$ triples. Since there are at most $|\Pi|$ distinct permutations appearing as the third coordinate of a triple, the permutation ϕ_{i+1} that appears most often must appear at least $|\Phi_i| \cdot |\Delta_i| / |\Pi|$ times, and hence at least this many elements are covered by Φ_{i+1} that are not covered by Φ_i.

We can now bound the number of elements not covered by Φ_{i+1} in terms of the number of elements not covered by Φ_i by

$$|\Delta_{i+1}| \leq |\Delta_i| - \frac{|\Phi_i| \cdot |\Delta_i|}{|\Pi|}$$

$$\begin{aligned} &= |\Delta_i|\left(1-\frac{i}{p}\right) \\ &= |\Delta_1|\prod_{j=1}^{i}\left(1-\frac{j}{p}\right) \\ &< p\prod_{j=1}^{i}\left(1-\frac{j}{p}\right). \end{aligned}$$

When we obtain $|\Delta_k| < 1$ for some k, the partial difference cover Φ_k is a difference cover for Π because Δ_k is empty. Thus, Φ_k is a difference cover when

$$p\prod_{j=1}^{k-1}\left(1-\frac{j}{p}\right) \le 1,$$

or equivalently, when

$$\ln p + \sum_{j=1}^{k-1}\ln\left(1-\frac{j}{p}\right) \le 0.$$

Using the inequality $\ln(1+x) \le x$, we have

$$\begin{aligned} \ln p + \sum_{j=1}^{k-1}\ln\left(1-\frac{j}{p}\right) &\le \ln p - \sum_{j=1}^{k-1}\frac{j}{p} \\ &= \ln p - \frac{1}{p}\sum_{j=1}^{k-1} j \\ &\le \ln p - \frac{(k-1)^2}{2p} \\ &\le 0. \end{aligned}$$

Thus, Φ_k is a difference cover when $k \ge \sqrt{2p\ln p} + 1$. ∎

This proof of Theorem 17 provides a construction which can be implemented as an deterministic, polynomial-time algorithm with $O(p^2 \lg p)$ algebraic steps. We could also have proved the theorem by relying on the result of Babai and Erdös [4] that any group has a small set of generators, but this method would have produced only an existential (nonconstructive) result.

Finkelstein, Kleitman, and Leighton [31] have recently improved our result for general groups to $O(\sqrt{p})$. Their proof uses a folk theorem [25] that every simple group of nonprime order p has a subgroup of size at least $\sqrt{p}$. The folk theorem is proved by checking each type of group in the classification theorem [39, pp. 135–136].

2.6 Multiple clock ticks

In this section, we discuss uniform permutation architectures that realize permutations in several clock ticks. By using more than one clock tick, further savings in the number of pins per chip can be obtained. We first generalize the notion of a difference cover to handle multiple clock ticks. We then describe a cyclic shifter on n chips with only $O(n^{1/2t})$ pins per chip that operates in t ticks.

2.6.1 The notion of a t-difference cover

We first generalize the notion of a difference cover to handle realization of permutations in $t \geq 1$ clock ticks.

Definition 8 A *t-difference cover* for a permutation set Π is a set Φ of permutations such that $(\Phi^{-1}\Phi)^t \supseteq \Pi$.

Using a t-difference cover Φ for the permutation set Π, any permutation $\pi \in \Pi$ can be expressed as the composition of t differences of permutations from Φ. The next lemma relates t-difference covers to permutation architectures that uniformly realize permutations in t clock ticks, for general values of t.

Lemma 18 *Let Φ be a t-difference cover with k elements for a permutation set Π. Then there is a permutation architecture with k pins per chip that uniformly realizes Π in t clock ticks.*

Proof. We define the permutation set $\Sigma = \Phi^{-1}\Phi$. Let $\mathcal{A} = \langle C, B, P, \text{CHIP}, \text{BUS}, \text{LABEL}\rangle$ be the permutation architecture, based on the difference cover Φ, that uniformly realizes Σ. Hence, the permutation architecture $\mathcal{A}$ can uniformly realize any $\sigma \in \Sigma$ in one clock tick. Each permutation $\pi \in \Pi$ can be expressed as $\pi = \sigma_{t-1}\sigma_{t-2}\cdots\sigma_0$, where $\sigma_i \in \Sigma$ for $0 \leq i \leq t-1$, since we have $\Sigma^t = (\Phi^{-1}\Phi)^t \supseteq \Pi$. In order to realize π in t clock ticks, the permutation architecture $\mathcal{A}$ uniformly realizes σ_i in clock tick i for $0 \leq i \leq t-1$. ∎

2.6.2 Constructing t-difference covers for cyclic shifters

Lemma 18 claims that the problem of uniformly realizing a permutation set Π in t clock ticks can be reduced to finding a permutation set Σ such that $\Sigma^t \supseteq \Pi$, and then finding a difference

cover for Σ. The great advantage of using more than one clock tick is in the further savings in the number of pins per chip. The following theorem, for example, describes a construction of a t-difference cover of size $O(n^{1/2t})$ for the set of cyclic shifts on n objects. This result can be used to build a uniform architecture on n chips with only $O(n^{1/2t})$ pins per chip that can realize any cyclic shift on the n chips in t clock ticks.

Theorem 19 *For any $n \geq 1$ and $t \geq 1$, the permutation set of all the n cyclic shifts on n objects has a t-difference cover of size $O(n^{1/2t})$.*

Proof. For the purpose of the proof, we denote the permutation set of all the n cyclic shifts on n objects by Π_n. (We remind that $\Pi_n \approx \mathbf{Z}_n$.) We first treat the case for those n such that there exists an integer m satisfying $n^{1/t} \leq m \leq 4n^{1/t}$ and $\gcd(m, n) = 1$. We then use this case to extend the proof to all values of n.

Since $\gcd(m, n) = 1$, there exists an $m^{-1} \in \mathbf{Z}_n$ such that $m \cdot m^{-1} = 1 \pmod n$. For each $r \in [m]$, define the permutation $\sigma_r : [n] \to [n]$ as $\sigma_r(c) = m^{-1}(c + r) \pmod n$, and define the permutation $\sigma'_r : [n] \to [n]$ as $\sigma'_r(c) = m^{t-1}(c + r) \pmod n$. Next define the permutation set $\Sigma = \{\sigma_r\} \cup \{\sigma'_r\}$. The set $\{\sigma_r\}$ is an arithmetic sequence of cyclic shifts on n elements (as in Corollary 13) followed by the fixed permutation corresponding to multiplication by m^{-1}, and thus $\{\sigma_r\}$ has a difference cover of size $O(\sqrt{m})$. Similarly, the set $\{\sigma'_r\}$ has a difference cover of size $O(\sqrt{m})$. Combining the two difference covers for $\{\sigma_r\}$ and $\{\sigma'_r\}$, we get a difference cover Φ of size $O(\sqrt{m}) = O(n^{1/2t})$ for Σ.

We now show the inclusion $\Sigma^t \supseteq \Pi_n$. Let $\pi \in \Pi_n$ be a permutation of a cyclic shift by s. We express the shift amount $s \in [n]$ as $s = s_0 + s_1 m + \cdots + s_{t-1} m^{t-1}$, where $s_i \in [m]$ for $0 \leq i \leq t - 1$. The permutation π can be described as

$$\begin{aligned}
\pi(c) &= c + s \pmod n \\
&= c + s_0 + s_1 m + \cdots + s_{t-1} m^{t-1} \pmod n \\
&= m^{t-1}\left(s_{t-1} + m^{-1}\left(s_{t-2} + \cdots + m^{-1}(s_0 + c)\right)\right) \pmod n \\
&= \sigma'_{s_{t-1}} \sigma_{s_{t-2}} \cdots \sigma_{s_0}(c),
\end{aligned}$$

which proves that $\pi \in \Sigma^t$. Hence, we get the inclusion $\Sigma^t \supseteq \Pi_n$, which together with the fact that there is a difference cover Φ of size $O(n^{1/2t})$ for Σ, proves the theorem for the case when there exists an integer m satisfying $n^{1/t} \leq m \leq 4n^{1/t}$ and $\gcd(m, n) = 1$.

Such an m need not exist for every n and every t, however. We can overcome this difficulty by factoring $n = n_1 n_2$ such that n_1 consists of no even-indexed primes (3, 7, 13, ...) and n_2 consists of no odd-indexed primes (2, 5, 11, ...). Since we have $\gcd(n_1, n_2) = 1$, we can use the Chinese remainders theorem to express $\mathbf{Z}_n$ as a Cartesian product $\mathbf{Z}_n \approx \mathbf{Z}_{n_1} \times \mathbf{Z}_{n_2}$. We let m_1 be the first even-indexed prime at least as large as $n_1^{1/t}$, and let m_2 be the first odd-indexed prime at least as large as $n_2^{1/t}$. Bertrand's postulate [44, p. 343] guarantees that for every x, there is a prime between x and $2x$, which means $m_j \in [n_j^{1/t}, 4n_j^{1/t}]$ for $j = 1, 2$. (Tighter bounds are possible.)

We can now use the previous construction to construct a t-difference cover Φ_1 of size $O(n_1^{1/2t})$ for $\mathbf{Z}_{n_1}$, which is isomorphic to Π_{n_1}, and a t-difference cover Φ_2 of size $O(n_2^{1/2t})$ for $\mathbf{Z}_{n_2}$, which is isomorphic to Π_{n_2}. Using the same technique as in the proof of Lemma 9, we can construct a t-difference cover of size $O(n_1^{1/2t}) \cdot O(n_2^{1/2t}) = O(n^{1/2t})$ for $\mathbf{Z}_{n_1} \times \mathbf{Z}_{n_2} \approx \mathbf{Z}_n \approx \Pi_n$. ■

One can rather straightforwardly use Corollary 13 to obtain a t-difference cover of size $O(tn^{1/2t})$. Based on the representation of the shift amount $s = s_0 + s_1 m + \cdots + s_{t-1} m^{t-1}$, one can come with t separate difference covers, each of size $O(n^{1/2t})$, for the t separate sequences of arithmetic shifts by $\{sm^i : s \in [m]\}$ for $0 \leq i \leq t-1$. Theorem 19 avoids the extra factor of t by constructing only one such difference cover and using its elements for each one of the t differences.

2.7 Applications and extensions

This section contains some additional results on permutation architectures and difference covers. We describe efficient uniform architectures that can realize the permutations implemented by various popular interconnection networks, including multidimensional meshes, hypercubes, and shuffle-exchange networks. We extend the lower bound technique of Section 2.4.3 to general permutation sets. We examine nonuniform permutation architectures, and adapt some combinatorial results in the literature to apply to permutation architectures. Finally, we describe directions for further research and some related work brought on by an earlier version [48] of this research.

2.7.1 More networks

By using busses, many popular interconnection networks can be realized with fewer pins than conventionally proposed. Here, we mention a few.

The permutation architectures for realizing compass shifts on two-dimensional arrays can be extended in a natural fashion to d-dimensional arrays. For the d-dimensional analogue of the shifts $\{\mathrm{I}, \mathrm{N}, \mathrm{E}, \mathrm{S}, \mathrm{W}\}$, there is a uniform architecture that uses only $d+1$ pins per chip to implement the $2d+1$ permutations. For the d-dimensional analogue of the shifts $\{\mathrm{I}, \mathrm{N}, \mathrm{E}, \mathrm{S}, \mathrm{W}, \mathrm{NE}, \mathrm{SE}, \mathrm{NW}, \mathrm{SW}\}$, there is a uniform architecture with only 2^d pins per chip that implements all 3^d shifts. (These results were independently obtained by Fiduccia [29, 30].)

A Boolean hypercube of dimension d is a degenerate case of a d-dimensional array. Only $d+1$ pins per chip are required by a permutation architecture that uses busses, whereas $2d$ pins per chip are needed if point-to-point wires are used. (To realize a swap of information across a dimension in one clock tick, each chip requires two pins for that dimension: one to read and one to write.) It is interesting to mention that in the case of the d-dimensional hypercube, the permutation set consists of d permutations of swapping data across each of the d dimensions. For this case, Fiduccia [30] shows that $d+1$ pins per chip is the least possible.

A permutation architecture that implements the permutations Shuffle, Inverse Shuffle, and Exchange can be constructed with three pins per chip instead of the usual four. This can be done by taking the set of three permutation: Identity, Shuffle, and Exchange, which forms a difference cover for the desired permutation set. Furthermore, we can also implement the Shuffle-Exchange and Inverse Shuffle-Exchange permutations in one clock tick as well.

2.7.2 Average number of pins per chip

Theorem 15 presents a lower bound on the average number of pins per chip in any cyclic shifter that operates in one clock tick. The following theorem is a natural extension of Theorem 15 for a general set of permutations.

Theorem 20 *Let Π be a permutation set on n objects with p permutations and with total of T nontrivial data transfers, and let $\mathcal{A} = \langle C, B, P, \textsc{chip}, \textsc{bus}, \textsc{label}\rangle$ be any permutation architecture for realizing Π. Then the average number of pins per chip is at least $T/n\sqrt{p}$.*

Proof. As in the proof of Theorem 15, we prove that $|P| \geq T/\sqrt{p}$ which implies the theorem. We make similar notational conventions:

1. The set of busses is $B = \{b_0, b_1, \ldots, b_{m-1}\}$. We denote by k_i the number of pins connected to bus b_i.

2. The r busses that have at least $\sqrt{p}$ pins each are indexed first, that is $k_i \geq \sqrt{p}$ for $i = 0, \ldots, r-1$ and $k_i < \sqrt{p}$ for $i = r, \ldots, m-1$.

We count the number of distinct data transfers that can be accomplished by each bus. Each of the first r busses can be employed to realize at most p out of the T nontrivial data transfers, since it can be used at most once for each of the p permutation. Any other bus b_i, where $r \leq i \leq m-1$, can realize at most $k_i(k_i - 1)$ out of the T nontrivial data transfers, since it has only k_i pins connected to it. We need to have $\sum_{i=r}^{m-1} k_i(k_i - 1) \geq T - rp$, which implies

$$\begin{aligned} \sum_{i=r}^{m-1} k_i &\geq \frac{T - rp}{\sqrt{p}} \\ &= \frac{T}{\sqrt{p}} - r\sqrt{p}. \end{aligned}$$

The number of pins in the architecture can now be bounded as follows:

$$\begin{aligned} |P| &= \sum_{i=0}^{m-1} k_i \\ &= \sum_{i=0}^{r-1} k_i + \sum_{i=r}^{m-1} k_i \\ &\geq r\sqrt{p} + \left(\frac{T}{\sqrt{p}} - r\sqrt{p}\right) \\ &= \frac{T}{\sqrt{p}}. \end{aligned}$$

■

Theorem 20 demonstrates that uniform architectures can achieve the optimal number (to within a constant factor) of pins per chip for certain classes of permutation sets. When there are relatively few permutations that are responsible for many nontrivial data transfers, the average number of pins per chip is high. The set of cyclic shifts is an example of this kind of permutation set.

2.7.3 Nonuniform architectures

When the uniformity condition on permutation architectures is dropped, one can do much better in terms of the number of pins per chip. The complexity of control may increase substantially, however, due to the irregular communication patterns and the number of possible permutations realizable for some of the architectures. Nevertheless, from a mathematical point of view, nonuniform architectures are quite interesting.

In fact, nonuniform architectures have been studied quite extensively in the mathematics literature in the guise of partitioning problems. For the problem of realizing all $n!$ permutations on n chips, a result due to de Bruijn, Erdös, and Spencer [84, pp. 106-108] implies that $O(\sqrt{n \lg n})$ pins per chip suffice. The nonuniform architecture that achieves this bound is constructed probabilistically, however. It is an open problem to obtain this bound deterministically. The best deterministic construction to date is due to Feldman, Friedman, and Pippenger [26] and uses $O(n^{2/3})$ pins per chip.

2.7.4 Further research

We list a few of the problems that have been left open by our research. We also describe briefly some further work brought on by an earlier version [48] of this research.

In Section 2.4 we described a difference cover of size $2\lceil\sqrt{n}\rceil - 1$ for the cyclic group $\mathbf{Z}_n$, and proved that when n is the order of a projective plane, there is a difference cover of size $\lceil\sqrt{n}\rceil$. It seems reasonable that any cyclic group $\mathbf{Z}_n$ might actually have a difference cover of size $\sqrt{n} + o(\sqrt{n})$, but we have been unable to prove or disprove this conjecture. Mills and Wiedemann [67] have computed a table of minimal difference covers for all the cyclic groups of cardinality up to 110. For any value of n up to 110, the difference cover they find has at most $\lceil\sqrt{n}\rceil + 2$ elements. In [68], they provide a "folk theorem" that establishes a stronger upper bound for the general case than $2\lceil\sqrt{n}\rceil - 1$.

Theorem 21 *The set of n cyclic shifts on n elements has a difference cover of size $(\sqrt{2} + o(1))\sqrt{n}$.*

Sketch of proof. [68] Let q be the smallest prime such that $l = q^2 + q + 1 \geq n/2$. We have $q = (1 + o(1))\sqrt{n/2}$, since for large x, there exists a prime between x and $x + o(x)$. Let

$\{d_0, d_1, \ldots, d_q\}$ be a difference cover for $\mathbf{Z}_l$ chosen as in Theorem 14. It can be verified that the set $\{d_0, d_1, \ldots, d_q\} \cup \{d_0 + l, d_1 + l, \ldots, d_q + l\}$ forms a difference cover for $\mathbf{Z}_n$. ■

Another interesting problem related to cyclic shifters involves finding an area-efficient VLSI layout of the cyclic shifter based on projective planes. In section 2.4 we presented an area-efficient layout using a difference cover whose size is twice the optimal size. Is there a good layout for the pin-optimal design?

To implement cyclic shifters that operate in t clock ticks, we showed how to construct a t-difference cover for $\mathbf{Z}_n$ of size $O(n^{1/2t})$. A simpler construction achieves the bound $O(tn^{1/2t})$. Theorem 15 gives a lower bound of $\lceil \sqrt{n} \rceil$ on the average number of pins per chip for a cyclic shifter that operates in one clock tick. It may be possible to prove a lower bound of $\Omega(n^{1/2t})$ on the average number of pins per chip when an architecture operates in t clock ticks, but we were unable to extend the argument. We were also unable to extend either of these constructions to give good t-difference covers for groups, either general or abelian. It would be interesting to know whether a general (or an abelian) group of permutations with p permutations has a t-difference cover of size $O(tp^{1/2t})$, for any $t \geq 1$.

We have concentrated primarily on permutation sets that have good structure, specifically group properties. In general, when the permutation set has no known structure, the best possible upper bound is given by Theorem 7 of Section 2.3.2. It would be interesting to identify other structural properties of permutation sets besides group properties that allow small difference covers to exist.

Chapter 3

Priority Arbitration with Busses

This chapter explores how busses can be used to efficiently implement arbitration mechanisms. We investigate priority arbitration schemes that use busses to arbitrate among n modules in a digital system. We focus on distributed mechanisms that employ m arbitration busses, for $\lg n \leq m \leq n$, and use asynchronous combinational arbitration logic. A widely used distributed asynchronous mechanism is the *binary arbitration* scheme, which with $m = \lg n$ busses arbitrates in $t = \lg n$ units of time. We present a new asynchronous scheme — *binomial arbitration* — that by using $m = \lg n + 1$ busses reduces the arbitration time to $t = \frac{1}{2} \lg n$. Extending this result, we present the *generalized binomial arbitration* scheme that achieves a bus-time tradeoff of the form $m = \Theta(tn^{1/t})$ between the number of arbitration busses m, and the arbitration time t (in units of bus-settling delay), for values of $\lg n \leq m \leq n$ and $1 \leq t \leq \lg n$. Our schemes are based on a novel analysis of *data-dependent delays* and generalize the two known schemes: *linear arbitration*, which with $m = n$ busses achieves $t = 1$ time, and *binary arbitration*, which with $m = \lg n$ busses achieves $t = \lg n$ time. Most importantly, our schemes can be adopted with no changes to existing hardware and protocols; they merely involve selecting a *good* set of priority arbitration codewords. We also investigate the capabilities of general asynchronous priority arbitration schemes that employ busses and present some lower bound arguments that demonstrate the efficiency of our schemes.

This chapter describes research that appeared partially in [50] and [51].

3.1 Introduction

In many electronic systems there are situations where several modules wish to use a common resource simultaneously. Examples include microprocessor systems where a decision is required concerning which of several interrupts to service first, multiprocessor environments where several processors wish to use some device concurrently, and data communication networks with shared media. To resolve conflicts, an arbitration mechanism is required that grants the resource to one module at a time.

Numerous arbitration mechanisms have been developed, including daisy chains, priority circuits, polling, token passing, and carrier sense protocols, to name a few (see [12, 16, 22, 40, 57, 61, 78, 82]). In this chapter we focus on distributed *priority arbitration* mechanisms, where contention is resolved using predetermined module priorities and arbitration processes are carried out in a distributed manner by participating system modules. In many modern systems, and especially in multiprocessor environments and data communication networks, distributed priority arbitration is the preferred mechanism.

