LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TM-408

PRIORITY ARBITRATION WITH BUSSES

Shlomo Kipnis

October 1989

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) MIT/LCS/TM-408	5. MONITORING ORGANIZATION REPORT NUMBER(S) N00014-87-K-0825
6a. NAME OF PERFORMING ORGANIZATION MIT Laboratory for Computer Science 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research/Department of Navy
6c. ADDRESS (City, State, and ZIP Code) 545 Technology Square Cambridge, MA 02139	7b. ADDRESS (City, State, and ZIP Code) Information Systems Program Arlington, VA 22217
8a. NAME OF FUNDING / SPONSORING ORGANIZATION DARPA/DOD 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Boulevard Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Priority Arbitration with Busses

12. PERSONAL AUTHOR(S)
Kipnis, S.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	1989 October	18

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	arbitration, arbitration priorities, asynchronous arbitration, binary arbitration, binomial arbitration, busses, bus-settling delay, combinational logic, data-dependent delays,

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper explores priority arbitration schemes that employ 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 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 $1 \leq t \leq \lg n$ and $\lg n \leq m \leq n$. Our schemes are based on a novel analysis of *data-dependent delays* and generalize the two known schemes: *linear arbitration*, which with $m = n$ busses achieves $t = 1$

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Judy Little, Publications Coordinator	22b. TELEPHONE (Include Area Code) (617) 253-5894 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

U.S. Government Printing Office: 1986—607-047

18. generalized binomial arbitration, linear arbitration, open-collector busses, priority arbitration, resource tradeoff, wired-OR

19. 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.

Priority Arbitration with Busses

(Extended Abstract)

Shlomo Kipnis

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

October 16, 1989

Abstract

This paper explores priority arbitration schemes that employ 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 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 $1 \leq t \leq \lg n$ and $\lg n \leq m \leq n$. Our schemes are based on a novel analysis of *data-dependent delays* 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.

Keywords: arbitration, arbitration priorities, asynchronous arbitration, binary arbitration, binomial arbitration, busses, bus-settling delay, combinational logic, data-dependent delays, generalized binomial arbitration, linear arbitration, open-collector busses, priority arbitration, resource tradeoff, wired-OR.

This research was supported in part by the Defense Advanced Research Projects Agency under Contract N00014-87-K-0825.

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 [5, 6, 10, 14, 18, 19, 22, 26]). In this paper we focus on distributed *priority arbitration* mechanisms, where contention is resolved using predetermined module priorities and the arbitration process is carried out in a distributed manner at all the 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 module applies to the busses, based on the module's arbitration priority and on logic values on other 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 [23], gained popularity and is used in many modern bus systems, such as Futurebus [7, 25], M3-bus [9], S-100 bus [13, 24], Multibus-II [14], Fastbus [15], and Nubus [28]. Other priority arbitration mechanisms that employ busses are described in [5, 6, 10, 12, 17, 18, 19, 22, 26].

The asynchronous binary arbitration scheme arbitrates among n modules in $t = \lg n$ units of time, using $m = \lg n$ open-collector (wired-OR) 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 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 $\lg n$ 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).

[1] Throughout this paper 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.

	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 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 process takes 4 stages.

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 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 bitwise ORing the four arbitration priorities. After one unit of bus-settling delay (stage 1), bus b_3 settles to the 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. In the meantime, however, each of modules c_9 and c_{10} disables its last two bits, because of the 1 on bus b_2. At the end of stage 2, bus b_2 settles to the 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 low order bits (stage 3), which results in bus b_1 settling to a 1 at the end of stage 3. Finally, in stage 4, module c_9 ceases to apply its last bit, because of the 1 it detects on bus b_1, which results in bus b_0 settling to a 0 at the end of stage 4. This arbitration process required $t = \lg 16 = 4$ stages to complete.

In this paper 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 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, that completes in 2 stages. It turns out that for any subset of the above 16 codewords, the corresponding arbitration process takes at most 2 stages. In Section 3, we show how to design a good set of codewords for general values of n by using *binomial codes* as arbitration priorities.

