A multiaccess bus arbitration scheme for VLSI-densed distributed systems

by JIE-YONG JUANG
and BENJAMIN W. WAH
Purdue University
West Lafayette, Indiana

ABSTRACT

A VLSI-densed shared-bus distributed system is a computer system consisting of a large number of VLSI processing units (VPUs) connected to one another by a high-speed bus. Data traffic in such a system is characterized by three distinct features: large population, bursty transmission, and task-dependent accesses with priority. A bus arbitration scheme is required to resolve contentions when several VPUs generate requests simultaneously. Conventional schemes such as daisy chaining, polling, and independent requests are shown to be inadequate. In this paper, a multiaccess code-deciphering (MACD) scheme is proposed. Two versions of the scheme are studied. The first version is a load-dependent scheme that can resolve contentions of N VPUs in an average time of $O(\log_{K/2} N)$ steps where K is equal to the bus width. The second version estimates the number of contending VPUs and resolves contention in a constant average time independent of load. The proposed schemes can support task-dependent accesses with priority.

INTRODUCTION

Recent advances in very large scale integrated logic (VLSI) and communication technology, coupled with the explosion in size and complexity of new applications, have led to the development of distributed computing systems. These systems possess a large number of general- and special-purpose processing units joined by an interconnection network. Notable examples are the PUMPS architecture,[1] the systolic-array architecture,[2] the recently announced Cyberplus computer,[3] and specialized systems, such as the processors at the joints of robot arms. PUMPS is a pattern analysis and image database machine that incorporates pools of special-purpose VLSI processing units. In a systolic-array architecture, sets of VLSI systolic processors, which perform functions such as matrix inversion, fast Fourier transform, and sorting, are connected to a host. The Cyberplus computer has a maximum configuration of 64 processors and a speed of 16 billion calculations per second. We call this kind of system a VLSI-densed system, and the processing unit, a VPU.

In a VLSI-densed system, one of the most important issues is the connection of the VPUs. A shared bus is widely used because of its simplicity in connection, flexibility in expansion, and efficiency in communication. Figure 1 depicts a typical configuration of such a system. Wah has shown that a shared bus provides enough bandwidth for a large class of VLSI-densed systems.[4] Large computer systems usually implement a number of relatively independent shared buses. The Cyberplus Computer has four independent "rings" that can partition the processors for four different applications.

In this paper, we propose a bus arbitration scheme for resolving contentions when several VPUs try to access the bus simultaneously. Characteristics of data traffic in a VLSI-densed system are discussed in the next section. Three conventional bus arbitration schemes, namely daisy chaining, polling, and independent requests are compared.[5-7] These schemes are found to be inadequate for VLSI-densed systems. A load-dependent Multiaccess Code-Deciphering (MACD) bus arbitration scheme is proposed, and this scheme is extended so that an estimate of the number of contending VPUs is taken into account. The enhanced scheme can resolve contentions in a constant average time, independent of the number of contending stations.

CONVENTIONAL BUS ARBITRATION SCHEMES

The operations of a VPU alternate between computations and data communications. We assume that when a VPU requests bus access, it has a large volume of data to transmit and requires a rapid response. That is, there is a large peak-to-average ratio of bus use. This type of data traffic is called *bursty* traffic.[8] Another characteristic of data traffic is that messages may have different priorities. Priority, in turn, depends on the urgency with which the bus is needed by a certain VPU for executing a task. The bus should be granted to the message with the highest priority.