Many distributed arbitration mechanisms employ a collection of *arbitration busses* to implement priority arbitration. To this end, each module is assigned a unique *arbitration priority,* which is an encoding of its name. An arbitration protocol determines the logic values that a contending module applies to the busses, based on the module's arbitration priority and on logic values on the busses. After some delay, the settled logic values on the busses uniquely identify the contending module with the highest priority. In particular, the *asynchronous binary arbitration* scheme, developed by Taub [79], gained popularity and is used in many modern bus systems, such as Futurebus [17, 81], M3-bus [21], S-100 bus [35, 80], Multibus-II [40], Fastbus [41], and Nubus [89]. Other priority arbitration mechanisms that employ busses are described in [12, 16, 22, 24, 47, 57, 61, 78, 82].

The asynchronous binary arbitration scheme arbitrates among n modules in $t = \lg n$ units of time, using $m = \lg n$ wired-OR (open-collector) arbitration busses.[1] The technology of open-collector busses is such that the default logic value on a bus is 0, unless at least one module applies a 1 to it, in which case it becomes a 1. Open-collector busses, thus, OR together the logic

[1] Throughout this chapter we count only arbitration busses that are used for encoding the priorities. Several additional control busses are used by all schemes and are therefore not counted.

values applied to them, with some time delay called *bus-settling delay.* In asynchronous binary arbitration, each module is assigned a unique ($\lg n$)-bit arbitration priority. When arbitration begins, competing modules apply their arbitration priorities to the $m = \lg n$ busses, each bit on a separate bus; the result being the bitwise OR of their arbitration priorities. As arbitration progresses, each competing module monitors the busses and disables its drivers according to the following rule: if the module is applying a 0 (that is, not applying a 1) to a particular bus but detects that the bus is carrying a 1 (applied by some other module), it ceases to apply all its bits of lower significance. Disabled bits are re-enabled should the condition cease to hold. The effect of this rule is that the arbitration proceeds in at most $\lg n$ stages from the most significant bit to the least significant bit. Each stage consists of resolving another bit of the highest competing binary priority, which leads to a worst-case arbitration time of $t = \lg n$ (in units of bus-settling delay).

For example, consider a system of $n = 16$ modules that uses $m = \lg 16 = 4$ arbitration busses, with the 16 arbitration priorities consisting of all the 4-bit codewords {0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111}. Figure 3-1 outlines an asynchronous binary arbitration process among four such modules c_2, c_5, c_9, and c_{10}, with corresponding arbitration priorities 0010, 0101, 1001, and 1010. The arbitration process begins by the competing modules applying their arbitration priorities to the busses. The open-collector busses, therefore, compute a bitwise-OR of the four arbitration priorities. After one unit of bus-settling delay (stage 1), bus b_3 settles to the logic value 1, where it will remain for the duration of the arbitration. By the above rule, each of modules c_2 and c_5 disables its last three bits because they each apply a logic 0 to bus b_3 that now carries a logic 1. In the meantime, however, each of modules c_9 and c_{10} disables its last two bits, because of the logic 1 they detect on bus b_2. At the end of stage 2, therefore, bus b_2 settles to the logic value 0, where it will remain for the rest of the process. As a result, modules c_9 and c_{10} now re-enable their two low order bits (stage 3), because the conflict they previously detected on bus b_2 had disappeared, which results in bus b_1 settling to a logic 1 at the end of stage 3. Finally, in stage 4, module c_9 ceases to apply its last bit, because of the logic value 1 it now detects on bus b_1, which results in bus b_0 settling to a logic 0 at the end of stage 4. This arbitration process required $t = \lg 16 = 4$ stages to complete.

	Stage 1					Stage 2					Stage 3					Stage 4				
	c_2	c_5	c_9	c_{10}	OR	c_2	c_5	c_9	c_{10}	OR	c_2	c_5	c_9	c_{10}	OR	c_2	c_5	c_9	c_{10}	OR
Bus b_3	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1
Bus b_2	0	1	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
Bus b_1	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	1	0	0	1	1
Bus b_0	0	1	1	0	1	0	1	1	0	0	0	1	1	0	1	0	1	1	0	0

Figure 3-1: Asynchronous binary arbitration process with 4 busses. The competing modules are c_2, c_5, c_9, and c_{10}, with corresponding arbitration priorities 0010, 0101, 1001, and 1010. Bits in shaded regions are not applied to the busses. The arbitration process takes 4 stages.

In this chapter we show that the asynchronous binary arbitration scheme can in fact be improved. We introduce the new *asynchronous binomial arbitration* scheme, that uses one more arbitration bus in addition to the $\lg n$ busses of binary arbitration, but, most surprisingly, reduces the arbitration time to $\frac{1}{2}\lg n$. In asynchronous binomial arbitration, we use $(\lg n + 1)$-bit codewords as arbitration priorities and follow the same arbitration protocol of asynchronous binary arbitration. Our binomial arbitration scheme guarantees fast arbitration by employing certain codewords that exhibit small *data-dependent delays* during arbitration processes. For example, by using the following set of 5-bit codewords {00000, 00001, 00010, 00011, 00100, 00110, 00111, 01000, 01100, 01110, 01111, 10000, 11000, 11100, 11110, 11111} as arbitration priorities, we can arbitrate among 16 modules using 5 busses in at most 2 stages. Figure 3-2 outlines an asynchronous binomial arbitration process among four such modules c_1, c_6, c_{11}, and c_{12}, with corresponding arbitration priorities 00001, 00111, 10000, and 11000 from the above set of arbitration priorities, that completes in 2 stages. It turns out that for any subset of the above 16 codewords, the corresponding arbitration process never takes more than 2 stages. In Section 3.3, we show how to design a good set of codewords for general values of n by using *binomial codes* as arbitration priorities.

The remainder of this chapter explores priority arbitration schemes that employ busses to arbitrate among n modules. In Section 3.2 we discuss distributed priority arbitration and

	Stage 1					Stage 2				
	c_1	c_6	c_{11}	c_{12}	OR	c_1	c_6	c_{11}	c_{12}	OR
Bus b_4	0	0	1	1	1	0	0	1	1	1
Bus b_3	0	0	0	1	1	0	0	0	1	1
Bus b_2	0	1	0	0	1	0	1	0	0	0
Bus b_1	0	1	0	0	1	0	1	0	0	0
Bus b_0	1	1	0	0	1	1	1	0	0	0

Figure 3-2: Asynchronous binomial arbitration process with 5 busses. The competing modules are c_1, c_6, c_{11}, and c_{12}, with corresponding arbitration priorities 00001, 00111, 10000, and 11000. Bits in shaded regions are not applied to the busses. The arbitration process takes 2 stages.

formally define the asynchronous model of priority arbitration with busses. Section 3.3 describes the two known asynchronous schemes: *linear arbitration* and *binary arbitration*, and presents our new asynchronous *binomial arbitration* scheme, which with $m = \lg n + 1$ busses arbitrates in $t = \frac{1}{2}\lg n$ units of time. In Section 3.4 we extend binomial arbitration and present the *generalized binomial arbitration* scheme that achieves a spectrum of bus-time tradeoff of the form $m = \Theta(tn^{1/t})$, between the number of arbitration busses m and the arbitration time t, for values of $1 \leq t \leq \lg n$ and $\lg n \leq m \leq n$. The established bus-time tradeoff is of great practical interest, enabling system designers to achieve a desirable balance between amount of hardware and speed. In Section 3.5 we investigate general properties of asynchronous priority arbitration schemes that employ busses and present some lower bound arguments that demonstrate the efficiency of our schemes. Several extensions and discussion of the results of this chapter are presented in Section 3.6, as well as directions for further research.

3.2 Asynchronous priority arbitration with busses

In this section we discuss priority arbitration and formally define the asynchronous model of priority arbitration with busses. The definitions in this section model typical implementations of asynchronous priority arbitration mechanisms that employ busses.

Arbitration is the process of selecting one module from a set of contending modules. In asynchronous priority arbitration with busses, each module is assigned a unique arbitration priority — an encoding of its name — which is used in determining logic values to apply to the busses during arbitration. An arbitration protocol determines the logic values that a competing module applies to the busses, based on the module's arbitration priority and potentially also on logic values on other busses. The beginning of an arbitration process is generally indicated by a system-wide signal, usually called REQUEST or ARBITRATE. The resolution of an arbitration process is the collection of settled logic values on the busses at the end of the process, which should uniquely identify the competing module having the highest arbitration priority.

Throughout this chapter we use the following notations and assumptions. The set $C = \{c_0, c_1, \ldots, c_{n-1}\}$ denotes the n system modules (chips), which are assumed to be indexed in increasing order of priority. The m wired-OR (open-collector) arbitration busses are denoted by $B = \{b_0, b_1, \ldots, b_{m-1}\}$, where the busses are indexed in increasing order of significance (to be elaborated later). The set $P = \{p_0, p_1, \ldots, p_{n-1}\}$ consists of n distinct arbitration priorities (in increasing order of priority), with p_i being the arbitration priority of module c_i. Arbitration priorities are only a convenient mechanism of encoding the modules' names, and in many asynchronous schemes the arbitration priorities are m-bit vectors that competing modules apply to the m busses during arbitration. When necessary, we denote the bits of an arbitration priority p by $p^{(0)}, p^{(1)}, p^{(2)}, \ldots$, in order of increasing significance. We assume that each module is connected to all busses and can thus read from and potentially write to any bus. All modules follow the same arbitration protocol in interfacing with the busses and reaching conclusions concerning the arbitration process. Finally, we assume that only competing modules apply logic values to the busses; noncompeting modules do not interfere with the busses. All our assumptions are standard design practice in many systems.

3.2.1 Acyclic arbitration protocols

In asynchronous priority arbitration with busses, we restrict the arbitration process to be purely combinational by requiring that the arbitration logic on all the modules together with the arbitration busses form an acyclic circuit. Using combinational logic with asynchronous feedback paths may introduce race conditions and metastable states, which can defer arbitration indef-

initely (see [2, 62, 72]). The acyclic nature of the arbitration logic imposes a partial order on the busses, corresponding to partitioning the busses according to their depth in the arbitration circuitry. This partial order can be extended to a linear order, by having busses at a given depth succeed busses of greater depth, and by arbitrarily ordering busses of the same depth. With a linear order on the busses in mind, the acyclic nature of the arbitration circuitry can be characterized as follows: logic values on higher indexed busses may be used to determine logic values on lower indexed busses, but not vice versa. We formalize this idea in the following definition of an acyclic arbitration protocol.

Definition 9 Let P be a set of arbitration priorities. An *acyclic arbitration protocol* of size m for P is a sequence $F = \langle f_{m-1}, \ldots, f_1, f_0 \rangle$ of m functions, $f_j : P \times \{0,1\}^{m-1-j} \rightarrow \{0,1\}$, for $j = 0, 1, \ldots, m-1$.

In asynchronous priority arbitration with busses, every module has arbitration circuitry that implements the same acyclic arbitration protocol, but with the module's unique arbitration priority as a parameter. The m arbitration busses are linearly ordered from b_{m-1} down to b_0, in accordance with the acyclic nature of the circuit. Informally, function f_j takes an arbitration priority $p \in P$ and $m-1-j$ bit values on the highest $m-1-j$ busses b_{m-1} through b_{j+1}, and determines the bit value that a competing module c with arbitration priority p applies to bus b_j, for $j = 0, 1, \ldots, m-1$. Collectively, an acyclic arbitration protocol F of size m can be interpreted as a function $F : P \times \{0,1\}^m \rightarrow \{0,1\}^m$, that determines the sequence of m logic values that a competing module c with arbitration priority p applies to the m busses when detecting a certain configuration of logic values on the m busses. (Notice that not every function from $\{0,1\}^m$ to $\{0,1\}^m$ constitutes an acyclic arbitration protocol of size m; it has to satisfy the requirements of Definition 9.)

An arbitration process among several contending modules consists of the modules independently applying logic values to the m busses, according to an acyclic arbitration protocol F of size m, until all the busses reach stable logic states. Since acyclic arbitration protocols have no feedback paths, it is guaranteed that any arbitration process among contending modules will terminate after a finite number of steps. To formally define and analyze arbitration processes, however, we first need to discuss some means of measuring the time for asynchronous arbitration mechanisms with busses.

3.2.2 Bus-settling delay: a unit of time

Measuring the arbitration time of asynchronous mechanisms is somewhat problematic. We follow a standard approach taken in many bus systems (see [16, 22, 23, 40, 42, 80, 81]) and measure the arbitration time in units of bus-settling delay. The time unit of bus-settling delay, typically denoted by T_{bus}, is the time it takes for a bus to settle to a stable logic value, once its drivers have stabilized. This time includes the delays introduced by the logic gates driving the bus, the bus propagation delay, and any additional time required to resolve transient effects. In effect, we model an open-collector bus as an OR gate with delay T_{bus}, the time it takes for the output of the gate to stabilize on a valid logic value, once its inputs have reached their final values. This approach models the situation in many bus systems rather accurately.

High speed busses are commonly modeled as analog transmission lines, where it takes finite amount of time for signals to propagate through the bus and bring the bus to a stable logic value. Since busses carry analog signals, the logic value on a bus cannot be used (and in fact is undefined) before the bus reaches a stable digital value. In addition, the response time of logic gates driving the busses and several transient effects need to be considered. In particular, the effect of the wired-OR glitch on bus-settling time and the use of special integration logic at module receivers to reduce this effect (see [5, 18, 42, 81]) indicate that the logic value on a bus may not be used before a unit of time, bus-settling delay, passes.

Some authors carry out a more elaborate analysis of high speed busses, where they take into account distances between modules on the bus and impose certain restrictions on the ordering of modules. Taub [79, 80, 81], for example, assumes a geographical ordering of modules by increasing priorities and equal distances between modules on a bus. Counterexamples to Taub's analysis, where these requirements are not met, were found [3, 87]. In Chapter 4, we introduce and use a digital transmission line model for a bus that takes into account distances and signal propagation. In this chapter, however, our model for the settling of a digital bus makes no restricting assumptions and is applicable to wide classes of systems, where priorities and module locations are not fixed or predetermined.

Using our model of a wired-OR (open-collector) bus as a delay element that exhibits a delay of T_{bus}, we can now model an arbitration process as a sequence of applications of an acyclic arbitration protocol, where each such application completes in one T_{bus} time.

3.2.3 Arbitration processes

We next formally define the notion of an arbitration process of an acyclic arbitration protocol on a set of competing arbitration priorities. We characterize the arbitration process by the collection of the logic values on the m busses at the end of each computation stage. We use $v_j[l]$ to denote the logic value on bus b_j at the end of the lth computation stage, for $j = 0, 1, \ldots, m-1$ and $l = 0, 1, \ldots$. Without loss of generality, we assume that an arbitration process begins with all busses being in logic value 0.

Definition 10 Let P be a set of arbitration priorities, F be an acyclic arbitration protocol of size m for P, and $Q \subset P$ be a set of competing arbitration priorities. The *arbitration process* of F on Q is the successive evaluation of

$$\begin{aligned} v_j[0] &= 0 , \\ v_j[l+1] &= \bigvee_{p \in Q} f_j(p, v_{m-1}[l], \ldots, v_{j+1}[l]) , \end{aligned}$$

for $j = 0, 1, \ldots, m-1$ and $l = 0, 1, \ldots$. We say that the arbitration process takes t stages if $t \geq 0$ is the smallest integer for which $v_j[t] = v_j[t+1]$, for $j = 0, 1, \ldots, m-1$. The *resolution* of the arbitration process is the stable configuration of values $(v_{m-1}[t], \ldots, v_1[t], v_0[t])$.

Definition 10 characterizes an arbitration process as a sequence of successive applications of the acyclic arbitration protocol F to the set of competing arbitration priorities Q and the configuration of the m busses. The arbitration process terminates when no more changes in the state of the busses occur, at which point a resolution is reached. One can verify that any arbitration process of an acyclic arbitration protocol F of size m takes at most m stages. This is the case because at each computation stage of an arbitration process of an acyclic arbitration protocol, at least one more bus stabilizes on its final value.

A better upper bound for the number of stages taken by arbitration processes can be given by the depth of the acyclic arbitration protocol. As discussed above, the acyclic nature of the arbitration logic imposes a partial order on the busses. We can therefore statically partition the m busses into d levels, such that the computation for a bus in a certain level uses only the values of busses in previous levels. More formally, given an acyclic arbitration protocol F of size m, we can simultaneously partition the m functions of F into d nonempty disjoint sets

$F_0, F_1, \ldots, F_{d-1}$, and the m busses of B into d corresponding sets $B_0, B_1, \ldots, B_{d-1}$, such that $f_j \in F_h$ if and only if $b_j \in B_h$, for $0 \leq j \leq m-1$, and $0 \leq h \leq d-1$. The partition must have the property that the computation of a function $f_j \in F_h$ depends only on the arbitration priorities and on values of busses in sets $B_0, B_1, \ldots, B_{h-1}$. The *depth* of an acyclic arbitration protocol F of size m is defined as the smallest d, for which a partition as above exists. The depth of an acyclic arbitration protocol is never greater than its size, since placing each bus in a separate level satisfies the requirements of the above partition and the number of levels in this partition is the size of the protocol. The next theorem shows that any acyclic arbitration protocol of depth d reaches a resolution after at most $t = d$ computation stages.

Theorem 22 *Let P be a set of arbitration priorities, F be an acyclic arbitration protocol of size m for P, and d be the depth of F. Then, for any subset $Q \subset P$ of competing arbitration priorities, the arbitration process of F on Q takes at most d stages.*

Proof. By induction on d, the depth of the acyclic arbitration protocol F.

Base case: $d = 0$. For depth $d = 0$, there are no arbitration busses and the claim holds immediately for arbitrary Q.

Inductive case: $d > 0$. Given an acyclic arbitration protocol $F = \langle f_{m-1}, \ldots, f_1, f_0 \rangle$ of size m and depth d for P, we can partition $F = \bigcup_{h=0}^{d-1} F_h$ and $B = \bigcup_{h=0}^{d-1} B_h$ as discussed above. Without loss of generality, we assume that the last level consists of the r functions and busses with indices $0, 1, \ldots, r-1$. The first $d-1$ levels of F constitute an acyclic arbitration protocol $F' = \bigcup_{h=0}^{d-2} F_h = \langle f_{m-1}, \ldots, f_{r+1}, f_r \rangle$ of size $m-r$ and depth $d-1$ for P. By induction, the arbitration process of F' on Q takes at most $d-1$ stages. That is, for any $r \leq j \leq m-1$ and $l \geq d-1$, we have $v_j[l] = v_j[d-1]$. In addition, according to the acyclic arbitration protocol F, we also have that for any $0 \leq i \leq r-1$ and $k \geq d > 0$,

$$\begin{aligned} v_i[k] &= \bigvee_{p \in Q} f_i(p, v_{m-1}[k-1], \ldots, v_r[k-1]) \\ &= \bigvee_{p \in Q} f_i(p, v_{m-1}[d-1], \ldots, v_r[d-1]) \\ &= v_i[d] \,, \end{aligned}$$

because the dth level depends only on busses b_{m-1} down to b_r and because $k-1 \geq d-1$. This proves that the arbitration process takes at most d stages. ∎

Theorem 22 shows that the number of stages that an arbitration process takes is bounded by the depth of the acyclic arbitration protocol F. This bound represents a standard *static* approach in the analysis of delays in digital circuits, namely, that of counting the number of gates on the longest path from the inputs to the outputs. In later sections of this chapter, however, we introduce and use a novel *dynamic* approach of bounding the number of stages that an arbitration process takes by a careful analysis of the *data-dependent delays* experienced in the arbitration circuits. In doing so, we exhibit arbitration schemes that guarantee termination of any arbitration process in a circuit of size and depth m after a fixed number of stages t, for values of t in the range $0 \leq t \leq m$.

3.2.4 Asynchronous priority arbitration schemes

To complete the definition of asynchronous priority arbitration schemes, we need to introduce the notion of an interpretation function. Suppose we have a set of arbitration priorities P and an acyclic arbitration protocol F of size m for P. An *interpretation function* for P and F is a function WIN : $\{0,1\}^m \rightarrow P$, such that for any $Q \subset P$, with $p \in Q$ being the highest arbitration priority in Q and $(v_{m-1}, \ldots, v_1, v_0)$ being the resolution of the arbitration process of F on Q, we have $\text{WIN}(v_{m-1}, \ldots, v_1, v_0) = p$. Informally, the function WIN interprets the resolution of any arbitration process of F by identifying the highest competing arbitration priority. We are now ready to define an asynchronous priority arbitration scheme for n modules, m busses, and t stages.

Definition 11 An *asynchronous priority arbitration scheme* for n modules, m busses, and t stages is a triplet $\mathcal{A}(n,m,t) = \langle P, F, \text{WIN} \rangle$, where

1. P is a set of n *arbitration priorities*;
2. F is an *acyclic arbitration protocol* of size m for P;
3. WIN is an *interpretation function* for P and F;

such that for any $Q \subset P$, the arbitration process of F on Q takes at most t stages.

Definition 11 emphasizes the role of the arbitration priorities, which are just a mechanism to distinguish between different modules. It will become apparent, however, that careful design

of the codewords used as arbitration priorities has a significant impact on the arbitration time. In the next Section, for example, we demonstrate that by using the set of $(\lg n + 1)$-bit *binomial codes* as arbitration priorities, we can achieve an arbitration time of $t = \frac{1}{2}\lg n$.