	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 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 process takes 2 stages.

The remainder of this paper explores priority arbitration schemes that employ busses to arbitrate among n modules. In Section 2 we discuss priority arbitration and formally define the asynchronous model of priority arbitration with busses. Section 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 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. We present a variety of extensions to the results of this paper in Section 5.

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 identified 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 paper 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 we assume are indexed in increasing order of priority. The m open-collector (wired-OR) 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, 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 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.

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 indefinitely (see [1, 20, 21]). The acyclic nature of the arbitration logic imposes a partial order on the busses, which can be extended to a linear order. The significance of the linear order on the busses is that logic values on higher indexed busses can be used to determine logic values of lower indexed busses but not vice versa. We formalize this idea in the following definition of an acyclic arbitration protocol.

Definition 1 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, $f_j : P \times \{0,1\}^{m-j-1} \rightarrow \{0,1\}$, 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 arbitration priority as a parameter. The m arbitration busses are 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 $m-j-1$ bit values on the highest $m-j-1$ 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 $j = 0, 1, \ldots, m-1$. An arbitration process among several contending modules consists of the competing modules applying logic values to the m busses according to the acyclic arbitration protocol of size m.

Measuring the arbitration time of asynchronous mechanisms is somewhat problematic. We follow a standard approach taken in many bus systems (see [6, 10, 11, 14, 16, 24, 25]) and measure the arbitration time in units of bus-settling delay. Bus-settling delay, T_{bus}, is the time it takes for a bus to settle to a stable logic value, once its drivers have stabilized, which includes the delays introduced by the logic gates driving the bus, the bus propagation delay, and any additional time required to resolve transient effects such as the wired-OR

glitch. 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. An arbitration process is modeled as a sequence of stages, each taking T_{bus} time, and the arbitration time is defined as the number of stages it takes until all busses stabilize. This approach models the situation in many bus systems rather accurately. (More discussion of measuring the arbitration time in units of bus-settling delay is deferred until Section 5.)

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 2 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 sequence of values $\langle v_{m-1}[t], \ldots, v_1[t], v_0[t] \rangle$.

Definition 2 characterizes an arbitration process as a successive application of the acyclic arbitration protocol F to the set of competing arbitration priorities Q and the current state 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. It is relatively easy to 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, at least one more bus stabilizes on its final value.

A better upper bound for the number of stages taken by arbitration processes is 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 only uses the values of busses in previous levels. More formally, given an acyclic arbitration protocol F of size m, we 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 $B_0, B_1, \ldots, B_{d-1}$, with $f_j \in F_h$ 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. The next theorem shows that any acyclic arbitration protocol of depth d reaches a resolution after at most $t = d$ computation stages.

Theorem 1 *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 $F = \cup_{h=0}^{d-1} F_h$ and $B = \cup_{h=0}^{d-1} B_h$ as 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 $F' = \cup_{h=0}^{d-2} F_h = \langle f_{m-1}, \ldots, f_{r+1}, f_r \rangle$ 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 1 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 this paper, 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 $0 \leq t \leq m$.

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 $\langle v_{m-1}, \ldots, v_1, v_0 \rangle$ 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, 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 3 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

- P is a set of n *arbitration priorities*;
- F is an *acyclic arbitration protocol* of size m for P;
- 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 3 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 Asynchronous Priority Arbitration Schemes

In this section we first use our framework to 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 $t = \frac{1}{2} \lg n$.

The Asynchronous 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 [10, 12, 18, 26]).

Formally, we define this scheme as $\text{LINEAR}(n,n,1) = \langle P, F, \text{WIN} \rangle$, where

- $P = \{p_i = 0^{n-i-1}\ 1\ 0^i \ : \text{ for } i = 0, 1, \ldots, n-1\}$.
- $F = \langle f_{n-1}, \ldots, f_1, f_0 \rangle$, where $f_j(p, v_{m-1} \ldots, v_{j+1}) = p^{(j)}$, for $j = 0, 1, \ldots, n-1$.
- $\text{WIN}(0^k\ 1\ \alpha) = 0^k\ 1\ 0^{n-k-1} = p_{n-k-1}$, for $0 \leq k \leq n-1$ and any $\alpha \in \{0,1\}^{n-k-1}$.