On the other hand, a bus shared by autonomous VPUs alternates between bus contentions and data transmissions (Figure 2). A VPU with data ready to transmit is allowed to contend for the bus during a contention period. In order to resolve the contentions in the minimum amount of time, a good bus arbitration scheme should be used. Three bus arbitration schemes have been proposed for conventional computer systems. They were identified by Thurber as daisy chaining, polling, and independent requests.[7]

In daisy chaining, all input–output devices are connected serially along a common control line. During the bus-granting process, a bus grant signal propagates sequentially, device by device, until a requesting device is encountered. This device blocks further propagation of the signal and gains control of the bus by setting the bus busy line. This scheme involves the

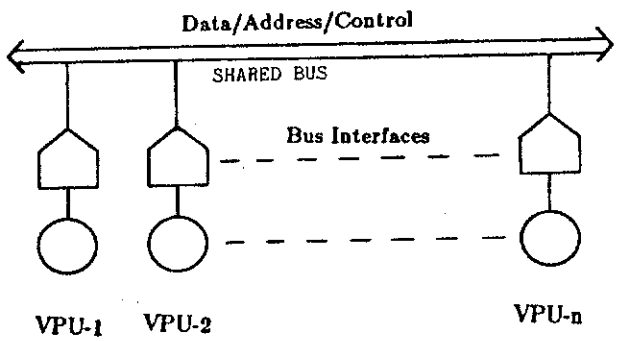

Figure 1—Configuration of a VLSI-densed system

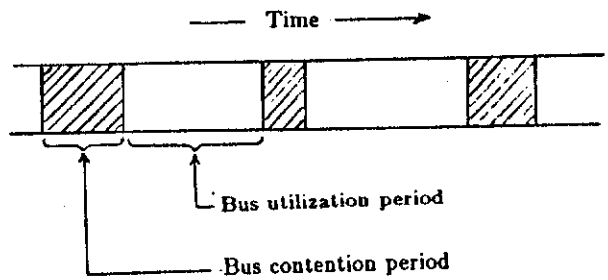

Figure 2—Operation mode of a shared bus

use of at least three control lines: bus grant, bus request, and bus busy.

In a bus system with polling, a set of poll count lines is connected directly to all the devices on the bus. In response to bus requests, a sequence of numbers, each of which corresponds to the address of a device, is generated on the poll count lines. When a requesting device finds that its address matches the number on the poll count lines, the bus is granted to this device, and the bus busy line is set. This scheme requires $\lceil \log_2 M \rceil$ poll count lines, where M is the number of devices on the bus, and two additional control lines are for bus request and bus busy.

In an independent-request scheme, each device has a separate pair of bus request and bus grant control lines connected to the arbitrator. When a device requests bus access, it sends a request signal on its bus request line. Bus control will be granted to one of the requesting devices based on predetermined priorities assigned to the devices. For M devices on a system implementing this scheme, more than 2M control lines are necessary. This scheme is the most costly as far as the number of control lines is concerned.

As VLSI-densed systems bear distinctions in the operating environment from that of conventional systems, the above bus arbitration schemes are found to be inadequate. We examined these schemes with respect to the control line complexity, the time complexity and the capability of task-dependent priority accesses.

1. Control-line complexity. The polling scheme is impractical when the number of VPUs is large because the number of poll count lines must be large enough so that each VPU can be identified by a unique address. A pair of control lines is needed for each VPU in the independent-request scheme. This is impractical even when the number of devices is moderately large.
2. Time complexity. Daisy chaining and polling are basically sequential schemes. They are inadequate for handling bursty traffic, which is characterized by a high ratio of peak-to-average data transmission rate and the fact that only a few VPUs are requesting bus access at any time. Suppose there are N out of M independent requesting devices, the average time to identify a requesting device is M/N. When N is small and the data transmission time is short, the overhead for bus arbitration is large.
3. Capability of task-dependent priority accesses. Priority of a device connected in a daisy chain is determined by its physical position in the chain. In a polling scheme, it is determined by the device's order in the sequence of polling counts. The priorities of the bus request lines in an independent-request scheme are usually fixed at design time. Since the priority of devices cannot be changed easily, the three existing schemes are incapable of handling task-dependent priority accesses.

The above observations reveal that none of the three conventional bus-arbitration schemes is sufficient for the needs of VLSI-densed systems. They call for a new arbitration scheme that can handle bursty traffic and that will access with priority.

LOAD-DEPENDENT MULTIACCESS ARBITRATION SCHEME FOR VLSI-DENSED SYSTEMS

In this section, a deterministic MACD scheme is presented. The scheme is discussed with respect to access without and with priority.