3.3 Asynchronous priority arbitration schemes

In this section we first describe two commonly used asynchronous priority arbitration schemes: *linear arbitration*, which with $m = n$ busses arbitrates in time $t = 1$, and *binary arbitration*, which with $m = \lg n$ busses arbitrates in time $t = \lg n$. We then present our new asynchronous scheme, *binomial arbitration*, which with $m = \lg n + 1$ busses arbitrates in time $t = \frac{1}{2}\lg n$.

3.3.1 The linear arbitration scheme

This scheme uses $m = n$ busses and arbitrates among n modules in $t = 1$ stages. To arbitrate, contending module c_i applies a 1 to bus b_i, for $0 \leq i \leq n-1$, and does not interfere with other busses. This translates to module c_i having an n-bit arbitration priority p_i, such that $p_i^{(j)} = 1$ if $i = j$ and $p_i^{(j)} = 0$ otherwise. After $t = 1$ units of time, all the busses stabilize on their final values, and the module with a 1 on the bus with the highest priority is recognized as the winner. This scheme can also be implemented with tri-state busses, since at most one module writes to any given bus. The scheme is also known as *decoded arbitration* and is used in a number of bus systems and interrupt arbitration mechanisms (see [22, 24, 57, 82]).

Formally, we define this scheme as $\textsc{Linear}(n, n, 1) = \langle P, F, \textsc{win}\rangle$, where

1. $P = \{p_i = 0^{n-1-i}\,1\,0^i \;:\; \text{for } i = 0, 1, \ldots, n-1\}$;

2. $F = \langle f_{n-1}, \ldots, f_1, f_0\rangle$, where $f_j(p, v_{n-1} \ldots, v_{j+1}) = p^{(j)}$, for $j = 0, 1, \ldots, n-1$;

3. $\textsc{win}(0^k\,1\,\alpha) = 0^k\,1\,0^{n-1-k} = p_{n-1-k}$, for $0 \leq k \leq n-1$ and any $\alpha \in \{0,1\}^{n-1-k}$.

Notice that although the size of the acyclic arbitration protocol of LINEAR is $m = n$, its depth is only $d = 1$, which according to Theorem 22 implies that the asynchronous linear arbitration scheme takes at most $t = 1$ stages to arbitrate.

3.3.2 The binary arbitration scheme

This scheme uses $m = \lceil \lg n \rceil$ busses and arbitrates among n modules in $t = \lceil \lg n \rceil$ stages. The arbitration priority p_i of module c_i is the binary representation of i, for $0 \leq i \leq n-1$. To arbitrate, contending module c drives its binary priority p onto the m busses, from $p^{(m-1)}$ (the most significant bit of p) onto bus b_{m-1}, down to $p^{(0)}$ (the least significant bit of p) onto bus b_0; the result being the bitwise OR of the binary priorities of the competing modules. During arbitration, each competing module c monitors the busses and disables its drivers according to the following rule: let $p^{(l)}$ be the lth bit of the binary priority p, and let v_l be the binary value observed on bus b_l, for $0 \leq l \leq m-1$. Then if $p^{(l)} = 0$ and $v_l = 1$, module c disables all its bits $p^{(j)}$ for $j < l$. Disabled bits are re-enabled should the condition cease to hold. After $t = \lceil \lg n \rceil$ units of time, all the busses stabilize on their final values, and the module whose arbitration priority appears on the busses is the winner. This scheme was developed by Taub [79], and is also known as encoded arbitration (see [16, 22, 40, 80, 81]).

Formally, we define this scheme $\textsc{Binary}(n, \lceil \lg n \rceil, \lceil \lg n \rceil) = \langle P, F, \textsc{win} \rangle$ as follows. For simplicity of notation we use $m = \lceil \lg n \rceil$.

1. $P = \{p_i = \epsilon_{m-1} \cdots \epsilon_1 \epsilon_0 \ : \ \text{where } \epsilon_{m-1} \cdots \epsilon_1 \epsilon_0 \text{ is the binary representation of } i, \text{ for } i = 0, 1, \ldots, n-1\}$;

2. $F = \langle f_{m-1}, \ldots, f_1, f_0 \rangle$, where

$$f_j(p, v_{m-1} \ldots, v_{j+1}) = \begin{cases} 0 & \text{if } \bigvee_{l=j+1}^{m-1} \left(p^{(l)} = 0 \ \wedge \ v_l = 1 \right) , \\ p^{(j)} & \text{otherwise} , \end{cases}$$

   for $j = 0, 1, \ldots, m-1$;

3. $\textsc{win}(\alpha) = \alpha$, for any $\alpha \in \{0,1\}^m$.

Notice that the size m and the depth d of the acyclic arbitration protocol of BINARY are equal, specifically $m = d = \lceil \lg n \rceil$. This can be verified by noticing that the computation for each bus b_j, where $0 \leq j \leq m-1$, takes into account values on busses b_l, for $j < l \leq m-1$. This implies, according to Theorem 22, that the asynchronous binary arbitration scheme takes at most $t = m = \lceil \lg n \rceil$ stages to arbitrate. On the other hand, it has been shown in [22, 23, 80, 81, 88] that there are examples where a binary arbitration process takes

exactly $\lceil \lg n \rceil$ stages. (Figure 3-1 presents such an example for $n = 16$ modules, $m = \lceil \lg n \rceil = 4$ busses, and $t = m = 4$ stages.) These examples consist of arbitrating among *bad* subsets of arbitration priorities, where at each stage the binary value of exactly one more bit of the highest competing binary priority is resolved. The asynchronous binomial arbitration scheme, presented next, guarantees fast arbitration by employing only certain codewords that exhibit small data-dependent delays.

3.3.3 The binomial arbitration scheme

This scheme uses $m = \lceil \lg n + 1 \rceil$ busses to arbitrate among n modules in $t = \left\lceil \frac{1}{2} \lg n \right\rceil$ stages. This scheme's acyclic arbitration protocol and interpretation function are identical to those of the binary arbitration scheme, and thus the same hardware can be used. The only difference is that *binomial codes* are used as arbitration priorities rather than all the 2^m possible m-bit codewords of binary arbitration. Alternatively, with m busses, this scheme can arbitrate among 2^{m-1} modules in $t = \left\lceil \frac{1}{2}(m-1) \right\rceil$ stages. We next describe the binomial codes and begin by defining the interval-number of a binary codeword.

Definition 12 The *interval-number* of a binary codeword p is the number of intervals of consecutive 1's or 0's that it contains, disregarding leading 0's.

Thus, for example, the interval-number of 001011 is 3, the interval-number of 0000 is 0, and the interval-number of 10101010 is 8. In general, an m-bit binary codeword p with interval-number r, has the form $p = 0^{m_0}1^{m_1}0^{m_2}1^{m_3}\cdots\delta^{m_r}$, where $\delta \in \{0,1\}$; $m_0 \geq 0$; $m_j > 0$ for $1 \leq j \leq r$; and $\sum_{j=0}^{r} m_j = m$. We next define the binomial codes of length m.

Definition 13 The set of *binomial codes* of length m, denoted by $D(m)$, is the set of all the m-bit binary codewords that have interval-number at most $\left\lceil \frac{1}{2}(m-1) \right\rceil$.

The binomial codes of length m are in fact all the m-bit codewords, that, after deleting leading 0's have at most $\left\lceil \frac{1}{2}(m-1) \right\rceil$ intervals of consecutive 1's or 0's. For example, the binomial codes of length 4 is $D(4) = \{0000, 0001, 0010, 0011, 0100, 0110, 0111, 1000, 1100, 1110, 1111\}$, consisting of 11 codewords that have interval-number at most 2. As another example, the binomial codes that were used in the Section 3.1 (the example of Figure 3-2) are

$D(5) = \{00000, 00001, 00010, 00011, 00100, 00110, 00111, 01000, 01100, 01110, 01111, 10000, 11000, 11100, 11110, 11111\}$, consisting of the 16 codewords of length 5 with interval-number at most 2. For general values of m, Corollary 24 in Section 3.4 shows that there are at least 2^{m-1} binomial codes of length m. By taking $m = \lceil \lg n + 1 \rceil$, this translates to at least $2^{\lceil \lg n+1 \rceil - 1} \geq n$ binomial codes, which means that there are enough arbitration priorities for n modules.

Formally, we define this scheme $\textsc{Binomial}(n, \lceil \lg n + 1 \rceil, \lceil \frac{1}{2} \lg n \rceil) = \langle P, F, \textsc{win} \rangle$ as follows. We use $m = \lceil \lg n + 1 \rceil$ and $t = \lceil \frac{1}{2} \lg n \rceil$ for simplicity of notation.

1. $P = D(m)$;

2. $F = \langle f_{m-1}, \ldots, f_1, f_0 \rangle$, where

$$f_j(p, v_{m-1} \ldots, v_{j+1}) = \begin{cases} 0 & \text{if } \bigvee_{l=j+1}^{m-1} \left(p^{(l)} = 0 \wedge v_l = 1 \right), \\ p^{(j)} & \text{otherwise}, \end{cases}$$

   for $j = 0, 1, \ldots, m-1$;

3. $\textsc{win}(\alpha) = \alpha$, for any $\alpha \in \{0,1\}^m$.

It remains to show that the asynchronous binomial arbitration scheme indeed arbitrates among n modules in at most $t = \lceil \frac{1}{2} \lg n \rceil$ stages. Notice that a standard static analysis of the arbitration circuitry, as given for example in Theorem 22, does not give the desired result, since both the size and the depth of the acyclic arbitration protocol F of binomial arbitration are $m = d = \lceil \lg n + 1 \rceil$. In Section 3.4, we use a novel dynamic approach of analyzing the data-dependent delays experienced in arbitration processes, and prove the correctness of the asynchronous binomial arbitration scheme as a special case of the generalized binomial arbitration scheme.

3.4 Generalized Binomial Arbitration

In this section we extend the ideas of the asynchronous binomial arbitration scheme by presenting the *generalized binomial arbitration* scheme that with m busses and in at most t stages arbitrates among $n = \sum_{l=0}^{t} \binom{m}{l}$ modules. By Stirling's approximation, the asymptotic bus-time tradeoff of generalized binomial arbitration is $m \approx \frac{1}{e} t n^{1/t}$. This bus-time tradeoff is of great practical interest, enabling system designers to achieve a desirable balance between amount of

hardware and speed. The performance of generalized binomial arbitration is based on analysis of data-dependent delays.

3.4.1 Generalized binomial codes

We first extend Definition 13 by defining the set of generalized binomial codes of length m and diversity r.

Definition 14 The set of *generalized binomial codes* of length m and diversity r, denoted by $G(m,r)$, is the set of all m-bit binary codewords that have interval-number at most r.

Generalized binomial codes serve as arbitration priorities for the generalized binomial arbitration scheme. The next lemma determines the cardinality of the set of the generalized binomial codes of length m and diversity r.

Lemma 23 *The set $G(m,r)$ contains $\sum_{l=0}^{r}\binom{m}{l}$ distinct codewords.*

Proof. To simplify the counting, we take all the codewords in $G(m,r)$ and append a 0 at their beginning. This results in a set of $(m+1)$-bit words, that begin with a 0 and have at most r switching points from a consecutive interval of 0's to a consecutive interval of 1's and vice versa. The number of such words is $\sum_{l=0}^{r}\binom{m}{l}$, since for any $0 \le l \le r$ there are exactly $\binom{m}{l}$ possibilities of choosing l switching points out of m possible positions. ■

Corollary 24 *There are at least 2^{m-1} binomial codes of length m.*

Proof. By our notation, the set $D(m)$ of binomial codes of length m, is defined by $D(m) = G(m, \lceil \frac{1}{2}(m-1) \rceil)$. According to Lemma 23, we have

$$|D(m)| = \sum_{l=0}^{\lceil \frac{1}{2}(m-1) \rceil} \binom{m}{l} .$$

This sum includes the first $\lceil \frac{1}{2}(m-1) \rceil + 1$ binomial coefficients, which constitute at least a half of all the $m+1$ binomial coefficients $\binom{m}{l}$. Since the binomial coefficients are symmetric, that is, $\binom{m}{l} = \binom{m}{m-l}$, the above partial sum is at least a half of the full sum, which is 2^m. We therefore conclude that $|D(m)| \ge \frac{1}{2} \cdot 2^m = 2^{m-1}$. ■

3.4.2 The generalized binomial arbitration scheme

This scheme uses m busses and arbitrates in at most t stages, for $0 \leq t \leq m$. With the m and t parameters determined, this scheme can arbitrate among at most $n = \sum_{l=0}^{t} \binom{m}{l}$ modules. The acyclic arbitration protocol and the interpretation function of this scheme are identical to those of the binary arbitration scheme of Section 3.3.2, and thus the same hardware can be used. The only difference is that generalized binomial codes from $G(m,t)$ are used as arbitration priorities.

Formally, we define this scheme GENERALIZED-BINOMIAL$(n,m,t) = \langle P, F, \text{WIN}\rangle$, where $n = \sum_{l=0}^{t} \binom{m}{l}$, as follows.

1. $P = G(m,t)$;

2. $F = \langle f_{m-1}, \ldots, f_1, f_0\rangle$, where

$$f_j(p, v_{m-1} \ldots, v_{j+1}) = \begin{cases} 0 & \text{if } \bigvee_{l=j+1}^{m-1} \left(p^{(l)} = 0 \wedge v_l = 1\right), \\ p^{(j)} & \text{otherwise}, \end{cases}$$

   for $j = 0, 1, \ldots, m-1$;

3. $\text{WIN}(\alpha) = \alpha$, for $\alpha \in \{0,1\}^m$.

The idea behind generalized binomial arbitration is that the interval-number of the highest competing arbitration priority bounds the number of arbitration stages. In binary arbitration, where all the 2^m possible m-bit codewords are used, there are arbitration processes that can take as many as m stages, where at each stage one more bit of the highest competing arbitration priority is resolved. For generalized binomial arbitration, however, we select codewords that have at most t intervals of consecutive 1's or 0's. The following theorem uses data-dependent analysis to argue that any arbitration process takes at most r stages, where r is the interval-number of the highest competing arbitration priority, by showing that at each stage the arbitration process resolves at least one more interval of consecutive bits, rather than a single bit.

Theorem 25 *Consider a generalized binomial arbitration process on m busses. Let Q be the set of competing arbitration priorities, p be the highest arbitration priority in Q, and r be the interval-number of p. Then after s stages, for any $s \geq r$, bus b_j carries the logic value $p^{(j)}$, for $0 \leq j \leq m-1$.*

Proof. We prove the theorem by induction on r for arbitrary values of m. We use the notation $v_j[k]$ to denote the logic value on bus b_j at the end of stage k, for $j = 0, 1, \ldots, m-1$ and $k = 0, 1, \ldots$.

Base case: $r = 0$. The codeword p consists of m consecutive 0's, that is, $p^{(j)} = 0$ for $j = 0, 1, \ldots, m-1$. Since p is the highest arbitration priority in Q, then any $q \in Q$ must also have $q^{(j)} = 0$ for $j = 0, 1, \ldots, m-1$. By our assumption that all the m busses are initially in logic value 0, and since according to the acyclic arbitration protocol no module ever applies a 1 to any of these busses, the m busses remain in logic value 0 forever. In other words, after s stages, for any $s \geq r = 0$, we have $v_j[s] = v_j[0] = 0 = p^{(j)}$, for $j = 0, 1, \ldots, m-1$, which proves the claim.

Inductive case: $r > 0$. The codeword p has m bits and interval-number r, and is thus of the form $p = 0^{m_0}1^{m_1}0^{m_2}1^{m_3}\ldots\delta^{m_r}$, where $\delta \in \{0,1\}$; $m_0 \geq 0$; $m_j > 0$ for $1 \leq j \leq r$; and $\sum_{j=0}^{r} m_j = m$. We first concentrate on the first $r-1$ intervals of p, and define the set R of reduced codewords of length $\hat{m} = m - m_r = \sum_{j=0}^{r-1} m_j$, by ignoring the last m_r bits of the codewords of Q. One can verify that $\hat{p}$, the reduced version of p, is the highest codeword in R, because we discarded the m_r least significant bits of codewords in Q. Furthermore, the interval-number of $\hat{p}$ is $r-1$, since the last interval of p of the form δ^{m_r} was ignored. By applying the claim inductively with $\hat{m}$ busses, the set of competing arbitration priorities R, and the highest arbitration priority $\hat{p}$ of interval-number $r-1$, we find that after $r-1$ stages the most significant $\hat{m} = m - m_r$ busses stabilize to the bits of $\hat{p}$. That is, for any $k \geq r-1$, we have $v_j[k] = v_j[r-1] = \hat{p}^{(j)} = p^{(j)}$, for $m_r \leq j \leq m-1$. We now consider the last m_r busses, $b_{m_r-1}, \ldots, b_1, b_0$. There are two cases to consider:

$\delta = 1$ The rth interval of p is an interval of m_r consecutive 1's, that is, $p^{(i)} = 1$ for $i = 0, 1, \ldots, m_r - 1$. After k stages, for any $k \geq r-1$, the most significant $m - m_r$ busses carry the bits of p, and therefore there is no l in the range $0 \leq l \leq m-1$, with $v_l[k] = 1$ and $p^{(l)} = 0$. As a result, the module with arbitration priority p applies all its last m_r consecutive 1's. Therefore, for any $s \geq r$ and $i = 0, 1, \ldots, m_r - 1$, we have $v_i[s] = v_i[r] = 1 = p^{(i)}$, since the busses implement a wired-OR in one stage.

$\delta = 0$ The rth interval of p is an interval of m_r consecutive 0's, that is, $p^{(i)} = 0$ for $i = 0, 1, \ldots, m_r - 1$. Since p is the highest arbitration priority in Q, then for any arbitration

priority $q \in Q$, $q \neq p$, there must exist an l in the range $m_r \leq l \leq m-1$, with $p^{(l)} = 1$ and $q^{(l)} = 0$. After k stages, for any $k \geq r-1$, the most significant $m - m_r$ busses carry the bits of p, and therefore any module with arbitration priority $q \neq p$ disables at least its last m_r bits. As a result, for any $s \geq r$ and $i = 0, 1, \ldots, m_r - 1$, we have $v_i[s] = v_i[r] = 0 = p^{(i)}$, because the busses implement a wired-OR in one stage and no module applies a 1 to busses b_0 through b_{m_r-1} anymore.

Thus, after s stages, for $s \geq r$, the m busses carry the corresponding bits of p. ■

The following corollary shows that by taking $G(m,t)$, the generalized binomial codes of length m and diversity t, as arbitration priorities, we guarantee that any arbitration process completes in at most t stages.

Corollary 26 *Consider* GENERALIZED-BINOMIAL(n,m,t), *the generalized binomial arbitration scheme. For any subset of arbitration priorities* $Q \subset G(m,t)$, *the corresponding arbitration process takes at most* t *stages.*

Proof. Let p be the highest arbitration priority in Q. Since the interval-number of p is at most t, Theorem 25 guarantees that the arbitration process on Q, with p as the highest arbitration priority, takes no more than t stages. ■

3.4.3 Tradeoff of generalized binomial arbitration

The generalized binomial arbitration scheme achieves a bus-time tradeoff of the form $n = \sum_{l=0}^{t} \binom{m}{l}$, which by Stirling's formula exhibits asymptotic behavior $m \approx \frac{1}{e} t n^{1/t}$. Figure 3-3 demonstrates this bus-time tradeoff for a system with n modules. The horizontal axis represents m, the number of arbitration busses used, which varies from $m = \lg n$ to $m = n$. The arbitration time t, measured in units of bus-settling delay (arbitration stages), is marked on the vertical axis. The arbitration time varies between $t = 1$ to $t = \lg n$ stages. Generalized binomial arbitration reduces to binary arbitration with $m = \lg n$ busses, to binomial arbitration with $m = \lg n + 1$ busses, and to a modified version of linear arbitration (see Section 3.5.2 for the canonical form of linear arbitration) with $m = n$ busses.