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 1 shows that the asynchronous linear arbitration scheme takes at most $t = 1$ stages to arbitrate.

The Asynchronous 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 [23], and is also known as encoded arbitration (see [6, 10, 14, 24, 25]).

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

- $P = \{p_i = \epsilon_{m-1} \cdots \epsilon_1 \epsilon_0$: where $\epsilon_{m-1} \cdots \epsilon_1 \epsilon_0$ is the binary representation of i, for $i = 0, 1, \ldots, n-1\}$.
- $F = \langle f_{m-1}, \ldots, f_1, f_0 \rangle$, 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 $j = 0, 1, \ldots, m-1$.
- $\text{WIN}(\alpha) = \alpha$, 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 1, that the asynchronous binary arbitration scheme takes at most $t = \lceil \lg n \rceil$ stages to arbitrate. On the other hand, it has been shown in [2, 10, 14, 24, 25, 27] that there are examples where a binary arbitration process takes exactly $\lceil \lg n \rceil$ 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. Our asynchronous binomial arbitration scheme, presented next, guarantees fast arbitration by employing certain codewords that exhibit small data-dependent delays.

The Asynchronous 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 4 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 5 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 introduction 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 3 in Section 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 $\textsc{Binomial}(n, \lceil \lg n + 1 \rceil, \left\lceil \frac{1}{2} \lg n \right\rceil) = \langle P, F, \textsc{win} \rangle$ as follows. We use $m = \lceil \lg n + 1 \rceil$ and $t = \left\lceil \frac{1}{2} \lg n \right\rceil$ for simplicity of notation.

- $P = D(m)$.
- $F = \langle f_{m-1}, \ldots, f_1, f_0 \rangle$, 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 $j = 0, 1, \ldots, m-1$.

- $\textsc{win}(\alpha) = \alpha$, 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 $t = \left\lceil \frac{1}{2} \lg n \right\rceil$ stages. Notice that a standard static analysis of the arbitration circuitry, as given for example in Theorem 1, 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 4, we use a novel dynamic approach of analyzing the data-dependent delays experienced in arbitration processes, and prove the correctness of our scheme as a special case of our generalized binomial arbitration scheme.

4 Generalized Binomial Arbitration

In this section we extend the ideas of the asynchronous binomial arbitration scheme of Section 3 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 the generalized binomial arbitration scheme is approximately $m = \frac{1}{e} t n^{1/t}$. 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 the generalized binomial arbitration scheme is based on an analysis of data-dependent delays.

We first define the set of generalized binomial codes of length m and diversity r.

Definition 6 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 in 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 2 *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 there are exactly that many possibilities of choosing at most r switching points out of m possible positions. ■

Corollary 3 *There are at least 2^{m-1} binomial codes of length m.*

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

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

The sum includes the first $\left\lceil \frac{1}{2}(m-1) \right\rceil + 1$ binomial coefficients, which constitute at least a half of all the $m+1$ binomial coefficients. The partial sum is therefore at least a half of the full sum, which is 2^m. We therefore conclude that $|D(m)| \geq \frac{1}{2} \cdot 2^m = 2^{m-1}$. ■

The Asynchronous Generalized Binomial Arbitration Scheme

This scheme uses m busses and arbitrates in at most t stages, for $t \leq m$. 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, 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$(n,m,t) = \langle P, F, \text{WIN} \rangle$, for $n = \sum_{l=0}^{t} \binom{m}{l}$, as follows.

- $P = G(m,t)$.
- $F = \langle f_{m-1}, \ldots, f_1, f_0 \rangle$, 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 $j = 0, 1, \ldots, m-1$.
- $\text{WIN}(\alpha) = \alpha$, 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 m-bit codewords are used, arbitration processes 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.