MACD Bus Arbitration for Access without Priority

We have previously studied a window search scheme to resolve contentions in a local multiaccess network.[9,10] In that scheme, a global window is maintained by all the stations, and each contender generates a contending parameter. A contender is eliminated from contention if its parameter is outside the window. A distributed control rule is applied to expand or to shrink the window in each contention step. As the contending process proceeds, the window size becomes smaller and smaller. Eventually, a unique contender is isolated in the window.

We can adapt the above scheme for resolving bus contentions. To support the scheme, two mechanisms are needed: a collision detection mechanism and a window control mechanism. The collision detection mechanism can be implemented by using the Wired-OR property of the bus. When two or more VPUs write simultaneously on the bus, the result is simply the bitwise logical OR of these numbers. By interpreting the result after a write, each VPU can determine whether a collision has occurred. The window control scheme described in References 9 and 10 is based on information of previous contentions and an estimate of the channel load. It is too complicated to be useful in the bus environment. The MACD technique, however, is a fast and effective scheme that combines window control and collision detection in a simple manner.

To describe the scheme formally, let us assume that there are N requesting VPUs, and each VPU writes a binary number X_i $(i = 1, 2, \ldots, N)$ to the bus. The X_is are chosen from a structured code space S with the following properties:

$$X_i, X_j \in S, \quad i \neq j, \text{ are related, i.e., } X_i > X_j \text{ or } X_i < X_j \tag{1}$$

$$f(X_1 \oplus X_2 \oplus \cdots \oplus X_N) \leq \max\{X_1, X_2, \ldots, X_N\} \; X_i \in S, N \geq 1 \tag{2}$$

where $\oplus$ is the bitwise logical OR operator. By reading data on the bus and applying the code-deciphering function, f, a VPU knows the maximum number written on the bus. This information provides a basis for the window search mechanism to set the window. If the initial window is set so that the maximum value is included in the window, then an optimal detection procedure can be designed so that exactly one VPU will be isolated finally.

In order for the MACD technique to work properly, we need to prove that a code space that satisfies Equations 1 and 2 does exist. The following theorem shows the existence of at least one such code space.

Theorem: There exists a code space S of n-bit binary numbers and a deciphering function f which satisfy the constraints in Equations 1 and 2.

Proof: Let $S = \{0^a 10^b \mid a + b = n - 1, \quad a \geq 0, \quad b \geq 0\}$ where 0^k represents a consecutive sequence of k zeroes. Then for any two different elements u and v in S, it is easy to verify the relatedness property. For any n-bit binary number, $X = (x_1 x_2 \cdots x_n)$, we define a deciphering function f on X such that:

$$f(X) = 0^p 10^{n-p-1}, \quad \text{if } x_{p+1} = 1,\ x_j = 0 \text{ for all } 1 \leq j \leq p.$$

We claim that S and f as defined above satisfy Equations 1 and 2. To verify this, we can define N codes such that:

$$c_i = 0^{a(i)} 10^{n-a(i)-1}, \quad i = 1, \ldots, N$$

By definition of S,

$$c_i \in S,$$

and

$$\max(c_1, c_2, \ldots, c_N) = 0^m 10^{n-m-1}$$

where $m = \min\{a(i) \mid i = 1, 2, \ldots, N\}$. An overlapped variable $Y = (y_1 y_2 \cdots y_n)$ is defined to be the bitwise logical OR of the c_is; that is,

$$(y_1 y_2 \cdots y_n) = c_1 \oplus c_2 \oplus \cdots \oplus c_N.$$

Y as defined retains the following properties:

$$y_{m+1} = 1,$$

and

$$y_k = 0 \quad \text{for all } k \leq m.$$

By definition of the deciphering function f,

$$f(Y) = 0^m 10^{n-m-1}$$

or

$$f(c_1 \oplus c_2 \cdots \oplus c_N) = \max(c_1, c_2, \ldots, c_N).$$

Using code deciphering, a bus arbitration protocol can be designed. The network supporting the protocol should have the following components: a synchronous parallel bus for transmitting data and codes, a bus status control line for indicating the busy status of the bus, and an intelligent VPU-bus interface for each VPU that is capable of (1) sensing the bus-status control line, (2) reading data from the bus, (3) writing data to the bus, (4) generating random codes, and (5) deciphering codes read from the bus. The time interval for generating a random number, writing the number to the bus, and deciphering the code read from the bus is called a *slot*.