Figure 3-3 demonstrates that neither linear arbitration nor binary arbitration efficiently utilize the resources. For example, increasing the number of busses used in binary arbitration by

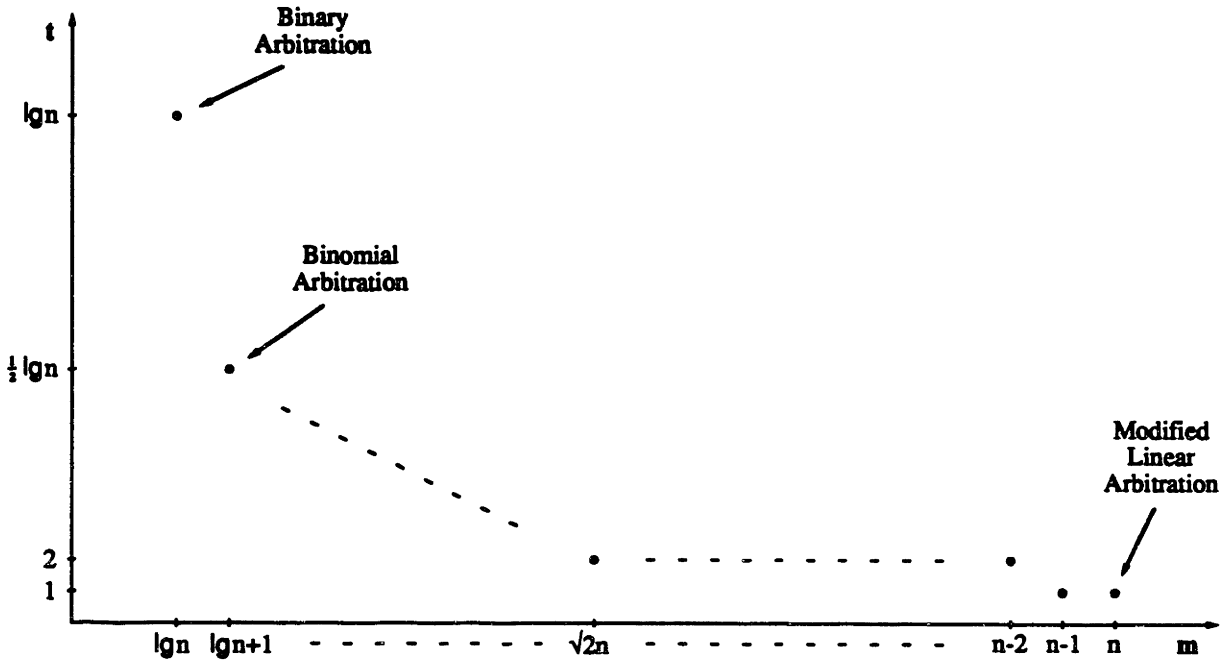

Figure 3-3: Bus-time tradeoff of the generalized binomial arbitration scheme for n modules, using $\lg n \leq m \leq n$ busses and $1 \leq t \leq \lg n$ stages.

one, results in speeding up the arbitration process by a factor of 2, as exhibited by our binomial arbitration scheme. On the other hand, allowing another time unit over linear arbitration enables reducing the number of busses from n to approximately $\sqrt{2n}$.

Notice, however, that in order to achieve another factor-of-2 improvement in the arbitration time, adding another constant number of busses to the $\lg n$ busses is not enough. Asymptotically, as n grows without bound, we need to use more than $(1+\epsilon)\lg n$ busses, for $\epsilon > 0.232$, in order for the sum $\sum_{l=0}^{t}\binom{m}{l}$, with $t = \frac{1}{4}\lg n$, to be at least n. This can be verified by Stirling's formula, since when m is greater than $\lg n$ but smaller than $1.232\lg n$, and when $t = \frac{1}{4}\lg n < m/4$, the sum of the first $m/4$ binomial coefficients $\binom{m}{l}$, for $0 \leq l \leq m/4$, does not exceed n. This demonstrates that our binomial arbitration scheme, which uses $\lg n + 1$ busses, exhibits a most economic balance, much more so than the binary arbitration scheme. Other authors [23] have also discovered that by excluding certain codewords, the arbitration time of binary arbitration can be reduced. Here, however, we give the first general scheme that provides a full spectrum of bus-time tradeoff.

3.5 Properties of asynchronous priority arbitration schemes

In this section we discuss properties and capabilities of general asynchronous priority arbitration schemes with busses, which were defined in Section 3.2.4. We first describe several properties and assumptions regarding asynchronous priority arbitration schemes with busses. We then define a canonical form for acyclic arbitration protocols that is easier to analyze and reason about than arbitrary acyclic arbitration protocols. Finally, we focus on the bus-time tradeoff of general synchronous priority arbitration schemes and present some lower bound arguments that demonstrate the efficiency of our schemes.

3.5.1 General properties and assumptions

Asynchronous priority arbitration schemes that employ busses arbitrate among contending modules by having the modules read logic values from the busses and apply logic values to the busses, according to an underlying acyclic arbitration protocol. For an asynchronous priority arbitration scheme $\mathcal{A} = \langle P, F, \text{WIN} \rangle$ that employs m busses, the acyclic arbitration protocol F is a sequence of m functions, each responsible for applying a binary value to a separate bus, based on the competing module's arbitration priority and on logic values on higher indexed busses. The acyclic nature of the arbitration protocol F guarantees termination of any arbitration process in at most $t = m$ stages, as was formally discussed in Section 3.2.3. We are also interested, however, is asynchronous priority arbitration schemes that arbitrate in t stages, for any value of t in the range $0 \leq t \leq m$.

The configurations of the m arbitration busses play a fundamental role in the analysis of arbitration processes. A configuration of the m busses at any given time is simply the m-bit vector of logic values on the busses. We denote a general configuration on the m busses by $v = (v_{m-1}, \ldots, v_1, v_0)$, and for arbitration processes we use $v[k] = (v_{m-1}[k], \ldots, v_1[k], v_0[k])$, for $k \geq 0$, to denote the configuration of the m busses at stage k. We assume that any arbitration process starts from a "clean" configuration of all 0's, that is, $v_j[0] = 0$ for $j = 0, 1, \ldots, m-1$. An acyclic arbitration protocol F of size m can be thought of as a function that maps an arbitration priority p and a configuration v to an m-bit vector u that a contending module c with arbitration priority p applies to the m busses, when detecting the configuration v. When convenient, we use the vector notation $F(p, v) = u$ to describe this situation.

For an asynchronous priority arbitration scheme, $\mathcal{A}(n, m, t) = \langle P, F, \text{WIN} \rangle$, on n modules, m busses, and t stages, any arbitration process on a subset $Q \subset P$ takes at most t computation stages. There may be, however, certain arbitration processes that take less than t stages, but it is guaranteed that after t stages, the busses are always stable. Since $\mathcal{A} = \langle P, F, \text{WIN} \rangle$ implements priority arbitration and since there are n modules in the system, there must be at least n distinct winning configurations, each being mapped by the interpretation function WIN to a unique arbitration priority p_i, which identifies module c_i as the winner of an arbitration process. Some modules may have more than one winning configuration, as is the case for example with the linear arbitration scheme of Section 3.3.1, but each module must have at least one. Because the number of intermediate and winning configurations in arbitration processes is hard to track, it is difficult to analyze the behavior of arbitration processes. In Section 3.5.2, we show how to translate arbitration protocols into a canonical form, which has the same arbitration power, but is easier to analyze.

3.5.2 Canonical form for arbitration protocols

In an arbitration process of an asynchronous priority arbitration scheme with busses, the competing module c with the highest arbitration priority p should direct the arbitration process to a winning configuration v that identifies it, that is, $\text{WIN}(v) = p$. This should be the case no matter which of the modules with arbitration priorities smaller than p participate in the arbitration process. For competing module c_i with arbitration priority p_i, therefore, there may be as many as 2^i different arbitration processes that module c_i should win, corresponding to all possible subsets of the modules $\{c_0, c_1, \ldots, c_{i-1}\}$ participating in the arbitration process. To simplify the analysis of arbitration processes, we introduce a canonical form of arbitration protocols, which has the same arbitration power, but is easier to analyze.

Definition 15 Let $P = \{p_0, p_1, \ldots, p_{n-1}\}$ be a set of n distinct arbitration priorities and let $F = \langle f_{m-1}, \ldots, f_1, f_0 \rangle$ be an acyclic arbitration protocol of size m for P. We say that F is in *canonical form*, if for any configuration $v = (v_{m-1}, \ldots, v_1, v_0)$, for any $j = 0, 1, \ldots, m-1$, for any $i = 0, 1, \ldots, n-1$, and for any $0 \leq k \leq i$, we have

$$f_j(p_i, v_{m-1}, \ldots, v_{j+1}) = 0 \implies f_j(p_k, v_{m-1}, \ldots, v_{j+1}) = 0 \, .$$

Definition 15, in effect, defines a canonical acyclic arbitration protocol as one that maps any arbitration priority p and configuration v to an m-bit vector u that "shadows" any activity of arbitration priorities of lesser priority that p. The definition guarantees that if a module applies a 0 to a certain bus in response to some configuration v of the busses, then no module with lesser priority applies a 1 to that bus in response to the same configuration v. In other words, for any arbitration priorities p and q, with p being of higher priority than q, and for any configuration v, the m-bit vector $F(p, v)$ is never component-wise smaller than the m-bit vector $F(q, v)$. In analyzing arbitration processes of canonical acyclic arbitration protocols, therefore, it is sufficient to focus only on the behavior of the highest competing arbitration priority p, since the protocol for p always "shadows" the behavior of smaller arbitration priorities. We call an asynchronous priority arbitration scheme *canonical* if its acyclic arbitration protocol is canonical. We typically denote that an arbitration scheme or an arbitration protocol are canonical by putting a bar over them, as in $\bar{A}$ or $\bar{F}$. Analyzing canonical asynchronous priority arbitration schemes is an easier task. The next theorem demonstrates that analyzing canonical asynchronous priority arbitration schemes is also general enough.

Theorem 27 *Let $\mathcal{A}(n, m, t) = \langle P, F, \text{WIN} \rangle$ be an asynchronous priority arbitration scheme on n modules, m busses, and t stages. Then there is also a canonical asynchronous priority arbitration scheme $\bar{\mathcal{A}}(n, m, t) = \langle P, \bar{F}, \text{WIN} \rangle$ on n modules, m busses, and t stages.*

Proof. To define the canonical asynchronous priority arbitration scheme $\bar{\mathcal{A}} = \langle P, \bar{F}, \text{WIN} \rangle$, we need only define the canonical acyclic arbitration protocol $\bar{F}$; the arbitration priorities P and the interpretation function WIN are identical to those of $\mathcal{A}$. We define $\bar{F} = \langle \bar{f}_{m-1}, \ldots, \bar{f}_1, \bar{f}_0 \rangle$ as follows. For any configuration $v = (v_{m-1}, \ldots, v_1, v_0)$, for any $j = 0, 1, \ldots, m-1$, and for any $i = 0, 1, \ldots, n-1$, we define

$$\bar{f}_j(p_i, v_{m-1}, \ldots, v_{j+1}) = \bigvee_{l=0}^{i} f_j(p_l, v_{m-1}, \ldots, v_{j+1}) \; .$$

In fact, we define the m-bit vector that module c_i with arbitration priority p_i applies to the m busses under protocol $\bar{F}$ in response to a configuration v, to be the bitwise OR of the m-bit vectors that modules $c_0, c_1, \ldots, c_i$ with corresponding arbitration priorities $p_0, p_1, \ldots, p_i$ apply to the m busses under protocol F in response to the same configuration v.

To show that $\bar{A} = \langle P, \bar{F}, \text{WIN} \rangle$ is a canonical asynchronous priority arbitration scheme on n modules, m busses, and t stages, we first notice that P is a set of n distinct arbitration priorities, as required. The arbitration protocol $\bar{F} = \langle \bar{f}_{m-1}, \ldots, \bar{f}_1, \bar{f}_0 \rangle$ is acyclic, since by definition, each function $\bar{f}_j$, for $j = 0, 1, \ldots, m-1$, takes an arbitration priority $p \in P$ and $m-1-j$ bit values $(v_{m-1}, \ldots, v_{j+1})$ and produces one bit, as required. Furthermore, $\bar{F} = \langle \bar{f}_{m-1}, \ldots, \bar{f}_1, \bar{f}_0 \rangle$ is in canonical form, since for any configuration $v = (v_{m-1}, \ldots, v_1, v_0)$, for any $j = 0, 1, \ldots, m-1$, for any $i = 0, 1, \ldots, n-1$, and for any $0 \leq k \leq i$, we have

$$\begin{aligned} & \bar{f}_j(p_i, v_{m-1}, \ldots, v_{j+1}) = 0 \\ \Longrightarrow \quad & \bigvee_{l=0}^{i} f_j(p_l, v_{m-1}, \ldots, v_{j+1}) = 0 \\ \Longrightarrow \quad & \bigvee_{l=0}^{k} f_j(p_l, v_{m-1}, \ldots, v_{j+1}) = 0 \\ \Longrightarrow \quad & \bar{f}_j(p_k, v_{m-1}, \ldots, v_{j+1}) = 0 \,, \end{aligned}$$

as required by Definition 15. We then have that $\bar{F}$ is a canonical acyclic arbitration protocol of size m for P.

We now argue that for any $Q \subset P$, the arbitration process of $\bar{F}$ on Q takes at most t stages. Let $p_i \in Q$ be the highest arbitration priority in Q. Because $\bar{F}$ is in canonical form, the arbitration process of $\bar{F}$ on $\{p_i\}$ is indistinguishable from the arbitration process of $\bar{F}$ on Q. (Under $\bar{F}$, arbitration priority p_i always "shadows" the activity of Q.) By our definition of $\bar{F}$, the arbitration process of $\bar{F}$ on $\{p_i\}$ is an exact simulation of the arbitration process of F on $\{p_0, p_1, \ldots, p_i\}$, which by definition of A takes at most t stages. We then conclude that the arbitration process of $\bar{F}$ on $\{p_i\}$ takes at most t stages, which also means that the arbitration process of $\bar{F}$ on Q takes at most t stages.

Last, we verify that the function WIN is indeed an interpretation function for P and $\bar{F}$. Let $Q \subset P$ be a set of competing arbitration priorities and let $p_i \in Q$ be the highest arbitration priority in Q. Let v be the resolution of the arbitration process of $\bar{F}$ on Q. As argued above, v is also the resolution of the arbitration process of $\bar{F}$ on $\{p_i\}$, which is the resolution of the arbitration process of F on $\{p_0, p_1, \ldots, p_i\}$. Since p_i is also the highest arbitration priority in $\{p_0, p_1, \ldots, p_i\}$, and since WIN is an interpretation function for P and F, we have $\text{WIN}(v) = p_i$, which implies that WIN is also an interpretation function for P and $\bar{F}$.

This completes the proof that $\bar{A} = \langle P, \bar{F}, \text{WIN} \rangle$ is a canonical asynchronous priority arbitration scheme on n modules, m busses, and t stages. ■

Theorem 27 shows that canonical acyclic arbitration protocols have the same arbitration power as other acyclic arbitration protocols. The proof transforms an acyclic arbitration protocol F into a canonical acyclic arbitration protocol $\bar{F}$, by having module c_i with arbitration priority p_i be paranoid and always assume that all the modules $c_0, c_1, \ldots, c_{i-1}$ with arbitration priorities $p_0, p_1, \ldots, p_{i-1}$ also participate in its arbitration processes. Under protocol $\bar{F}$, then, module c_i responds to any configuration by simulating the combined responses of modules $c_0, c_1, \ldots, c_i$ to the same configuration under protocol F.

For example, transforming the asynchronous linear arbitration scheme of Section 3.3.1 to canonical form, results in a scheme where to arbitrate, contending module c_i applies a 1 to busses $b_i, \ldots, b_0$, and does not interfere with other busses. After $t = 1$ units of time, all the busses stabilize on their final values, and the module with a 1 on the highest indexed bus is recognized as the winner. Formally, this scheme is derived from $\text{LINEAR}(n, n, 1) = \langle P, F, \text{WIN} \rangle$, and is defined as $\text{CANONICAL-LINEAR}(n, n, 1) = \langle P, \bar{F}, \text{WIN} \rangle$, where

1. $P = \{p_i = 0^{n-1-i}\ 1\ 0^i : \text{ for } i = 0, 1, \ldots, n-1\}$;
2. $\bar{F} = \langle \bar{f}_{n-1}, \ldots, \bar{f}_1, \bar{f}_0 \rangle$, where for $j = 0, 1, \ldots, n-1$ and $i = 0, 1, \ldots, n-1$, we have $\bar{f}_j(p_i, v_{n-1} \ldots, v_{j+1}) = 1$ if $j \leq i$ and $\bar{f}_j(p_i, v_{n-1} \ldots, v_{j+1}) = 0$ if $j > i$;
3. $\text{WIN}(0^k\ 1\ \alpha) = 0^k\ 1\ 0^{n-1-k} = p_{n-1-k}$, for $0 \leq k \leq n-1$ and any $\alpha \in \{0,1\}^{n-1-k}$.

We use the canonical forms of arbitration protocols for analysis purposes only. In practice, there may be several drawbacks to using canonical forms of acyclic arbitration protocols, due to their overly paranoid behavior. The advantage of canonical forms arises in investigating the computational power of asynchronous priority arbitration schemes with busses. When analyzing an asynchronous priority arbitration scheme for n modules, there may be a need to investigate all possible 2^n arbitration processes, corresponding to the 2^n possible subsets of competing modules. For a canonical asynchronous priority arbitration scheme on n modules, however, there are exactly n different arbitration processes to analyze, and there are exactly n reachable winning configurations. This is the case since for canonical protocols, higher arbitration priorities always "shadow" the activity of smaller arbitration priorities.

3.5.3 The bus-time tradeoff

Analytically, the simplest way to define the optimal bus-time tradeoff of asynchronous priority arbitration schemes is to fix m, the number of arbitration busses used, to fix t, the number of arbitration stages allowed, and to investigate the largest number of modules that can be arbitrated by some asynchronous priority arbitration scheme with m busses in at most t stages. Formally, we define $\mathcal{R}(m,t)$, for $m \geq 0$ and $t \geq 0$, as the smallest integer, such that any $\mathcal{A}(n,m,t) = \langle P, F, \text{WIN} \rangle$, an asynchronous priority arbitration scheme for n modules, m busses, and t stages, satisfies $n \leq \mathcal{R}(m,t)$. Theorem 27 implies that in investigating $\mathcal{R}(m,t)$, it suffices to focus only on canonical asynchronous priority arbitration schemes with m busses and t stages. We take advantage of this fact when convenient. The following lemma shows that the value of $\mathcal{R}(m,t)$ is well defined for any $m \geq 0$ and $t \geq 0$.

Lemma 28 *For any $m \geq 0$ and $t \geq 0$, we have $\mathcal{R}(m,t) \leq 2^m$.*

Proof. Let $\bar{\mathcal{A}}(n,m,t) = \langle P, \bar{F}, \text{WIN} \rangle$ be a canonical asynchronous priority arbitration scheme on n modules, m busses, and t stages. With m busses there are no more than 2^m possible configurations of binary values on the busses, but there must be exactly n distinct resolutions of arbitration processes of $\bar{\mathcal{A}}$. We must then have $n \leq 2^m$. Since this bound holds for arbitrary canonical asynchronous priority arbitration schemes, we also have $\mathcal{R}(m,t) \leq 2^m$. ■

Lemma 28 states that no more than 2^m modules can be arbitrated with m busses. Given enough time, we can arbitrate among exactly $n = 2^m$ modules, as the following lemma implies.

Lemma 29 *For any $m \geq 0$ we have $\mathcal{R}(m,m) = 2^m$.*

Proof. The asynchronous binary arbitration scheme of Section 3.3.2 arbitrates among n modules, using $m = \lg n$ busses and $t = m = \lg n$ stages. Said another way, with m busses and in $t = m$ stages, exactly $n = 2^m$ modules can be arbitrated. Combining this with the result of Lemma 28, we have $\mathcal{R}(m,m) = 2^m$. ■

From Lemmas 28 and 29 it follows that there is no advantage in using more units of time than the number of busses. We summarize this observation in the following theorem.

Theorem 30 *For any $t \geq m \geq 0$ we have $\mathcal{R}(m,t) = 2^m$.*

The next theorem shows that $\mathcal{R}(m,t)$ is monotonicly nondecreasing in both m and t.

Theorem 31 *For any $m \geq 0$ and $t \geq 0$ we have*

1. $\mathcal{R}(m+1,t) \geq \mathcal{R}(m,t)$,
2. $\mathcal{R}(m,t+1) \geq \mathcal{R}(m,t)$.

Proof. Increasing the number of arbitration busses or the number of arbitration stages cannot decrease the number of modules that can arbitrate. We show this by describing how to simulate any asynchronous priority arbitration scheme $\mathcal{A}(n,m,t) = \langle P, F, \text{WIN} \rangle$ by a scheme with more busses or time.

1. Define $\mathcal{A}'(n,m+1,t) = \langle P', F', \text{WIN}' \rangle$ as follows. The arbitration priorities $P' = P$ are unchanged. If $F = \langle f_{m-1}, \ldots, f_1, f_0 \rangle$ then define $F' = \langle f'_m, f'_{m-1}, \ldots, f'_1, f'_0 \rangle$, where $f'_j = f_{j-1}$ for $j = 1,2,\ldots,m$, and $f'_0(p,v) = 0$ for any $p \in P'$ and $v \in \{0,1\}^m$. Finally, we define $\text{WIN}'(v_m, v_{m-1}, \ldots, v_1, v_0) = \text{WIN}(v_m, v_{m-1}, \ldots, v_1)$ for $v_j \in \{0,1\}$ and $j = 0,1,\ldots,m$. Informally, the asynchronous priority arbitration scheme $\mathcal{A}'$ simulates $\mathcal{A}$ on the first m busses and ignores the last bus. Since this simulation method works for arbitrary asynchronous priority arbitration schemes, we then have $\mathcal{R}(m+1,t) \geq \mathcal{R}(m,t)$.
2. Since $\mathcal{A}(n,m,t) = \langle P, F, \text{WIN} \rangle$ arbitrates among any $Q \subset P$ in at most t stages, it also arbitrates in at most $t+1$ stages, which shows that $\mathcal{R}(m,t+1) \geq \mathcal{R}(m,t)$. ∎

We now turn to investigate $\mathcal{R}(m,t)$, for values of $m \geq t \geq 0$. The next lemma investigates the case $t = 0$.