Theorem 4 *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 $p = 0^{m_0}1^{m_1}0^{m_2}1^{m_3}\cdots\delta^{m_r}$, 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 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. It is easy to 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 5 *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 4 guarantees that the arbitration process on Q, with p as the highest arbitration priority, takes no more than t stages. ■

The Generalized Binomial Arbitration Tradeoff

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 $m = \frac{1}{e} t n^{1/t}$. Figure 3 presents this bus-time tradeoff for a system consisting of $n = 64$ modules. The number of busses varies from $\lg n = 6$ to $n = 64$, and the arbitration time is in the range 1 to $\lg n = 6$ stages. Generalized binomial arbitration reduces to binary arbitration with $m = \lceil \lg n \rceil = 6$ busses, to binomial arbitration with $m = \lceil \lg n + 1 \rceil = 7$ busses, and to a modified version of linear arbitration (see Section 5) with $m = n = 64$ busses.

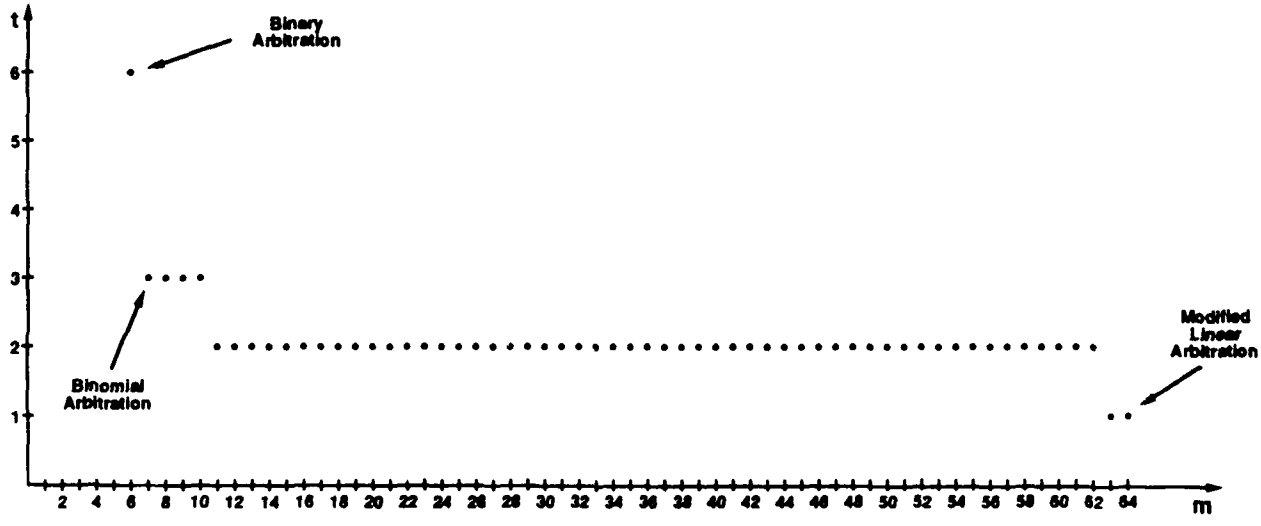

Figure 3: Bus-time tradeoff of the generalized binomial arbitration scheme for $n = 64$ modules, using $6 \leq m \leq 64$ busses and $1 \leq t \leq 6$ stages.

Figure 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 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 [11] have also discovered that by excluding certain codewords, the arbitration time of binary arbitration can be reduced. We, however, give the first general scheme that provides a full spectrum of bus-time tradeoff.

5 Extensions

This section contains some discussion, additional results, and directions of research concerning priority arbitration with busses.

Bus Propagation Delay, Settling Time, and Wired-OR Glitch

High-speed busses are commonly modeled as electrical transmission lines, where it takes some finite amount of time for a signal to propagate through the bus and bring the bus to a stable logic value. In addition, there are the response time of logic gates and the effect of the wired-OR glitch that 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 [3, 8, 16, 25]), seem to support our model.