Whenever a VPU has data ready to transmit, it checks the bus status first. If the bus is in use, it waits until the bus becomes idle. To contend for the bus, a VPU chooses a code randomly from the code space S and writes it to the bus. The resulting code written on the bus is the bitwise logical OR of all the codes written by the contending VPUs. Each contending VPU reads the resulting code written and computes the deciphered code using the code-deciphering function. It compares the deciphered code with the code generated locally. Three results are possible:

1. the locally generated code is equal to the code read
2. the locally generated code is not equal to the code read but is equal to the deciphered code
3. the locally generated code is equal to neither the code read nor the deciphered code.

The last outcome implies that this VPU has not generated the maximum code and has to wait until the next contention period. The first and second outcomes imply that this VPU has generated the maximum code and should be allowed to transmit. However, there may be other VPUs that have generated the same code. If there are more than one VPU in this set *(hidden collision)*, the contention resolution process has to be repeated. There are two ways to detect hidden collision. First, each VPU in this set generates an *n*-bit random number and writes it to the bus. To prevent the possibility of two VPUs generating the same random number, each VPU can use a distinct n-bit station identification code as the random number. If the number read from the bus matches the number written, then hidden collision has been resolved. If collision is detected, the MACD scheme is repeated. Second, we can assume that hidden collision is not resolved, and the collision-detection process is repeated. The process has to be repeated a number of times until there is high confidence that exactly one VPU is isolated.

When the probability is high that a large number of stations have generated the maximum code, the second method of resolving hidden collision is better because it is very likely that the MACD process has to be repeated, and the time for propagating the random number in the first method is lost. On the other hand, if the probability is high that exactly one station has generated the maximum code, the first method is better because hidden collision can be detected efficiently. In the second method, the code space S is much smaller (the size is n for an n-bit number). As a result, a few additional steps are necessary in order to achieve a high enough confidence that there is no hidden collision. In this paper, we have used the first method of resolving hidden collisions because the number of contending VPUs is usually relatively small compared to the bus width. Even when this is not true, we propose in the next section a method of using a variable-sized code space so that the number of VPUs contending in a slot is small.

It is important to note that the code space discussed in Theorem 1 (unary representation) is not unique. If binary codes are used, Equation 1 is still satisfied. A new code-deciphering function has to be designed so that Equation 2 is satisfied. By detecting the most significant bit that is mismatched among the codes generated by the contending VPUs, half of the stations, on the average, can be eliminated in each contention. This is not as efficient as unary-code representations because 1/W stations remain (W is the bus width) after each contention, if unary codes are used.

MACD Bus Arbitration for Priority Access

In a system with priority accesses, a VPU is assigned a *priority level* by the task that invokes its execution. The set of

VPUs with the same priority level constitutes a *priority class*. The *global priority class* is the class of contending VPUs with the highest priority level in the system. In a contention period, bus control is granted to a VPU that belongs to the global priority class.

To support accesses with priority, the system should be able to identify the global priority. One way to do so is to add a set of control lines to the system, each of which corresponds to a priority level. A requesting VPU is responsible for setting the corresponding priority line. The global priority level can then be identified by finding the control line with the highest priority level that is being set. This scheme works well when there are a limited number of priority levels.