Lemma 32 *For any $m \geq 0$ we have $\mathcal{R}(m,0) = 1$.*

Proof. With $t = 0$ stages and m busses to arbitrate, for any value of $m \geq 0$, the reading of the busses after $t = 0$ stages consists of m zeros. It then follows that we can arbitrate among at most one module, that is $\mathcal{R}(m,0) = 1$ for any $m \geq 0$. ∎

We next investigate $\mathcal{R}(m,t)$ for the case $t = 1$. The following theorem demonstrates that any canonical asynchronous priority arbitration scheme with m busses can be in at most $m+1$ different configurations after $t = 1$ stages.

Theorem 33 *Let $\bar{A}(n,m,t) = \langle P, \bar{F}, \text{WIN} \rangle$ be a canonical asynchronous priority arbitration scheme on n modules, m busses, and t stages. Let $U = \{u : u = \bar{F}(p, 0^m)$ for $p \in P\}$ be the set of all possible responses of modules of A to the initial configuration $v = 0^m$. Then we have $|U| \leq m + 1$.*

Proof. For convenience of analysis, we refine the definition of U. Corresponding to $P = \{p_0, p_1, \ldots, p_{n-1}\}$, the set of responses U is a set of m-bit vectors $U = \{u_i : u_i = \bar{F}(p_i, 0^m)$ for $i = 0, 1, \ldots, n-1\}$. Each m-bit vector $u_i \in U$, is the response of p_i under $\bar{F}$ to the configuration $v = 0^m$. Since $\bar{F}$ is a canonical acyclic arbitration protocol and since the arbitration priorities are indexed in increasing order of priority, we must have that for any $0 \leq k \leq i \leq n-1$, the m-bit vector u_i has a 1 at component j if the m-bit vector u_k has a 1 at component j, for $j = 0, 1, \ldots, m-1$. This implies, by the pigeonhole principle, that there cannot be more than $m+1$ such m-bit vectors in U, or that $|U| \leq m+1$. ■

Armed with Theorem 33, we can now show that $\mathcal{R}(m,1) = m+1$.

Lemma 34 *For any $m \geq 0$ we have $\mathcal{R}(m,1) = m+1$.*

Proof. From Theorem 33 it follows that any canonical asynchronous priority arbitration scheme $\bar{A}(n,m,1) = \langle P, \bar{F}, \text{WIN} \rangle$ on n modules, m busses, and $t = 1$ stages, can reach at most $m+1$ distinct resolutions. For any such canonical asynchronous priority arbitration scheme, $\bar{A}$, there must be exactly n resolutions, which implies that $n \leq m+1$. Since this bound holds for arbitrary $\bar{A}$, we then also have $\mathcal{R}(m,1) \leq m+1$.

With $t = 1$, our generalized binomial arbitration scheme of Section 3.4.2 achieves $n = \sum_{l=0}^{t=1} \binom{m}{l} = \binom{m}{0} + \binom{m}{1} = 1 + m$. We therefore conclude that $\mathcal{R}(m,1) = m+1$. ■

We next generalize Theorem 33, by showing that any canonical asynchronous priority arbitration scheme with m busses and t stages can be in at most $\binom{m+1}{s} s!$ different configurations after $0 \leq s \leq t$ stages.

Theorem 35 *Let $\bar{A}(n,m,t) = \langle P, \bar{F}, \text{WIN} \rangle$ be a canonical asynchronous priority arbitration scheme on n modules, m busses, and t stages. Let $U[0] = \{0^m\}$ be the set of the initial configuration of m bits of 0, and let $U[s]$, for $1 \leq s \leq t$, be the set of possible configurations of $\bar{A}$ after s stages. Then, for any $0 \leq s \leq t$, we have $|U[s]| \leq \binom{m+1}{s} s!$.*

Proof. We prove the theorem by induction on s. For convenience of analysis, we first refine the definition of $U[s]$, for $0 \leq s \leq t$.

Due to the canonical nature of the asynchronous priority arbitration scheme $\bar{A}$, there are exactly n distinct arbitration processes to analyze, each corresponding to a different module c_i being the highest priority module arbitrating. We begin by defining the sequence of configurations that module c_i with arbitration priority p_i generates if c_i is the highest priority module that arbitrates. For any $0 \leq i \leq n-1$, we define $u_i[0] = 0^m$ and we inductively define $u_i[s] = \bar{F}(p_i, u_i[s-1])$, for values of $s \geq 1$. The canonical nature of the acyclic arbitration protocol $\bar{F}$ guarantees that the m-bit vector $u_i[s]$ is the configuration of the m busses after s stages, when module c_i is the highest priority module arbitrating, no matter which of the modules $c_0, c_1, \ldots, c_{i-1}$ also arbitrates. The set $U[s]$ of all possible configurations of $\bar{A}$ after s stages, for any $s \geq 0$, can now be defined as

$$U[s] = \bigcup_{i=0}^{n-1} \{u_i[s]\} .$$

This is the case because if module c_i is the highest priority module arbitrating, then the configuration of the m busses after s stages is $u_i[s]$.

We now prove the theorem by induction on s. For the case $s = 0$, we have $U[0] = \{0^m\}$ and $|U[0]| = 1 = \binom{m+1}{0}0!$. For $s = 1$, we have from Theorem 33 that $|U[1]| \leq m + 1 = \binom{m+1}{1}1!$. We now assume that for $s-1$ we have $|U[s-1]| \leq \binom{m+1}{s-1}(s-1)!$, and show that $|U[s]| \leq \binom{m+1}{s}s!$.

The set of possible configurations of the m busses after $s-1$ stages is $U[s-1]$. Each configuration $u \in U[s-1]$ defines an equivalence class, $C_u = \{c_i \ : \ u_i[s-1] = u\}$, of all the modules c_i that bring the busses to configuration u after $s-1$ stages. (Correspondingly, we define $P_u = \{p_i \ : \ u_i[s-1] = u\}$, for each $u \in U[s-1]$.) This definition implies that for any $u \in U[s-1]$, the configuration of the m busses after $s-1$ stages is u if and only if some module $c_i \in C_u$ is the highest priority module arbitrating. Furthermore, for each $u \in U[s-1]$ (or for each $c_i \in C_u$), the first $s-1$ busses $b_{m-1}, b_{m-2}, \ldots, b_{m-s+1}$ have stabilized on the first $s-1$ bits of u. The modules in C_u have only the last $m-s+1$ busses $b_{m-s}, b_{m-s-1}, \ldots, b_0$ to which they can apply new values at stage s. Focusing on the last $m-s+1$ busses, an argument similar to that of the proof of Theorem 33 shows that there are at most $m-s+2$ different responses of modules in C_u during stage s. Said formally, for any $u \in U[s-1]$ we have

$$\left| \bigcup_{p \in P_u} \{\bar{F}(p,u)\} \right| \leq m - s + 2 .$$

That is, any configuration $u \in U[s-1]$ can develop to no more than $m-s+2$ configurations during stage s. By definition, we have

$$U[s] = \bigcup_{u \in U[s-1]} \bigcup_{p \in P_u} \{\bar{F}(p,u)\} ,$$

which implies

$$\begin{aligned} |U[s]| &\leq |U[s-1]| \cdot (m-s+2) \\ &\leq \binom{m+1}{s-1}(s-1)! \cdot (m-s+2) \\ &= \frac{(m+1)!}{(m-s+2)!} \cdot (m-s+2) \\ &= \frac{(m+1)!}{(m-s+1)!} \\ &= \binom{m+1}{s} s! , \end{aligned}$$

which completes the proof of the theorem. ∎

Theorem 35 demonstrates that any canonical asynchronous priority arbitration scheme with m busses and t stages can be in no more than $\binom{m+1}{s}s!$ different configurations after s stages, for any $0 \leq s \leq t$. The result of Theorem 35 implies the following theorem.

Theorem 36 *For any $m \geq t \geq 0$, we have $\mathcal{R}(m,t) \leq \binom{m+1}{t}t!$.*

Proof. Let $\bar{A}(n,m,t) = \langle P, \bar{F}, \text{WIN} \rangle$ be a canonical asynchronous priority arbitration scheme on n modules, m busses, and t stages. From Theorem 35 we have that the number of possible configurations that $\bar{A}$ can be in after t stages in at most $\binom{m+1}{t}t!$. We then have $n \leq \binom{m+1}{t}t!$, because $\bar{A}$ has exactly n resolutions. Since this discussion holds for arbitrary $\bar{A}$, we conclude that $\mathcal{R}(m,t) \leq \binom{m+1}{t}t!$. ∎

The preceding analysis provides several nontrivial bounds for the bus-time tradeoff of general asynchronous priority arbitration schemes. These bounds were obtained by analyzing the canonical forms of such schemes. We conjecture, however, that the bounds of Theorem 35 and of Theorem 36 are not tight in general, and that the tight bound for the bus-time tradeoff is $\mathcal{R}(m,t) = \sum_{l=0}^{t} \binom{m}{l}$, exhibited by our generalized binomial arbitration scheme.

3.6 Discussion and extensions

This section contains some discussion, additional results, and directions for further research on priority arbitration with busses.

3.6.1 The k-ary arbitration scheme

The linear arbitration and binary arbitration schemes of Section 3.3 use n-ary and binary representations, respectively, of module priorities. We can also use radix-k representation of module priorities, for other values of k, to arbitrate among $n = k^t$ modules in t units of time, using $m = tk$ busses. We sketch the asynchronous k-ary arbitration scheme here due to its simplicity and because it generalizes the linear and binary arbitration schemes rather straightforwardly. This scheme exhibits a bus-time tradeoff of the form $m = tn^{1/t}$, which is a factor of e worse than the asymptotic bus-time tradeoff exhibited by our generalized binomial arbitration scheme of Section 3.4.2.

Asynchronous k-ary arbitration, for $2 \leq k \leq n$, can be described as follows. Each module is assigned a unique k-ary arbitration priority consisting of t radix-k digits. We divide the $m = tk$ busses into t disjoint groups, each consisting of k busses. During arbitration, competing module c applies the t radix-k digits of its arbitration priority p to the t groups of busses, using linear encoding of its digits on each group of k busses. As arbitration progresses, competing module c monitors the t groups of busses and disables its drivers according to the following rule: let $p^{(l)}$ be the lth radix-k digit of p and d_l be the highest index of a bus in the lth group of busses that carries a 1. Then if $p^{(l)} < d_l$, module c disables all its digits $p^{(j)}$ for $j < l$. Disabled digits are re-enabled should the condition cease to hold. Arbitration proceeds in t stages, each of which consists of resolving the value of another radix-k digit of the highest competing k-ary arbitration priority.

The asynchronous k-ary arbitration scheme combines the ideas of the asynchronous binary protocol with linear encoding of arbitration priorities, to achieve an intermediate bus-time tradeoff, $m = tn^{1/t}$. The acyclic arbitration protocol of k-ary arbitration is of size $m = tk$, but its depth is only $d = t$. The analysis of k-ary arbitration is a static one, similar to the analysis of binary arbitration. Implementing the asynchronous k-ary arbitration scheme, however, may require a different circuitry for arbitration in radix k. Our generalized binomial arbitration

scheme, besides achieving a better bus-time tradeoff, is also immediately implementable on any arbitration circuitry of binary arbitration, which is the most commonly used asynchronous priority arbitration scheme with busses.

3.6.2 Bus-time tradeoff of asynchronous priority arbitration

In Section 3.5.3, we proved that any asynchronous priority arbitration scheme on n modules, m busses, and t stages, satisfies $n \leq \binom{m+1}{t} t!$. Our generalized binomial arbitration scheme of Section 3.4.2 achieves a better bus-time tradeoff of the form $n = \sum_{l=0}^{t} \binom{m}{l}$. There is still a gap between the upper and the lower bounds on the bus-time tradeoff of asynchronous priority arbitration schemes. We conjecture that the bus-time tradeoff exhibited by the generalized binomial arbitration scheme is optimal for our model of asynchronous priority arbitration with busses, but we were unable to prove or disprove it. Using the notation of Section 3.5.3, we conjecture that $\mathcal{R}(m,t) = \sum_{l=0}^{t} \binom{m}{l}$, for any $m \geq 0$ and $t \geq 0$.

3.6.3 Synchronous priority arbitration schemes

In this chapter we discussed the asynchronous model of priority arbitration with busses and presented several asynchronous schemes. Considering synchronous priority arbitration scheme that use clocked arbitration logic, one can show that a synchronous version of k-ary arbitration achieves a bus-time tradeoff of the form $m = n^{1/t}$. (Variants of this scheme are used in synchronous communication protocols (see [45, 71]). In synchronous priority arbitration, busses can be reused on successive clock cycles, which enables a better bus-time tradeoff than that of asynchronous priority arbitration, in that there is no multiplicative factor of t in the bus-time tradeoff $m = n^{1/t}$.

For synchronous priority arbitration schemes, a related arbitration model can be defined. In this model it is possible to prove that the tradeoff $m = \Theta(n^{1/t})$ is optimal. The proof utilizes the result of Lemma 34 that with m busses at most $n = m + 1$ modules can be arbitrated in $t = 1$ stages. Using synchronous priority arbitration in t stages, one cannot do any better than arbitrating among at most $n = (m+1)^t$ modules, which implies the optimality of the tradeoff $m = \Theta(n^{1/t})$ exhibited by the synchronous version of k-ary arbitration.

3.6.4 Resource tradeoffs

Resource tradeoffs of the form $m = \Theta(tn^{1/t})$, based on multiway trees and the special class of binomial trees, are discussed in [8] for a variety of problems such as parallel sorting algorithms, searching algorithms, and VLSI layouts. Asynchronous priority arbitration with busses can in fact be considered as a selection process on trees. Asynchronous k-ary arbitration corresponds to a selection process on regular trees of branching factor k, while asynchronous generalized binomial arbitration corresponds to a selection process on the more economical "modified binomial trees" of [8].

3.6.5 Directions for further research

In this chapter we investigated a model for the settling of a digital bus that assumes a unit of time (bus-settling delay) for the bus to stabilize to a valid logic value. There are several situations, such as electrical transmission line, radio channels, and optical fibers, however, where a different analysis based on distances and directions may be required. In Chapter 4 we examine the performance of priority arbitration schemes in a more elaborate model of a bus as a digital transmission line.

The busses in the arbitration mechanisms investigated serve as a shared memory into which modules write and from which they read. These busses/memory implement the OR function of the values written to them. There might be some interest in other logic functions that busses/memory can implement. One interesting case would be memory cells that can compute the majority function on 0/1 values written into them.

Our work has concentrated on analyzing the data-dependent behavior of arbitration mechanisms that use fixed module priorities. There are several mechanisms that do not use deterministic module priorities or that arbitrate by using randomized protocols. It would be interesting to extend our analysis to these more flexible or randomized schemes.

Finally, the domain of data-dependent analysis has not been heavily investigated. There are many interesting circuits that exhibit faster performance than implied by the static measure of their depth. A more systematic approach for data-dependent analysis would prove to be a valuable tool for circuit designers. There has been some focus on the structure of delay-insensitive codes [85], for example, but not on data-dependent performance of logic circuits.

Chapter 4

Priority Arbitration on Digital Transmission Busses

This chapter examines the performance of priority arbitration schemes presented in Chapter 3 under the *digital transmission line* bus model. This bus model accounts for the propagation time of signals along bus lines and assumes that the propagating signals are always valid digital signals. A widely held misconception is that in the digital transmission line model the arbitration time of the binary arbitration scheme is at most 4 units of bus-propagation delay. We formally disprove this conjecture by demonstrating that the arbitration time of the binary arbitration scheme is heavily dependent on the arrangement of the arbitrating modules in the system. We provide a general scenario of module arrangement on m busses, for which binary arbitration takes at least $m/2$ units of bus-propagation delay to stabilize. We also prove that for general arrangements of modules on m busses, binary arbitration settles in at most $m/2+2$ units of bus-propagation delay, while binomial arbitration settles in at most $m/4+2$ units of bus-propagation delay, thereby demonstrating the superiority of binomial arbitration for general arrangements of modules under the digital transmission line model. For linear arrangements of modules in increasing order of priorities and equal spacings between modules, we show that 3 units of bus-propagation delay are necessary for binary arbitration to settle, and we sketch an argument that 3 units of bus-propagation delay are also asymptotically sufficient.

4.1 Introduction

The nature of signal propagation through a communication medium has a significant impact on the design of communication protocols for that medium. In any communication system, the time required for a signal sent by a given module to reach another module depends on the propagation speed of signals in the communication medium, the distance between the modules, and the directionality of signal propagation. Although different communication media may have different signal-propagation speeds, qualitatively they can be modeled in similar ways. Communication protocols must account for signal propagation delays by allowing enough time for information to disseminate through the system.

In this chapter we investigate the effects of signal propagation delays through bus lines on the performance of priority arbitration schemes presented in Chapter 3. For high-speed signals, a bus acts like an analog transmission line with associated impedance that affects the propagation delays (see [5, 22, 40, 88]). A complete characterization of signal propagation on analog transmission lines involves several transient effects such as reflections, superposition, and attenuation of signals. Analyzing the performance of communication protocols in such detailed analog models is a rather difficult task, however, and to make such analyses tractable a *digital transmission line* model for a bus is commonly used. This model accounts for the propagation delays of signals along a bus, assumes that the propagating signals are always valid digital signals, and ignores reflections, superposition, and attenuation of signals. The digital transmission line model is a model of an idealized digital bus, which ignores the delays caused by the analog nature of signals on electrical busses and focuses on the delays that arise from signal propagation along bus lines.

Several researchers studied the performance of the asynchronous binary arbitration scheme of Section 3.3.2 in the digital transmission line bus model. Taub [79, 81] investigated the maximal propagation delay of signals in the binary arbitration scheme, under the assumptions that modules are linearly arranged in increasing order of priorities and that they are equally spaced on the bus lines. Taub showed that in such situations 4 units of bus-propagation delay are sufficient for the binary arbitration scheme to settle, no matter how many bus lines are involved. However, Taub claimed that such an arrangement of system modules exhibits a worst-case scenario and concluded that 4 units of bus-propagation delay are always sufficient for the

binary arbitration scheme on any number of bus lines. Empirical counterexamples to Taub's claim were found [3, 87], which consist of arranging system modules in certain arrangements that require more than 4 units of bus-propagation delay for binary arbitration to settle. In [3], for instance, Ashcroft, Rivest, and Ward provide a specific example of arranging $n = 4$ modules on $m = 7$ bus lines, such that 5 units of bus-propagation delay are required for the binary arbitration scheme to stabilize. Other such empirical examples were found that contradict Taub's hypothesis for general cases. In this chapter, we identify the flaw in Taub's hypothesis, provide tight upper and lower bounds on the time (in units of bus-propagation delay) required by binary arbitration for general arrangements of modules, and reexamine linear arrangements of modules in increasing order of priorities.

In the remainder of this chapter, we investigate the binary arbitration scheme in the digital transmission line bus model. Section 4.2 discusses some issues of signal propagation on electrical transmission lines and describes the digital transmission line model of a bus. In Section 4.3, we formally disprove Taub's conjecture by providing a general scenario of module arrangement on m busses, for which binary arbitration takes at least $m/2$ units of bus-propagation delay. We also prove that for arbitrary arrangements of modules on m busses, binary arbitration settles in at most $m/2 + 2$ units of bus-propagation delay, while binomial arbitration from Chapter 3 settles in at most $m/4+2$ units of bus-propagation delay, thereby demonstrating the superiority of binomial arbitration for general module arrangements in the digital transmission line model. Section 4.4 examines linear arrangements of modules in increasing order of priorities and equal spacings between modules on the bus lines. In such arrangements, we show that 3 units of bus-propagation delay are necessary for binary arbitration to settle, and we sketch an argument that 3 units of bus-propagation delay are also asymptotically sufficient. Finally, in Section 4.5, we discuss the results of this chapter and indicate directions for further research.

4.2 Busses as transmission lines

In this section we discuss the transmission line nature of electrical busses. We first describe some of the analog issues of electrical bussed transmission lines, which affect the design of many bus systems and protocols. We then present the digital transmission line model of a bus, which serves as a low-level digital abstraction of a bus.

4.2.1 Analog issues of bussed transmission lines

The electrical transmission of signals on a bus line is an analog phenomenon, although the digital abstraction of logic design tries to hide the analog nature of signal transmission. The nature of signal transmission on a bus line includes the propagation speed of signals, reflections of signals, superposition of wave forms, voltage glitches and spikes, and signals attenuation, among others. Here we briefly discuss some of these phenomena.

The propagation of signals on a bussed transmission line is a time-consuming rather than an instantaneous event. The speed of signal propagation on a bus is determined by various physical and geometrical properties, such as the material, shape, temperature, and electrical properties of the bus in its environment. The length of a bus line determines the maximal duration that a signal needs to propagate through the bus, which is termed *bus-propagation delay*. However, there are other factors that affect the validity of digital signals that propagate on a bus line, thereby affecting the propagation speed of digital signals on the bus.

A bus has a characteristic impedance that depends on its geometrical and physical properties. This characteristic impedance is computed in terms of the inductance, capacity, and length of the bus (see [5, 40]). Impedance discontinuities along the bus, such as at connectors or at its ends, cause reflections of a fraction of each wave form passing through them. Reflected signals generate standing waves and noise on the bus line, which complicate the transfer of digital data. Signal reflections and termination can be considerably reduced by careful engineering of the bus and its connectors, but such fine tuning is rather complex and expensive.