Some authors carry out a more elaborate analysis of high speed busses (see [2, 8, 23, 24, 25]), which takes into account the distances between modules on the bus and imposes certain assumptions on the arbitration priorities. In [24, 25], for example, Taub assumes geographical ordering of module priorities and equal distances between modules on a backplane bus. Counterexamples to Taub's analysis, where these requirements are not met, have been found [2, 27]. Our model, on the other hand, is applicable to a wider classes of systems, such as data communication broadcast channels and bus systems were priorities and module locations are not predetermined and fixed.

The Asynchronous k-ary Arbitration Scheme

The linear arbitration and binary arbitration schemes of Section 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 our generalized binomial arbitration scheme.

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.

Modified Linear Arbitration

A modified version of linear arbitration, which uses the same acyclic arbitration protocol of binary arbitration, achieves the same bus-time tradeoff as linear arbitration. This version is the generalized binomial arbitration scheme with $m = n$ busses and $t = 1$ time, where the arbitration priority of module c_i is $p_i = 0^{n-i-1}\ 1^{i+1}$, for $i = 0, 1, \ldots, n-1$. This observation poses an interesting question regarding the universality of the acyclic arbitration protocol of binary arbitration.

Lower Bound for Asynchronous Priority Arbitration

The asynchronous generalized binomial arbitration scheme achieves a bus-time tradeoff of the form $n = \sum_{l=0}^{t} \binom{m}{l}$, where n is the number of modules, m is the number of busses, and t is the arbitration time. We conjecture that this tradeoff is optimal for our asynchronous priority arbitration model, in that no more than $n = \sum_{l=0}^{t} \binom{m}{l}$ modules that can be arbitrated with m busses in at most t stages.

Synchronous Priority Arbitration Schemes

In this paper 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, we can show that a synchronous version of k-ary arbitration achieves a bus-time tradeoff of the form $m = n^{1/t}$ and that this tradeoff is optimal in a related synchronous model of arbitration. We can also demonstrate how to combine asynchronous combinational schemes with synchronous clocked schemes to achieve a wide spectrum of bus-time tradeoff.

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 [4] 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 [4].

Acknowledgements

Charles Leiserson of MIT introduced me to the problem of bus arbitration, offered many meaningful insights, and assisted in the related patent application process. Steve Ward of MIT helped with issues related to backplane bus systems. Hershel Safer of MIT provided many helpful suggestions. I would also like to thank Joe Kilian, Tom Knight, Tom Leighton, James Park, and John Wolfe of MIT, Mark Manasse, James Saxe, Chuck Thacker of DEC SRC, Nicholas Pippenger of the University of British Columbia, and Marc Snir of IBM T. J. Watson Research Center, for their comments and references.

References

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

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

[3] 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.

[4] 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.

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

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

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

[8] 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.

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

[10] D. Del Corso, H. Kirrman, and J. D. Nicoud, *Microcomputer Buses and Links,* Chapter 5, Academic Press, London, 1986.

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

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

[13] 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.

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

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

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

[17] 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.

[18] 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.

[19] E. C. Luczak, "Global bus computer communication techniques," *Proceedings of the Computer Networking Symposium,* IEEE, December 1978.

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

[21] C. L. Seitz, "System timing," in *Introduction To VLSI Systems,* by C. A. Mead and L. A. Conway, Chapter 7, Addison-Wesley, Reading, Massachusetts, 1980.

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

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

[24] 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.

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

[26] 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.

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

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

OFFICIAL DISTRIBUTION LIST

Director
Information Processing Techniques Office
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

2 copies

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217
Attn: Dr. Gary Koop, Code 433

2 copies

Director, Code 2627
Naval Research Laboratory
Washington, DC 20375

6 copies

Defense Technical Information Center
Cameron Station
Alexandria, VA 22314

12 copies

National Science Foundation
Office of Computing Activities
1800 G. Street, N.W.
Washington, DC 20550
Attn: Program Director

2 copies

Dr. E.B. Royce, Code 38
Head, Research Department
Naval Weapons Center
China Lake, CA 93555

1 copy