On the other hand, the MACD scheme proposed earlier can be adapted to priority accesses in two ways: First is MACD by code space partitioning. The code space of the original MACD scheme is partitioned into subspaces so that each subspace corresponds to a priority level. The partition should satisfy the following condition:

$$\text{If } X \in S_i,\ Y \in S_j \text{ and } i > j, \text{ then } X > Y$$

where S_i and S_j are subspaces corresponding to priority levels i and j respectively. Using this partitioning, priority levels are encoded into the contending codes, and the deciphering function proposed in Theorem 1 can identify the global priority level and the largest code in this level.

The other method of adaptation is MACD by two-phase identification. The partitioning of code space is practical when the number of priority levels is relatively small as compared to the size of the code space. When the number of priority levels is large, a contention period can be divided into two phases: a priority resolution phase followed by an intraclass contention phase. In the priority resolution phase, a strictly increasing function, which maps a set of priority levels onto a code space, is defined in each contention slot. The mapping is done so that the minimum number of priority levels is assigned to the same code. In a contention slot, every contending VPU writes its code to the bus and deciphers the number read from the bus. A set of VPUs with the highest priority levels (corresponding to the deciphered code) is identified. The process is repeated until the set of VPUs with the highest priority level is identified. When the bus width is larger than or equal to the number of priority levels, this phase can be completed in one contention slot.

Evaluation of Load-dependent MACD Bus Arbitration Scheme

Bus arbitration schemes can be evaluated with respect to the following attributes: complexity of implementation, complexity of contention time, flexibility, reliability, and priority access capability. The MACD scheme requires one control line (bus busy). The control logic for the bus interface is relatively simple. A VPU can be added to or removed from the bus without disturbing other components of the system. This system is, therefore, flexible for expansion and convenient for the removal of faulty units. The MACD scheme can support accesses with priority. Moreover, the scheme is efficient as far as contention time is concerned. The analysis and simulation results are shown in the remaining part of this section.

The time complexity of contention resolution can be measured by the mean number of contention slots in each contention period. To analyze this complexity, let N be the number of contending VPUs at the beginning of a contention period and K be the size of the code space equal to the bus width W. Assuming that codes are chosen randomly, a VPU generates a given code c_i $(i = 1, 2, \ldots, N)$ with probability 1/W. Designate the maximum of N such c_is as c_m, the m-th code in the code space, i.e., $c_m = \max\{c_i \mid i = 1, 2, \ldots, N\}$. If exactly one VPU generates code c_m and other VPUs generate codes less than c_m, then the contention is resolved. The probability for this event to occur is:

$$q(m \mid N, K = W) = N\left(\frac{1}{W}\right)\left(\frac{m-1}{W}\right)^{N-1} \tag{3}$$

Since m ranges from 1 to W and these W events are mutually exclusive, the probability that contention is resolved in one step is $P_{K,W,N}$ where K = W is:

$$\begin{aligned} P_{K,W,N} &= \sum_{m=1}^{W} q(m \mid N, K = W) \\ &= \sum_{m=1}^{W} N\left(\frac{1}{W}\right)\left(\frac{m-1}{W}\right)^{N-1} \\ &= \frac{N}{W^N} \sum_{u=1}^{W-1} u^{N-1} \end{aligned} \tag{4}$$

In Figure 3, $P_{K,W,N}$ is plotted against N/W. It is observed that the probability of success in one attempt is higher if the code space (equal to bus width) is larger and the number of contending VPUs is kept constant. It is observed that $P_{K,W,N}$ is a strictly decreasing function of N and decreases to zero when N is large. This means that the MACD technique is unable to resolve contention in one step when the load is extremely heavy. However, most of the contending VPUs are eliminated in one attempt. The number of survivors is reduced significantly as contention proceeds, and the probability of success is increased consequently. The following analysis demonstrates this phenomenon.