A transmission line can simultaneously propagate numerous wave forms at different locations and in either direction. Different wave forms pass through each other without interference to create the spatial and temporal sum of the propagating wave forms. This phenomenon is known as the superposition principle. Superposition of valid digital signals may cause non-valid digital voltage levels at various places on a bus. The effect of superposition of signals is especially problematic with open-collector bus drivers, where several signals, applied by different modules, may be traveling on the bus in different directions. A discussion of wired-OR glitches, which result from superposition of signals on open-collector busses, appears in [42].

The number of modules connected to a bus line and the distances between modules play an important role in the propagation of signals on the bus. Electrical signals traveling on a

bus line experience some attenuation, which depends on the distance traveled and the driver's power. If several modules drive the bus to the same logic level, the bus may reach this level faster than if only one module drives the bus. In addition, the length of the bus and the number of modules on it determine the power at which modules should drive electrical signals onto the bus to guarantee that the signals driven are at valid digital levels.

As a consequence of all the analog complications in driving digital signals onto bus lines, most bus systems strive for engineering simplicity at the cost of reduced bus performance. In Chapter 3 we discussed a bus model that assumes that the voltage level on a bus may not be a valid digital value before a unit of bus-settling delay, T_{bus}, passes. In this chapter we introduce another bus model, the digital transmission line model, which attempts to capture the transient nature of traveling digital signals on a bus line and ignores the analog phenomena of signal reflections, waveform superposition, and voltage glitches and spikes. Very careful design and engineering of a bus can reduce much of the analog phenomena on transmission lines with the exception of the finite propagation speed of signals.

4.2.2 The digital transmission line bus model

The digital transmission line model accounts for propagation delays of digital signals along bus lines, which depend on the distances and the directions that signals travel. This model abstracts over the analog nature of reflected, superposed, and attenuated signals, by assuming that the propagating signals are always valid digital signals. The digital transmission line model is a model of an idealized bus, which enables examining certain inherent properties of bussed systems (see for example [3, 23, 79, 80, 81, 87]). A careful design of high-speed bus lines can result in a good approximation to this idealized model (see [5, 17, 81]).

In the digital transmission line bus model, we make the following assumptions. The system consists of n modules that are arranged along m parallel bus lines. The m bus lines all have the same length L. Each of the n modules is connected to all the m bus lines at the same spatial location, that is, at the same distance from the beginning of each bus line. Under these assumptions, the distance between two modules on the bus lines is well defined; it is the distance between the modules as measured on any of the m bus lines. There is a module at each of the two ends of the bus lines, such that the distance between the two furthest away modules is

exactly L and no other two modules are at distance L from each other.

Each module can drive digital signals on any of the m bus lines. All the m busses have the same signal propagation speed, which we denote by V. Signals driven on a bus line propagate at the same speed and in both directions on the bus. A signal that a module drives on any bus line, therefore, can not be noticed at distance d away from that module before time $t = d/V$ passes. The time it takes for a signal to travel the whole length L of a bus line is $T_p = L/V$ and is termed the *bus-propagation delay*. For simplicity, we assume that the signal propagation speed V is $V = 1$. This enables identifying a distance d on the bus lines and the time t that it takes for a signal to travel this distances d, since $t = d/V$. With this assumption we also have that $T_p = L$.

In the digital transmission line model, we assume that signals propagation on bus lines is a digital phenomenon that exhibits no analog behavior. There are no reflections of signals or of fractions of signals anywhere on the bus lines. The bus lines are terminated properly and signals reaching either end of a bus line simply disappear. Digital signals that meet on a bus line superpose in a logic OR manner according to the wired-OR nature of the bus medium, that is, at any given point on a bus line the resultant level measured is always the logic OR of the digital signals passing there. No signal spikes, glitches, or attenuation are experienced; signals are always at valid digital levels. Signals on parallel bus lines do not interfere with each other, that is, there is no "cross talk" between bus lines. Finally, we assume that modules do not experience any gate delays in driving signals on bus lines; the only delays considered are the propagation delays of digital signals along bus lines. In spite of its abstract characterization of bus lines, the digital transmission line model is a useful tool for investigating the effects of signal propagation delays on the performance of various protocols.

4.3 General arrangements of modules

In this section, we investigate the arbitration time of the binary arbitration scheme for general arrangements of modules. A widely held misconception is that in the digital transmission line model the arbitration time of binary arbitration is at most 4 units of bus-propagation delay. Here, we formally disprove this conjecture by demonstrating that the arbitration time of the binary arbitration scheme depends on the arrangement of the arbitrating modules in

the system. We first provide a scenario of module arrangement on m busses, for which binary arbitration takes at least $m/2$ units of bus-propagation delay to settle. We then prove that for any arrangement of modules on m busses, binary arbitration stabilizes after at most $m/2 + 2$ units of bus-propagation delay. Finally, we relate these results to the binomial arbitration scheme and demonstrate that it settles in at most $m/4 + 2$ units of bus-propagation delay.

4.3.1 Lower bound for binary arbitration

To prove the lower bound on the arbitration time of binary arbitration with m bus lines in the digital transmission line model, we describe a scenario for arranging a selected set of arbitrating modules on the m bus lines. We assume that all the arbitrating modules start their arbitration process simultaneously and follow the binary arbitration protocol, which is described in Section 3.3.2. We remind that this protocol states that each module applies its arbitration priority to the m bus lines, and that if a module applies a logic 0 to a certain bus line but detects that the bus line carries a logic 1, then the module disables all its bits of lower significance for as long as the conflict on that bus line remains. This rule guarantees that after sufficient delay only the bits of the highest arbitration priority are applied to the m bus lines. Until this time delay passes, however, there may be many modules applying and disabling low-order bits, which may generate many transient digital signals on the bus lines. The system stabilizes when all the transient signals on all the bus lines have disappeared. Our lower bound scenario arranges selected modules on the m bus lines in such a way that there is a sequence of $m/2$ transient signals, each of which is stimulated by its predecessor in the sequence, that travel from side to side on the m bus lines. This has the effect of delaying system settlement until at least $m/2$ units of bus-propagation delay pass.

Our lower bound scenario partitions the selected arbitrating modules into two sets, which we shall denote by A and B. The set A of modules is located at the very far right end of the m bus lines and the set B of modules is located at the very far left end. The distances between modules inside each set are very small compared to the distance between the two sets. The distance between the two sets (between the leftmost module in the right set A and the rightmost module in the left set B) is almost the whole length L of the bus system. This has the effect that arbitration inside each of the two sets settles much faster than even the time

required for a signal to propagate from one set to the other. (These distances and delays will be discussed in more detail towards the end of this subsection.) Figure 4-1 illustrates this high level partitioning of the selected arbitrating modules into sets A and B.

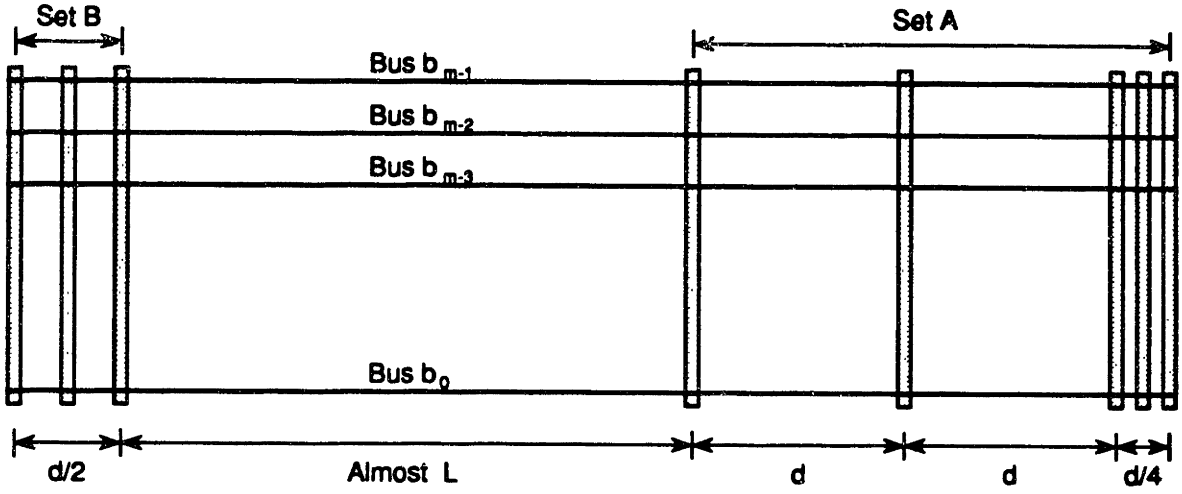

Figure 4-1: High level partitioning of the selected arbitrating modules into sets A and B. With a parameter d (to be determined later), the total length of set A is $9d/4$, the total length of set B is $d/2$, and the distance between the two sets is almost L, such that $d \ll L$.

Inside each of the sets A and B, modules are organized in linear order of priorities, with priorities increasing from left to right in set A and from right to left in set B. Each set by itself settles rather fast, due to its relatively short total length. However, the arbitration priorities in the two sets are selected in such a way that they interact with each other. Initially, when arbitration begins, a special "wave form" is generated by modules in set A on 2 top bus lines and is propagated towards set B. This special "wave form" arrives at set B after the arbitration in set B have already settled and causes some temporary confusion there. As a result, a similar, reflected, and shrunk-by-2 "wave form" is generated by modules in set B on the next 2 bus lines and is propagated back towards set A, where it causes a similar temporary confusion. This, in turn, results in a similar, reflected, and again shrunk-by-2 "wave form", which is generated by modules in set A and is now propagated back towards set B on the next 2 bus lines. This ping-pong of "wave forms" lasts for $m/2$ iterations, since each such iteration utilizes 2 distinct bus lines. The duration of each such iteration is almost T_p, since this is the time required by any "wave form" to propagate from set A to set B or vice versa. The arbitration process of the whole system is therefore not completed before $(m/2)T_p$ time passes.

We now describe the "ping-pong wave forms" that propagate back and forth between the sets A and B. Each "ping-pong wave form" is the combination of two signals traveling together in the same speed and direction on two consecutive bus lines. Odd-indexed "ping-pong wave forms" are generated by modules in set A and propagate towards set B (from right to left), while even-indexed "ping-pong wave forms" are generated by modules in set B and propagate towards set A (from left to right). The first "ping-pong wave form" is spontaneously generated by set A when arbitration begins. The ith "ping-pong wave form", for $1 < i \leq m/2$, is generated as a result of receiving the $(i-1)$st "ping-pong wave form". In general, the ith "ping-pong wave form", for $1 \leq i \leq m/2$, can be described as follows:

- a 1-signal of duration $2d/2^i$ on bus line b_{m-2i}, and
- a 0-signal of duration $4d/2^i$ on bus line b_{m-2i-1}.

Figure 4-2 illustrates the ith "ping-pong wave form". The parameter d is the distance between the modules generating the first "ping-pong wave form" and will be discussed later.

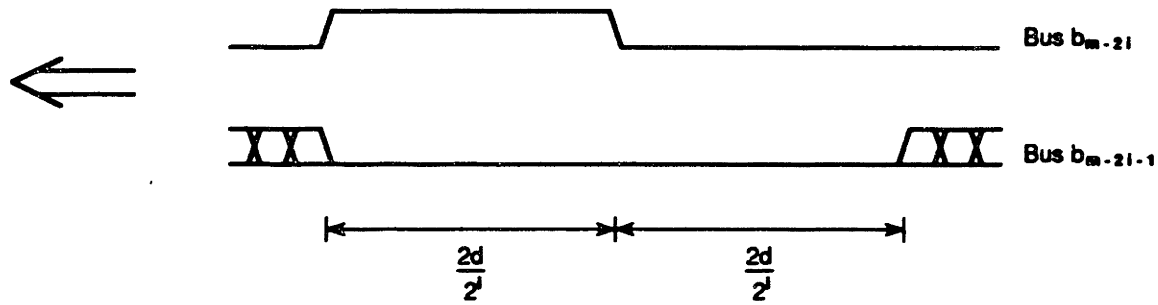

Figure 4-2: The ith "ping-pong wave form" on two consecutive bus lines. This wave form propagates from right to left, that is, i is assumed to be odd. For even i, this wave form should be reflected.

We now turn to describe the relative arrangement of modules inside the sets A and B, which is responsible for the "ping-pong wave forms" phenomenon. For simplicity, we focus first on the structure of set B, which is somewhat simpler than that of set A. The location of modules in set B and their relative distances from each other are of primary importance. The modules in set B are responsible for receiving the odd-indexed "ping-pong wave forms" (first, third, etc.) coming from the right, and for generating the even-indexed "ping-pong wave forms" (second, fourth, etc.) that propagate to the right. To examine the generation of the second "ping-pong wave form", for example, we need to describe the location and priorities of three modules in set

B. These three modules with arbitration priorities p_1, p_2, and p_3 are illustrated in Figure 4-3. Module p_3 is at the left end of the bus system, module p_2 is at distance $d/4$ from the left end of the bus system, and module p_1 is at distance $d/2$ from the left end of the bus system. (The parameter d, to be reminded, is related to the duration of the first "ping-pong wave form".) Furthermore, module p_1 is the only arbitrating module (in both sets A and B) with a 1 on bus b_{m-4}; no other arbitrating module has a 1-bit on this bus. The space between modules p_1 and p_2 contains no other arbitrating modules. The space between modules p_2 and p_3 may contain other arbitrating modules for generation of future even-indexed "ping-pong wave forms". However, each arbitrating module in the space between p_2 and p_3 must agree with p_2 and p_3 on their high order bits, as illustrated in Figure 4-3.

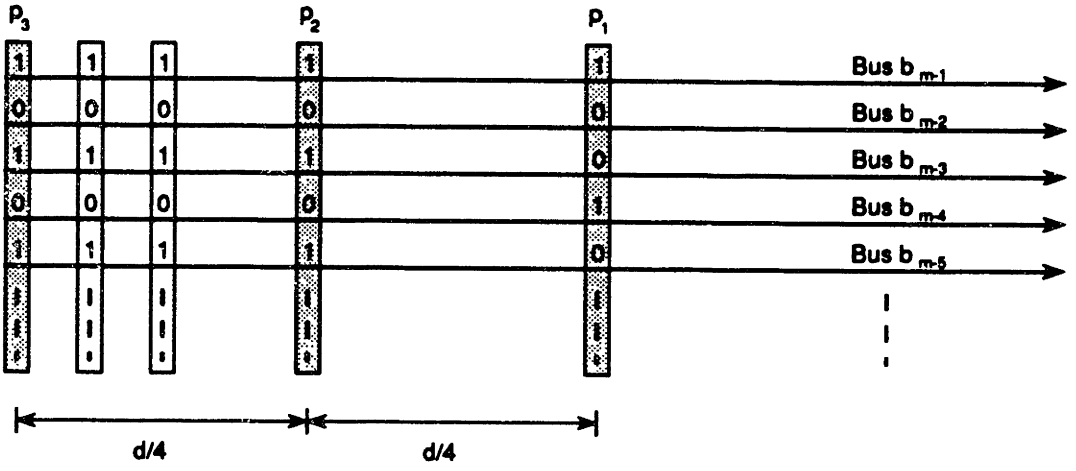

Figure 4-3: The arrangement of the three modules in set B that are responsible for receiving the first "ping-pong wave form" and for generating the second "ping-pong wave form". The space between p_1 and p_2 contains no other arbitrating modules. The space between p_2 and p_3 may contain other arbitrating modules for generation of future even-indexed "ping-pong wave forms".

We next examine how the arrangement of the three set-B modules, illustrated in Figure 4-3, receives the first "ping-pong wave form" and generates the second "ping-pong wave form". We assume that the first "ping-pong wave form", which propagates from right to left, arrives at the location of module p_1 at time t. This first "ping-pong wave form" consists of 2 left-traveling signals as follows: a 1-signal of duration d on bus line b_{m-2} accompanied by a 0-signal of duration $2d$ on bus line b_{m-3} (see Figure 4-2). In the following discussion, we keep track of right-traveling wave forms generated on bus lines b_{m-4} and b_{m-5}, as detected at the location of module p_1, starting at time $t + d$.

We first concentrate on the wave form generated on bus line b_{m-4} at the location of p_1 after time $t + d$. Notice that the left-traveling 1-signal on bus line b_{m-2} (one part of the first "ping-pong wave form") arrives at module p_1 at time t, is of duration d, and thus it leaves module p_1 at time $t + d$. At time $t + d/2$ the leading edge of this signal arrives at module p_3, and at time $t + 5d/4$ the trailing edge of this signal leaves module p_2 (see Figure 4-3). Therefore, in the time interval $(t + d/2, t + 5d/4)$, all modules between p_2 and p_3 disable their bits on the bus lines below b_{m-2}. Specifically, this causes a right-traveling 0-signal on bus line b_{m-3}, originated at module p_3, which arrives at module p_1 at time $t + d$. This right-traveling 0-signal on bus line b_{m-3} is terminated at time $t + 5d/4$ at the location of module p_2, since the signal on bus b_{m-2} passes p_2 at that time. However, at the location of p_1, the right-traveling 0-signal on bus b_{m-3} is detected until time $t + 5d/4 + d/4 = t + 3d/2$ (it takes time $d/4$ for the change at p_2 to reach p_1). In addition, the left-traveling 0-signal on bus line b_{m-3} (the other part of the first "ping-pong wave form") guarantees that no 1-signal arrives on this bus from the right until time $t + 2d$. The result of all the above discussion is that between time $t + d$ and time $t + 3d/2$ the digital values on bus lines b_{m-1} through b_{m-3} at the location of p_1 agree with the bits of priority p_1. Consequently, module p_1 generates a 1-signal on bus line b_{m-4} in the time interval $(t + d, t + 3d/2)$, which propagates both left and right and is of duration $d/2$. The right-traveling portion of this signal is one part of the second "ping-pong wave form".

We now concentrate on the wave form generated on bus line b_{m-5} at the location of p_1 after time $t + d$. The discussion in the previous paragraph about the right-traveling 0-signal on bus line b_{m-3} is also applicable to bus line b_{m-5}, since the modules between p_2 and p_3 disable *all* their bits below bus b_{m-2}. Therefore, there is a right-traveling 0-signal on bus b_{m-5} between time $t + d$ and time $t + 3d/2$. However, the 1-signal on bus b_{m-4}, generated by module p_1 between time $t + d$ and $t + 3d/2$, propagates both to the left and to the right. The left-traveling portion of this 1-signal on bus b_{m-4} arrives at modules to the left of p_1 just as the left-traveling 1-signal on bus b_{m-2} leaves those modules. Consequently, modules to the left of p_1 continue to disable their bits on bus b_{m-5} for at least another $d/2$ time, which is the duration of the 1-signal that module p_1 generates. As a result, we have a right-traveling 0-signal on bus line b_{m-5} in the time interval $(t + d, t + 2d)$, which is the other part of the second "ping-pong wave form". The right-traveling signals on bus lines b_{m-4} and b_{m-5} leave set B at time $t + d$ on their

way towards set A, where a similar, reflected, and shrunk-by-2 process occurs.

The structure of set A is almost identical to that of set B. The only difference is that the first "ping-pong wave form" is spontaneously created by modules in set A when the arbitration process begins. Figure 4-4 illustrates the five set-A modules that are responsible for the first and the third "ping-pong wave forms". Modules p_4, p_5, and p_6 spontaneously create the first "ping-pong wave form" on bus lines b_{m-2} and b_{m-3}. To see that, we concentrate on the left-propagating wave forms detected at the location of module p_4 immediately after arbitration starts. When arbitration begins, module p_4 generates a 1-signal on bus line b_{m-2} for a duration of d, since after time d the 1-signal that module p_5 generates on bus line b_{m-1} disables module p_4 forever. Also, when arbitration begins, bus line b_{m-3} at the location of p_4 carries a 0-signal for a duration of $2d$, until the 1-signal from module p_6 arrives from the right to the location of module p_4. The combination of the signals on bus lines b_{m-2} and b_{m-3} is the first "ping-pong wave form" that propagates towards set B. Modules p_6, p_7, and p_8 are responsible for the third "ping-pong wave form" on bus lines b_{m-6} and b_{m-7}. The arrangement of these modules is a shrunk-by-2 mirror image of the arrangement of set B.

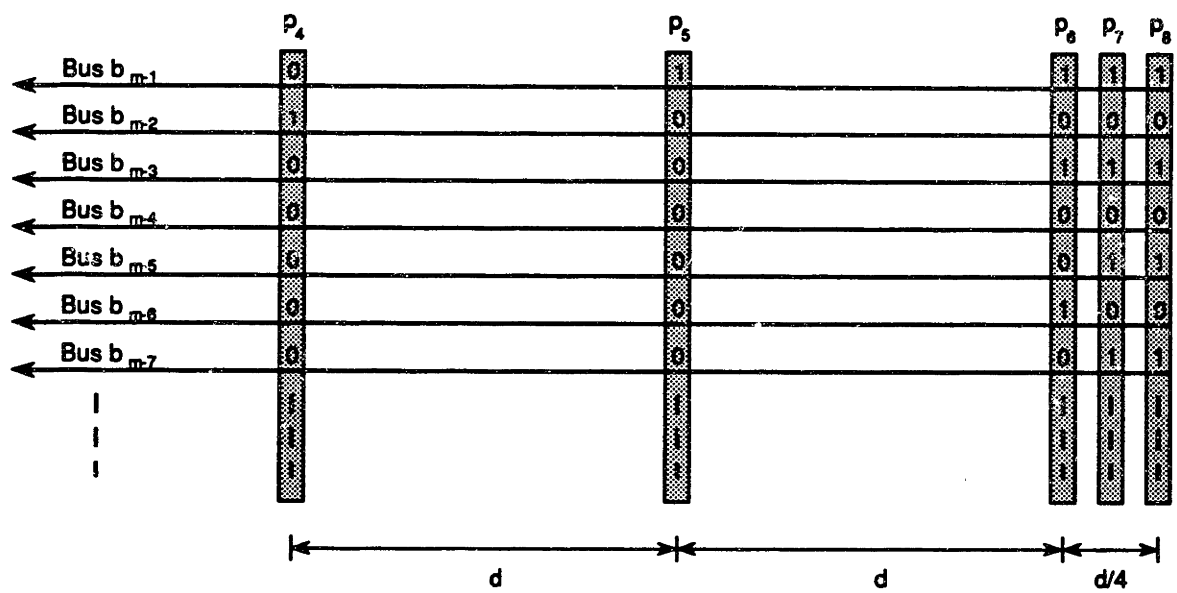

Figure 4-4: The arrangement of the five modules in set A that are responsible for creating the first "ping-pong wave form", and for receiving the second "ping-pong wave form" and generating the third "ping-pong wave form". The spaces between p_4 and p_5, between p_5 and p_6, and between p_6 and p_7 contain no other arbitrating modules. The space between p_7 and p_8 may contain other arbitrating modules for generation of future odd-indexed "ping-pong wave forms".

The scenario for module priorities and placement continues in a recursive fashion. For example, the region in set B, which is responsible for the fourth "ping-pong wave form", is a shrunk-by-4 image of the modules in Figure 4-3. The three new modules are placed in total space of $d/8$ from the left end of the bus lines, with the leftmost module of the three coinciding with the module already there. The leftmost module on the bus lines, thus, has the following string $(10)^{m/2}$ as its arbitration priority. Formally, for the generation of the $(2k)$th "ping-pong wave form", we place a module with priority $(10)^{2k-1}1010^{m-4k-1}$ at distance $d/2^{2k}$ from the left end, and another module with priority $(10)^{2k-1}010^{m-4k}$ at distance $2d/2^{2k}$ from the left end. Similar recursion is applied to the structure of the right set A.

We now discuss the design parameters d, L, T_p, and the duration of the arbitration process. The parameter d is the spacing between the modules that generate the first "ping-pong wave form", and the parameter L is the length of a bus line. The total length occupied by the two sets A and B combined is no more than $3d$, which leaves a distance of at least $L - 3d$ between the two sets for "ping-pong wave forms" to travel back and forth. The arbitration scenario, thus, consists of $m/2$ iterations, each of which takes at least $(L-3d)/L$ units of bus-propagation delay. To maximize the arbitration time, we need to minimize the value of d. If the system design is such that there is no lower limit on the distance between modules, then d could be made as small as desirable and the arbitration process would take $m/2$ units of T_p. If, however, modules are required to be equally spaced on the bus lines, then the following analysis shows that the lower bound of $m/2$ units of T_p is asymptotically attainable.

Suppose that Δ is the spacing between any two consecutive modules on the bus lines. To enable $m/2$ iterations of the lower bound scenario, the duration of the last "ping-pong wave form" must be at least Δ. Alternatively, we must have $\Delta = 2^{-(m/2-1)}d$, or $d = 2^{m/2-1}\Delta$. However, on m bus lines there are 2^m modules, which implies $L = (2^m - 1)\Delta$. The ratio $(L - 3d)/L$ is then at least $1 - 1/(2^{m/2-2})$, which approaches 1 as m increases. This indicates that asymptotically almost the full length of the bus lines is traveled in each iteration. We summarize this discussion in the following theorem.

Theorem 37 *There is a scenario of module arrangement on m bus lines, such that under the digital transmission line bus model, the binary arbitration scheme asymptotically requires at least $m/2$ units of bus-propagation delay to settle.*

4.3.2 Upper bound for binary arbitration

In this subsection, we prove that for any arbitrary arrangement of modules on m busses, binary arbitration stabilizes after at most $m/2 + 2$ units of bus-propagation delay. This upper bound is derived by concentrating on the number of 0-intervals in the highest competing arbitration priority and on the relative locations of arbitrating modules. We first define the number of 0-intervals of a codeword.

Definition 16 The number of 0-intervals of a binary codeword p is the number of intervals of consecutive 0's that p contains, disregarding the leading 0's.

The nature of the binary arbitration protocol is such that an interval of consecutive same bits of a codeword can be regarded as a basic unit. For an interval of consecutive 1's this is the case, since such interval cannot be interrupted in the middle (there is no 0-bit there where 1-signals can penetrate). An interval of consecutive 1's it thus either applied as one unit or entirely disabled. An interval of consecutive 0's can be interrupted in the middle, but then it has the effect of disabling all the bits below that interval, no matter where inside the interval the interruption occurs. In a binary arbitration process, the number r of 0-intervals of the highest competing priority is related to the arbitration time, as the next theorem implies. The theorem also relates the arbitration time to L, the length of the bus lines. This connection is rather important, as the proof relies on the fact that arbitration among modules that are close on the bus lines terminates faster than among far away modules.

Theorem 38 *Consider a binary arbitration process on m bus lines of length L under the digital transmission line bus model. Let Q be the set of arbitrating priorities, p be the highest priority in Q, and r be the number of 0-intervals of p. Then the arbitration process settles after at most $(r+2)L$ time, that is, there are no more transient signals on any bus line after time $t = (r+2)L$.*

Proof. Since the number of 0-intervals of the highest competing arbitration priority p is r, then p is of the form $p = 0^{k_0}1^{l_1}0^{k_1}1^{l_2}0^{k_2}\cdots 1^{l_r}0^{k_r}1^{l_{r+1}}$, where $k_0 \geq 0$; $l_j, k_j > 0$ for $1 \leq j \leq r$; $l_{r+1} \geq 0$; and $k_0 + l_{r+1} + \sum_{j=0}^{r}(l_j + k_j) = m$. In the following discussion, we ignore the k_0 leading 0's since the first k_0 bus lines carry 0's throughout the arbitration process. For notation simplicity, we then assume that $k_0 = 0$. We now prove the theorem by induction on r for arbitrary values of L.

Base case: $r = 0$. The codeword p consists of m consecutive 1's, that is, $p = 1^m$. This interval of m consecutive 1's propagates together on the m bus lines, and after at most one unit of bus-propagation delay all bus lines have settled forever. Arbitration in this case takes no more than L time, which does not exceed $(r+2)L$ time.

Base case: $r = 1$. The codeword p has the form $p = 1^{l_1}0^{k_1}1^{l_2}$. The first interval of l_1 consecutive 1's propagates together on the first l_1 bus lines, and after at most one unit of bus-propagation delay all these l_1 bus lines settle to 1's forever. As a result, any module that has some 1-bits in the second interval of k_1 bus lines, disables these bits after at most one unit of bus-propagation delay. Therefore, after at most two units of bus-propagation delay, the second interval of k_1 bus lines settles to 0's forever. Consequently, after at most two units of bus-propagation delay, module p re-enables its last interval of l_2 consecutive 1's forever, which brings the bus lines to stable state after at most three units of bus-propagation delay. (See Section 4.4.1 for a proof that this scenario is indeed possible.) Arbitration in this case takes no more than $3L$ time, which does not exceed $(r+2)L$ time.

Inductive case: $r > 1$. The codeword p has the form $p = 1^{l_1}0^{k_1}1^{l_2}0^{k_2}\cdots 1^{l_{r+1}}$. We define the set $\bar{Q}$ of all arbitrating modules in Q that have their first $l_1 + k_1$ bits identical to those of p, that is, $\bar{Q} = \left\{q \in Q \ : \ q = 1^{l_1}0^{k_1}\cdots\right\}$. We focus on possible 1-signals sent by other arbitrating modules (from $Q - \bar{Q}$) in the second interval of p, that is, the interval of k_1 consecutive 0's. There are three cases to consider:

(a) There are no arbitrating modules with 1-bits in the second interval of p. In this case the first three intervals of p, which have the form $1^{l_1}0^{k_1}1^{l_2}$, behave like one uninterrupted interval that could be replaced by an interval of $l_1 + k_1 + l_2$ consecutive 1's with no change in the behavior. The number of 0-intervals remained to be considered is now $r - 1$. By induction, such arbitration processes take at most $((r-1)+2)L < (r+2)L$.

(b) There is an arbitrating module $q \in Q - \bar{Q}$ with a 1-bit in the second interval of p, and there is another arbitrating module $p' \in \bar{Q}$, such that q is physically between p and p' on the bus lines (see Figure 4-5). Let d_1 be the distance of q from p and let d_2 be the distance of q from p'. Without loss of generality, we assume that $d_1 < d_2$. (This also implies that $d_1 < L/2$, since otherwise $d_1 + d_2 > L$). Then the 1-signal that module q generates in the 0-interval of p has duration d_1 (it is disabled by module p after time d_1). This 1-signal

completely disappears from the system after at most one unit of bus-propagation delay, since both its right-traveling and left-traveling portions go over the corresponding ends of the bus lines after at most time L. Consequently, not only is module q disabled after one unit of bus-propagation delay, but also the effects that it caused in the system are gone and the modules of $\bar{Q}$ are re-enabled after that time. By induction now, the modules of $\bar{Q}$ complete the arbitration after at most $((r-1)+2)L$ time, which with the extra unit of time L for disabling modules like q give an arbitration time of at most $(r+2)L$.

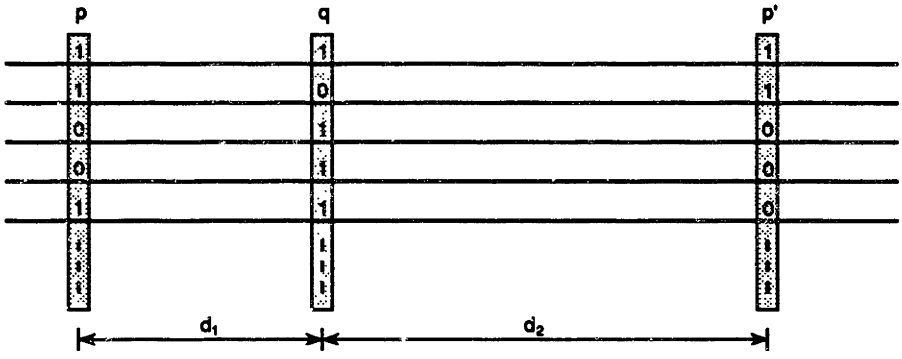

Figure 4-5: An interrupting module q on the first 0-interval of the highest arbitration priority p. There is another module p' with the same first two intervals as p on the other side of q.

(c) There is an arbitrating module $q \in Q - \bar{Q}$ with 1-bits in the second interval of p, and all the modules in $\bar{Q}$ are on the same side of q (see Figure 4-6). Let p' be the module in $\bar{Q}$ that is closest to q and let d be the distance between p' and q. The 1-signal that module q generates in the 0-interval of p has duration d, since it is disabled by module p' after time d. However, this 1-signal may take another time L to completely disappear from the system, since it may be the case that module q is at the very end of the bus lines. Therefore, after at most $d+L$ time the effects that modules like q cause are gone and the modules of $\bar{Q}$ are re-enabled after that time. Notice, however, that the modules of $\bar{Q}$ have cleared the first 0-interval of p, so that there are $r-1$ more 0-intervals of p to consider. In addition, notice that the the modules of $\bar{Q}$ are located in a bus-region of length at most $L-d$. By induction now, the modules of $\bar{Q}$, on the reduced region of the bus lines, complete the arbitration after at most $((r-1)+2)(L-d) = rL - rd + L - d$ time. To this time we need to add the extra $d+L$ time required to eliminate modules like q. Finally, we add another d units of time to allow the final signals of p to propagate

beyond the region of length $L - d$ and to cover the full length of the bus lines. The total time is, therefore, $(rL - rd + L - d) + (d + L) + d = rL - rd + 2L = (r+2)L - rd \leq (r+2)L$, as required.

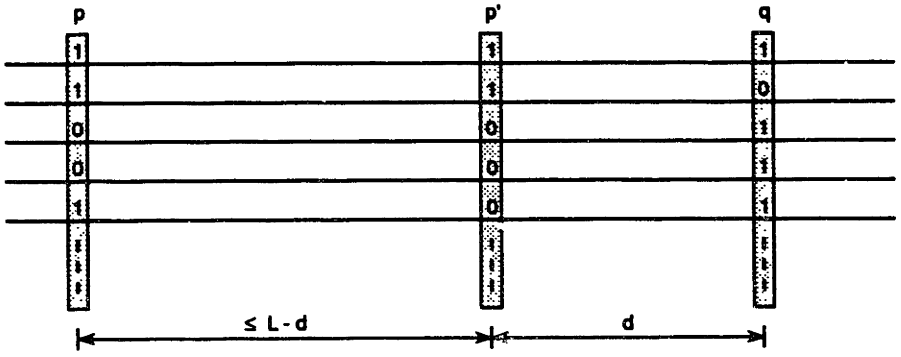

Figure 4-6: An interrupting module q on the first 0-interval of the highest arbitration priority p. All the arbitrating modules with the same first two intervals as p are on the same side of q.

We conclude that any binary arbitration process on bus lines of length L, with p, the highest competing arbitration priority, having r 0-intervals, completes after $(r+2)L$ time. ■

Theorem 38 bounds the arbitration time of any binary arbitration process by $(r+2)L$, where r is the number of 0-intervals in the highest arbitrating priority and L is the length of the bus lines. With m bus lines to arbitrate, the number of 0-intervals of any arbitration priority is no more than $m/2$. In addition, we assume that $L = T_p$, where T_p is the bus-propagation delay. These observations imply the following corollary.

Corollary 39 *For any binary arbitration process on m bus lines under the digital transmission line bus model, arbitration settles in at most $m/2 + 2$ units of bus-propagation delay.*

4.3.3 Lower and upper bounds for binomial arbitration

Binomial arbitration uses the same arbitration protocol as binary arbitration. The results of the preceding subsections, which provided lower and upper bounds on the arbitration time of binary arbitration, are, therefore, directly applicable to binomial arbitration as well.

A lower bound scenario, similar to that of Theorem 37, can be applied to the binomial arbitration scheme. The only difference is that the binomial arbitration priorities have no

more than $m/4$ 0-intervals, where "ping-pong wave forms" can penetrate and cause temporary confusion. This implies the following corollary.

Corollary 40 *There is a scenario of module arrangement on m bus lines, such that under the digital transmission line bus model, the binomial arbitration scheme asymptotically requires at least $m/4$ units of bus-propagation delay to settle.*

The upper bound for binomial arbitration is derivable from Theorem 38. Since for binomial arbitration on m bus lines any arbitration priority has at most $m/2$ intervals, the number of 0-intervals in any priority is no more than $m/4$. This implies the following corollary.

Corollary 41 *For any binomial arbitration process on m bus lines under the digital transmission line bus model, arbitration settles in at most $m/4 + 2$ units of bus-propagation delay.*

4.4 Linear arrangements of modules

In this section we examine linear arrangements of modules in increasing order of priorities with the modules equally spaced on the bus lines. For such arrangements, we show that 3 units of bus-propagation delay are necessary for binary arbitration to settle. We also sketch an argument that indicates that 3 units of bus-propagation delay, rather than the 4 claimed in [79, 81], are asymptotically sufficient for binary arbitration.

4.4.1 Lower bound for binary arbitration

To demonstrate a lower bound of 3 units of bus-propagation delay on the arbitration time of binary arbitration, we present an arrangement of two modules as in Figure 4-7. The arbitration priority p of the module on the left side is $1^{m-2}01$ and the arbitration priority q of the module on the right side is $0^{m-2}10$. We use d to denote the distance between modules p and q. When arbitration begins module q sends its 1-bit towards module p during the time interval $(0, d)$, since after time d the high order bits of p disable the 1-bit of q. At the location of p, the 1-signal on bus b_1 is detected during the time interval $(d, 2d)$, which causes p to disable its last bit of 1 during that time interval. Only after time $2d$, module p re-enables its last bit of 1, but it takes slightly more than time d for this change to propagate throughout the system.

The arbitration time, therefore, is at least $3d$. The reader may verify that if Δ is the distance between consecutive modules on the bus lines, then the distance between modules p and q in Figure 4-7 is $d = (2^m - 5)\Delta$. The total length of the bus lines is $L = (2^m - 1)\Delta$, and thus the ratio d/L asymptotically approaches 1. This shows that the arbitration time of $3d$ approaches 3 units if bus-propagation delay asymptotically, as m increases.

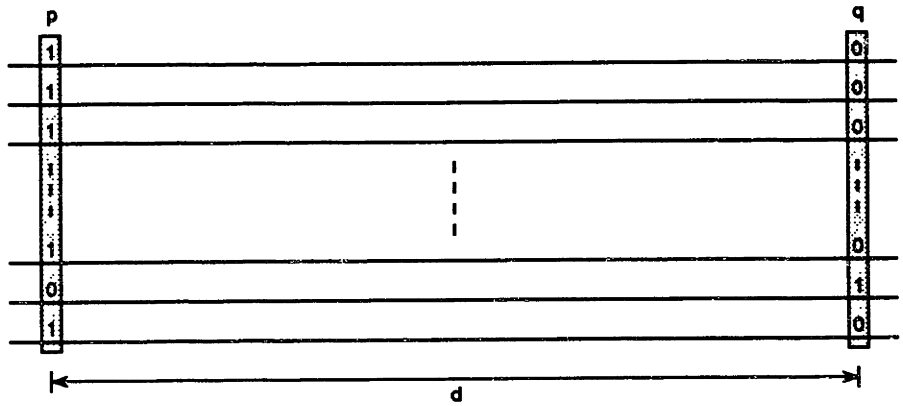

Figure 4-7: Linear arrangement of 2 modules very close to the two ends of the bus lines. The arbitration process on this arrangement asymptotically takes 3 units of bus-propagation delay.

4.4.2 Upper bound for binary arbitration

We now sketch an argument that indicates that the arbitration time of binary arbitration can be shown to be close to 3 units of bus-propagation delay. The argument involves inspection of several cases and only a high-level description of it is presented here. With m bus lines there are 2^m modules and the total length of the bus system is L. We partition the modules into 2^k subregions, each of length $L/2^k$, according to the first k bits of their arbitration priorities. By inspecting each of the 2^k subregions, one can verify that if the highest priority is in a given subregion, then after at most 2 units of bus-propagation delay all the possible transient signals sent by modules in lower-priority subregions have disappeared. This leaves only the subregion under inspection, whose length is $L/2^k$, for the rest of the arbitration, which we shall analyze recursively. In addition, after the arbitration is completed on the inspected subregion of length $L/2^k$, at most another $1 - 1/2^k$ units of bus-propagation delay are required for the bit-signals of the highest priority to spread throughout the bus lines. If we let $T(n)$ denote the maximal time required by binary arbitration on n modules, then we get the following recurrence:

$T(n) = 2+T(n/2^k)+(1-1/2^k)$, which solves to give $T(n) = 3+2/(2^k-1)$. Now, as k increases (there are more cases to inspect), the maximal arbitration time can be shown to asymptotically approach 3 units of bus-propagation delay.

4.5 Discussion and extensions

In this section, we discuss the results of this chapter and indicate directions for further research.

4.5.1 Discussion

In this chapter, we investigated how the finite propagation speed of signals on bussed transmission lines affects the performance of the priority arbitration schemes of Chapter 3. We formally disproved Taub's conjecture by providing a general scenario of module arrangement on m busses, for which binary arbitration takes at least $m/2$ units of bus-propagation delay. We also proved that for any arrangement of modules on m busses, binary arbitration settles in at most $m/2+2$ units of bus-propagation delay, while binomial arbitration settles in at most $m/4+2$ units of bus-propagation delay. This demonstrates the superiority of binomial arbitration for general arrangements of modules under the digital transmission line model. For linear arrangements of modules in increasing order of priorities and equal spacings between modules, we showed that 3 units of bus-propagation delay are necessary for binary arbitration to settle, and we indicated that 3 units of bus-propagation delay are also asymptotically sufficient. System designers and engineers may wish to reconsider the use of Taub's assumptions and analyses, since different arrangements of system modules exhibit substantially different behavior.

4.5.2 Further research

Several directions for extending the results of this chapter are listed.

- Average-case arbitration time of binary arbitration for arbitrary and linear arrangements.
- Linear arrangements of modules with arbitrary spacings between modules.
- The performance of binomial arbitration for linear arrangements of modules.
- Models of bussed transmission lines that characterize other aspects of the media.

Chapter 5

Conclusion

Bussed interconnections are extensively used in many digital systems. Investigating the characteristics, capabilities, and organization of bussed systems are the subject of ongoing research. In this thesis, we focused on two application domains for busses: communication architectures and control mechanisms, and examined the capabilities of busses as interconnection media, computation devices, and transmission channels. This chapter presents some concluding remarks and motivates further research on bussed interconnections, in general, and on each of the aspects of bussed systems that this thesis explored, in particular.

5.1 Bussed interconnections

Busses are shared communication media. A single bus can only implement one communication transaction at any given time and thus constitutes a scarce resource that must be utilized intelligently. Much research is directed at investigating techniques and mechanisms that can enrich the bandwidth of a bus. Several techniques, such as time multiplexing, frequency multiplexing, spatial multiplexing, and angular multiplexing have been suggested for some communication media, such as radio channels and optical communications (see [12, 13, 78]). Some of these techniques have also been applied to electrical busses, but a more thorough exploration of bus multiplexing techniques is required.

Busses enable communication among several system modules, in contrast with point-to-point wires that establish communication only between pairs of modules. This property of busses

may or may not be desirable, depending on the application. On busses, any communication transaction, whether a one-to-one or a broadcast, can be detected by all system modules, while point-to-point wires feature privacy of communication. Busses require sophisticated controlling mechanisms and protocols to enable sharing and to support sequencing of transactions, while controlling the communication with point-to-point wires is somewhat more straightforward. Busses, however, offer simple, standard, and scalable communication channels, which are the desired features of many digital systems.

Bus technology is more complicated than the technology of direct communication channels. Signal propagation on busses is a complex phenomenon that is ignored or poorly dealt with in many systems. Bus driving technologies use special drivers for transmitting signals along busses. Most digital systems employ the digital abstraction and ignore the analog nature of busses. But even with the digital abstraction, some analog issues of busses may still be noticeable, such as effects of signal reflections, transient glitches, and analog noise. To overcome these issues, most digital busses are slowed down until they work properly. As a result, digital communication over busses tend to be slower than the communication over direct channels. These penalties can be minimized by careful engineering of the electrical bus in its intended environment.

5.2 Communication architectures

Many schemes have been suggested as the interconnection infrastructure for supporting various communication patterns in digital systems, including point-to-point wires, multistage interconnection networks, and bussed interconnections. In Chapter 2, we investigated how busses (multiple-pin wires) can be employed to efficiently realize certain classes of permutations among modules in a digital system. We demonstrated that by connecting modules with bussed interconnections, as opposed to point-to-point wires, the number of pins per module can often be significantly reduced.

Our bussed approach to realizing permutations compares favorably with both the point-to-point and the multistage-interconnect approaches. Bussed permutation architectures realize general classes of permutations in one clock cycle, exhibit small number of pins per module, and use virtually no switching hardware. Point-to-point architectures, for comparison, can support any communication pattern in one clock cycle, utilize no switching hardware, but use

many pins per module. Multistage interconnection architectures, as another alternative, realize general classes of permutations, exhibit a constant number of pins per module, but operate in multiple clock cycles and use a considerable amount of switching hardware. We conclude that bussed interconnections constitute an attractive alternative as a communication architecture. It would be interesting to study other classes of communication patterns that can be efficiently implemented on bussed interconnections.

Several theoretical studies of systems with bussed interconnections use hypergraphs to model such systems. The topology of a system with bussed interconnections can be modeled as a hypergraph, much as the topology of a system with point-to-point wires can be modeled as a graph. (See [9] for definitions and basic properties of graphs and hypergraphs.) In systems with bussed interconnections, system modules are modeled as hypergraph nodes and the busses (multiple-pin wires) are modeled as hyperedges. This analogy enables many graph-theoretic results to be interpreted in the domain of architectural design, as was done for instance in [10, 11, 13, 30, 48, 49, 64, 73, 77]. We believe that more research in this direction would be fruitful.

The problem of realizing permutations on uniform architectures in several clock cycles presents an interesting direction for further exploration. Our research nave demonstrated that cyclic shifts, for example, can be uniformly realized in t clock cycles by uniform architectures with $O(n^{1/2t})$ pins per module. It would be interesting to develop a pin-time tradeoff for general classes of permutations on bussed architectures, similar to the tradeoff exhibited by multistage interconnection networks and point-to-point wires. An advantage of generalized pin-time bussed interconnections, over multistage interconnection networks, would be the avoidance of special switching hardware.

5.3 Control mechanisms

Numerous digital systems use busses for implementing many control mechanisms. Busses are useful media for broadcasting control signals and for performing various systemwide protocols. In Chapter 3, we explored how busses can be efficiently used for arbitration. We focused on distributed asynchronous priority arbitration schemes and demonstrated that by using data-dependent analysis, certain popular mechanisms can be significantly improved.

In Chapter 3, we investigated bussed priority arbitration mechanisms under a standard digital bus model that assumes a time unit of bus-settling delay for a bus to stabilize to a valid logic value. A more elaborate bus model that takes into account distances between modules and signals propagation was examined in Chapter 4. In both of these bus models, the superiority of the binomial arbitration scheme over the binary arbitration scheme was established. Analyzing these arbitration schemes in an analog model of bus lines, which models various transient effects, would probably be a difficult task. However, simulating the analog behavior of these arbitration schemes could be a tractable goal.

On a more general note, the domain of data-dependent analysis of digital systems has not been investigated much in the past. The results of our work demonstrate that a careful analysis of the delays experienced in existing systems, may result in an improved performance of such systems without changing them. A more systematic approach to analyzing data-dependent delays in digital systems will prove as a valuable tool for digital circuit designers.

5.4 Transmission lines

In Chapter 4 we introduced and examined a digital transmission line model for a bus. In fact, transmission lines exhibit analog behavior, but for the purposes of digital computation they can be modeled as digital devices. The transmission line model enables a bus line to carry multiple transactions at different locations simultaneously. This feature of a bus is utilized in other shared media, such as radio channels and optical communication, but mostly is ignored in electrical busses. It would be interesting to explore ways for using the transmission line properties of electrical busses as well.

The design of digital communication protocols over busses should be a careful engineering task, since high-speed busses are in effect analog transmission lines. Many bus systems work properly only because the busses are slowed down until their analog behavior can be neglected and the digital functions are correctly performed. However, ignoring the analog nature of busses results in severe limitations to the performance of many bussed systems. It would be interesting to investigate other models of transmission lines that capture somewhat more of the analog behavior of this media.

Bibliography

[1] A. Aggarwal, "Optimal bounds for finding maximum on array of processors with k global buses," *IEEE Transactions on Computers,* Vol. C-35, No. 1, January 1986, pp. 62–64.

[2] J. H. Anderson and M. G. Gouda, "A new explanation of the glitch phenomenon," manuscript, August 1988, revised July 1989, to appear in *Acta Informatica.*

[3] R. N. Ashcroft, R. L. Rivest, and S. A. Ward, "Counterexample to arbitration bus scheme," unpublished manuscript, MIT, September 1988.

[4] L. Babai and P. Erdös, "Representation of group elements as short products," *Annals of Discrete Mathematics,* Vol. 12, 1982, pp. 27–30.

[5] R. V. Balakrishnan, "The proposed IEEE 896 Futurebus—a solution to the bus driving problem," *IEEE Micro,* Vol. 4, No. 4, August 1984, pp. 23–27.

[6] K. E. Batcher, "Sorting networks and their applications," *AFIPS Conference Proceedings,* Vol. 37, 1968 Spring Joint Computer Conference, pp. 307–314.

[7] V. E. Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic,* Academic Press, New York, 1965.

[8] J. L. Bentley and D. J. Brown, "A general class of resource tradeoffs," *Journal of Computer and System Sciences,* Vol. 25, No. 2, October 1982, pp. 214–238.

[9] C. Berge, *Graphs and Hypergraphs,* North-Holland Publishing Company, Amsterdam, 1973.

[10] J. C. Bermond, J. Bond, and C. Peyrat, "Interconnection network with each node on two buses," *Proceedings of the International Colloquium on Parallel Algorithms and Architectures,* Marseille Luminy, France, 1986, pp. 155–167.

[11] J. C. Bermond, J. Bond, and J. F. Scalé, "Large hypergraphs of diameter one," in *Graph Theory and Combinatorics, Proceedings of the 1983 Cambridge Colloquium,* Academic Press, London, 1984, pp. 19–28.

[12] D. P. Bertsekas and R. G. Gallager, *Data Networks,* Prentice Hall, Englewood Cliffs, New Jersey, 1987.

[13] Y. Birk, *Concurrent Communication among Multi-Transceiver Stations over Shared Media,* Ph.D. dissertation, Stanford University, March 1987. Also appeared as Technical Report CSL-TR-87-321, Stanford University, March 1987.

[14] G. S. Bloom and S. W. Golomb, "Numbered complete graphs, unusual rulers, and assorted applications," in *Theory and Applications of Graphs,* Y. Alavi and D. R. Lick, eds., Springer-Verlag, New York, 1978.

[15] S. H. Bokhari, "Finding maximum on an array processor with a global bus," *IEEE Transactions on Computers,* Vol. C-33, No. 2, February 1984, pp. 133–139.

[16] P. L. Borill, "Microprocessor bus structures and standards," *IEEE Micro,* Vol. 1, No. 1, February 1981, pp. 84–95.

[17] P. L. Borill, "IEEE P896—the Futurebus project," *Microprocessors and Microsystems,* Vol. 6, No. 9, November 1982, pp. 489–495.

[18] P. L. Borill and J. Theus, "An advanced communication protocol for the proposed IEEE 896 Futurebus," *IEEE Micro,* Vol. 4, No. 4, August 1984, pp. 42–56.

[19] I. Chlamtac and O. Ganz, "A study of communication interfaces for multiple bus networks," *Computer Networks and ISDN Systems,* Vol. 9, No. 3, March 1985, pp. 177–189.

[20] A. L. DeCegama, *Parallel Processing Architectures and VLSI Hardware,* The Technology of Parallel Processing, Vol. 1, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[21] D. Del Corso, "Experiences in designing the M3 backplane bus standard," *Microprocessors and Microsystems,* Vol. 10, No. 2, March 1986, pp. 101–107.

[22] D. Del Corso, H. Kirrman, and J. D. Nicoud, *Microcomputer Buses and Links,* Academic Press, London, 1986.

[23] D. Del Corso and L. Verrua, "Contention delay in distributed priority networks," *Microprocessing and Microprogramming,* Vol. 13, No. 1, January 1984, pp. 21–29.

[24] Digital Equipment Corporation, *The Synchronous Backplane Interconnect,* Vax 11/780 Architecture Handbook, Digital Press, Maynard, 1978.

[25] W. Feit, private communications, 1987.

[26] P. Feldman, J. Friedman, and N. Pippenger, "Wide-sense nonblocking networks," *SIAM Journal of Discrete Mathematics,* Vol. 1, No. 2, May 1988, pp. 158–173.

[27] T. Y. Feng, "A survey of interconnection networks," *IEEE Computer,* Vol. 14, No. 12, December 1981, pp. 12–27.

[28] C. M. Fiduccia, public communication, MIT, 1984.

[29] C. M. Fiduccia, private communication, April 1987.

[30] C. M. Fiduccia, "A bussed hypercube and other optimal permutation networks," presented at the *4th SIAM Conference on Discrete Mathematics,* June 1988.

[31] L. Finkelstein, D. Kleitman, and T. Leighton, "Applying the classification theorem for finite simple groups to minimize pin count in uniform permutation architectures," in *VLSI Algorithms and Architectures,* Lecture Notes in Computer Science, Vol. 319, J. H. Reif, ed., Springer-Verlag, New York, 1988, pp. 247–256.

[32] M. A. Franklin, "VLSI performance comparison of banyan and crossbar communications networks," *IEEE Transactions on Computers,* Vol. C-30, No. 4, April 1981, pp. 283–291.

[33] K. T. Fung and H. C. Torng, "On the analysis of memory conflicts and bus contentions in a multiple-microprocessor system," *IEEE Transactions on Computers,* Vol. C-27, No. 1, January 1978, pp. 28–37.

[34] M. Gardner, *The Incredible Dr. Matrix,* Charles Scribner's Sons, New York, 1976.

[35] M. Garetz, "P696/S100—a bus which supports a wide range of 8- and 16-bit processors," *Microprocessors and Microsystems,* Vol. 6, No. 9, November 1982, pp. 466–470.

[36] L. A. Glasser and D. W. Dobberpuhl, *The Design and Analysis of VLSI Circuits,* Addison-Wesley, Reading, Massachusetts, 1985.

[37] L. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessor systems," *Proceedings of the 1st Annual Symposium on Computer Architecture,* IEEE/ACM, 1971, pp. 21–28.

[38] S. W. Golomb, "How to number a graph," in *Graph Theory and Computing,* R. C. Read, ed., Academic Press, New York, 1972, pp. 23–37.

[39] D. Gorenstein, *Finite Simple Groups,* Plenum Press, New York, 1982.

[40] D. B. Gustavson, "Computer busses—a tutorial," *IEEE Micro,* Vol. 4, No. 4, August 1984, pp. 7–22.

[41] D. B. Gustavson, "Introduction to the Fastbus," *Microprocessors and Microsystems,* Vol. 10, No. 2, March 1986, pp. 77–85.

[42] D. B. Gustavson and J. Theus, "Wire-OR logic transmission lines," *IEEE Micro,* Vol. 3, No. 3, June 1983, pp. 51–55.

[43] M. Hall, Jr., *Combinatorial Theory,* Blaisdell Publishing Company, Waltham, Massachusetts, 1967.

[44] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers,* Oxford University Press, London, 1938.

[45] J. Hayes, "An adaptive technique for local distribution," *IEEE Transactions on Communications,* Vol. COM-26, No. 8, August 1978, pp. 1178–1186.

[46] A. P. Jayasumana and P. D. Fisher, "The token-skipping channel access scheme for bus networks," *Computer Networks and ISDN Systems,* Vol. 9, No. 3, March 1985, pp. 201–208.

[47] J. Juang and B. W. Wah, "A multiaccess bus arbitration scheme for VLSI-densed distributed systems," *AFIPS Conference Proceedings,* Vol. 53, 1984 National Computer Conference, pp. 13–22.

[48] J. Kilian, S. Kipnis, and C. E. Leiserson, "The organization of permutation architectures with bussed interconnections," *Proceedings of the 28th Annual Symposium on Foundations of Computer Science,* IEEE, 1987, pp. 305–315.

[49] J. Kilian, S. Kipnis, and C. E. Leiserson, "The organization of permutation architectures with bussed interconnections," Technical Report MIT/LCS/TM-379, MIT, January 1989. To appear in *IEEE Transactions on Computers.*

[50] S. Kipnis, "Bus-based priority arbitration system with optimum codewords," Patent Application, MIT-5126, MIT, October 16, 1989.

[51] S. Kipnis, "Priority arbitration with busses," *Proceedings of the Sixth MIT Conference on Advanced Research in VLSI,* MIT, 1990, pp. 154–173. Also appeared as Technical Report MIT/LCS/TM-408, MIT, October 1989.

[52] R. R. Koch, *An Analysis of the Performance of Interconnection Networks for Multiprocessor Systems,* Ph.D. Dissertation, Department of Mathematics, Massachusetts Institute of Technology, May 1989. Also appeared as MIT VLSI Memo, No. 89-561, Sept 1989.

[53] C. P. Kruskal and M. Snir, "The performance of multistage interconnection networks for multiprocessors," *IEEE Transactions on Computers,* Vol. C-32, No. 12, December 1983, pp. 1091–1098.

[54] T. Lang, M. Valero, and M. A. Fiol, "Reduction of connections for multibus organization," *IEEE Transactions on Computers,* Vol. C-32, No. 8, August 1983, pp. 707–715.

[55] D. H. Lawrie, "Alignment and access of data in an array processor," *IEEE Transactions on Computers,* Vol. C-24, No. 12, December 1975, pp. 1145–1155.

[56] J. Leech, "On the representation of $1, 2, \ldots, n$ by differences," *Journal of the London Mathematical Society,* Vol. 31, 1956, pp. 160–169.

[57] J. V. Levy, "Busses, the skeleton of computer structures," in *Computer Engineering: DEC View of Hardware Systems Design,* by C. G. Bell, J. C. Mudge, and J. E. McNamara, Digital Press, Bedford, Massachusetts, 1978.

[58] D. J. Lewis, *Introduction To Algebra,* Harper and Row, New York, 1965.

[59] G. J. Lipovski, A. Goyal, and M. Malek, "Lookahead networks," *AFIPS Conference Proceedings,* Vol. 51, 1982 National Computer Conference, pp. 153–165.

[60] R. J. Lipton and R. Sedgewick, "Lower bounds for VLSI," *Proceedings of the 13th Annual Symposium on the Theory of Computing,* ACM, 1981, pp. 300–307.

[61] E. C. Luczak, "Global bus computer communication techniques," *Proceedings of the Computer Networking Symposium,* IEEE, 1978, pp. 452–464.

[62] C. A. Mead and L. A. Conway, *Introduction to VLSI Systems,* Addison-Wesley, Reading, Massachusetts, 1980.

[63] R. M. Metcalfe and D. R. Boggs, "Ethernet: distributed packet switching for local computer networks," *Communications of the ACM,* Vol. 19, No. 7, July 1976, pp. 395–404.

[64] M. D. Mickunas, "Using projective geometry to design bus connection networks," *Proceedings of the Workshop on Interconnection Networks for Parallel and Distributed Processing,* ACM/IEEE, 1980, pp. 47–55.

[65] K. P. Mikkilinei and S. Y. W. Su, "An analysis of sorting algorithms for common-bus local networks," *Journal of Parallel and Distributed Computing,* Vol. 5, No. 1, February 1989, pp. 59–81.

[66] J. C. P. Miller, "Difference bases, three problems in additive number theory," in *Computers in Number Theory,* A. O. L. Atkin and B. J. Birch, eds., Academic Press, London, 1971, pp. 299–322.

[67] W. H. Mills and D. H. Wiedemann, "A table of difference coverings," unpublished abstract, Institute for Defense Analyses, Communications Research Division, January 1988.

[68] W. H. Mills and D. H. Wiedemann, private communication, November 1988.

[69] T. N. Mudge, J. P. Hayes, G. D. Buzzard, and D. C. Winsor, "Analysis of multiple-bus interconnection networks," *Journal of Parallel and Distributed Computing,* Vol. 3, No. 3, September 1986, pp. 328–343.

[70] T. N. Mudge, J. P. Hayes, and D. C. Winsor, "Multiple bus architectures," *IEEE Computer,* Vol. 20, No. 6, June 1987, pp. 42–48.

[71] L. Nisnevitch and E. Strasbourger, "Decentralized priority control in data communication," *Proceedings of the 2nd Annual Symposium on Computer Architecture,* IEEE/ACM, 1975, pp. 1–6.

[72] R. Palais and L. Lamport, "On the glitch phenomenon," Technical Report CA-7611-0811, Massachusetts Computer Associates, Wakefield, Massachusetts, November 1976.

[73] C. S. Raghavendra, "HMESH: a VLSI architecture for parallel processing," in *Proceedings of CONPAR 86,* Lecture Notes in Computer Science, Vol. 237, G. Goos and J. Hartmanis, eds., Springer-Verlag, New York, 1986, pp. 76–83.

[74] R. D. Rettberg and R. T. Thomas, "Contention is no obstacle in shared-memory multiprocessing," *Communications of the ACM,* Vol. 29, No. 12, December 1986, pp. 1202–1212.

[75] H. J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing,* Lexington Books, D. C. Heath and Company, Lexington, Massachusetts, 1985.

[76] Q. F. Stout, "Mesh-connected computers with broadcasting," *IEEE Transactions on Computers,* Vol. C-32, No. 9, September 1983, pp. 826–830.

[77] Q. F. Stout, "Meshes with multiple busses," *Proceedings of the 27th Annual Symposium on Foundations of Computer Science,* IEEE, 1986, pp. 264–273.

[78] A. S. Tanenbaum, *Computer Networks,* Prentice Hall, Englewood Cliffs, New Jersey, 1981.

[79] D. M. Taub, "Contention resolving circuits for computer interrupt systems," *Proceedings of the IEE,* Vol. 123, No. 9, September 1976, pp. 845–850.

[80] D. M. Taub, "Worst-case arbitration time in S-100 type computer bus systems," *Electronics Letters,* Vol. 18, No. 19, September 1982, pp. 833–835.

[81] D. M. Taub, "Arbitration and control acquisition in the proposed IEEE 896 Futurebus," *IEEE Micro,* Vol. 4, No. 4, August 1984, pp. 28–41.

[82] K. J. Thurber, E. D. Jensen, L. A. Jack, L. L. Kinney, P. C. Patton, and L. C. Anderson, "A systematic approach to the design of digital bussing structures," *AFIPS Conference Proceedings,* Vol. 41, Part II, 1972 Fall Joint Computer Conference, pp. 719–740.

[83] J. D. Ullman, *Computational Aspects of VLSI,* Computer Science Press, Rockville, Maryland, 1984.

[84] J. H. van Lint, "Solutions: Problem 350," *Nieuw Archief voor Wiskunde,* Vol. 22, 1974, pp. 94–109.

[85] T. Verhoeff, "Delay-insensitive codes—an overview," *Distributed Computing,* Vol. 3, No. 1, 1988, pp. 1–8.

[86] A. Waksman, "A permutation network," *Journal of the ACM,* Vol. 15, No. 1, January 1968, pp. 159–163.

[87] S. A. Ward, private communication, May 1989.

[88] S. A. Ward and R. H. Halstead, Jr., *Computation Structures,* MIT Press, Cambridge, Massachusetts, and McGraw-Hill, New York, 1990.

[89] S. A. Ward and C. J. Terman, "An approach to personal computing," *Proceedings of COMPCON,* IEEE, 1980, pp. 460–465.