Given that the maximum of codes generated by the contending VPUs is c_m, the m-th code in the code space. Define indicator variables X_i, $i = 1, \ldots, N$,

$$X_i = \begin{cases} 1 & \text{with probability } 1/m \\ 0 & \text{with probability } 1 - 1/m \end{cases}$$

Let

$$Z = \sum_{i=1}^{N} X_i.$$

The random variable Z indicates the number of VPUs that generate c_m in the contention. These VPUs are allowed to contend in the following steps. The expected value of Z given

m, N, and W, E(Z | m, N, W), represents the average number of surviving VPUs. It is easy to show that:

$$E(Z \mid m, N, W = K) = \frac{N}{m}. \tag{5}$$

Furthermore, the probability that the current maximum code with N contending stations and a bus width of W is c_m can be expressed as:

$$p(m \mid N, W = K) = \left(\frac{m}{W}\right)^N - \left(\frac{m-1}{W}\right)^N \tag{6}$$

The expected number of VPUs that would survive a contention is:

$$\begin{aligned} E(Z \mid N, W = K) &= \sum_{m=1}^{W} E(Z \mid m, N, W = K)\, p(m \mid N, W = K) \\ &= \sum_{m=1}^{W} \left(\frac{N}{m}\right)\left[\left(\frac{m}{W}\right)^N - \left(\frac{m-1}{W}\right)^N\right] \\ &= \left[\left(\frac{N}{1} - \frac{N}{2}\right)\left(\frac{1}{W}\right)^N\right] + \left[\left(\frac{N}{2} - \frac{N}{3}\right)\left(\frac{2}{W}\right)^N\right] \\ &\quad + \cdots + \left[\left(\frac{N}{W-1} - \frac{N}{W}\right)\left(\frac{W-1}{W}\right)^N\right] + \left[\frac{N}{W}\left(\frac{W}{W}\right)^N\right] \\ &= \frac{N}{W^N}\left\{\frac{1^{N-1}}{2} + \frac{2^{N-1}}{3} + \cdots + \frac{(W-1)^{N-1}}{W}\right\} + \frac{N}{W} \\ &\le \frac{N}{W^N}\left\{W \cdot \frac{(W-1)^{N-1}}{W}\right\} + \frac{N}{W} \\ &\le \frac{N}{W} - \left(\frac{W-1}{W}\right)^{N-1} + \frac{N}{W} \\ &\le \frac{2N}{W} \end{aligned} \tag{7}$$

The ratio $\gamma = \dfrac{E(Z \mid N, W = K)}{N} \le \dfrac{2}{W}$ is a measure of the average fraction of contending VPUs that can survive a contention. Let $N_t (t = 0, 1, \ldots)$ be the expected number of contending VPUs in step t. By Equation 7, we have

$$N_t \le \left(\frac{2}{W}\right)^t N_0 \qquad t \ge 0.$$

Therefore,

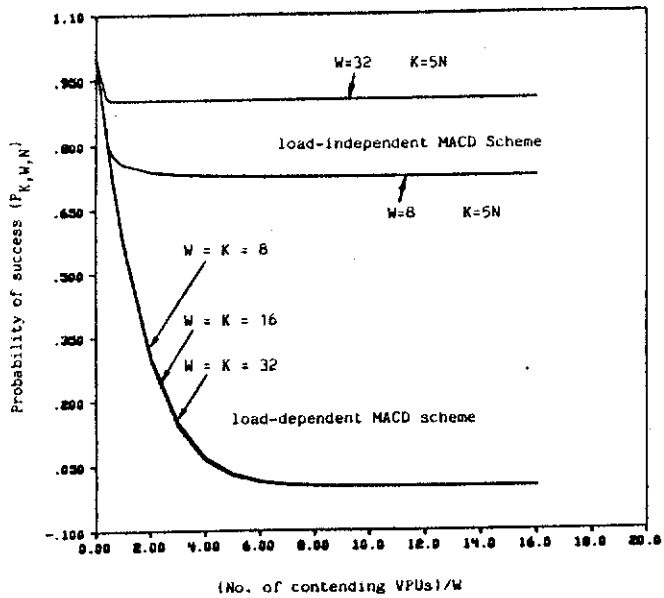

Figure 3—Probability of success in one contention using multiaccess code-deciphering bus arbitration scheme (K is the size of code space; W is the bus width, N is the number of contending VPUs)

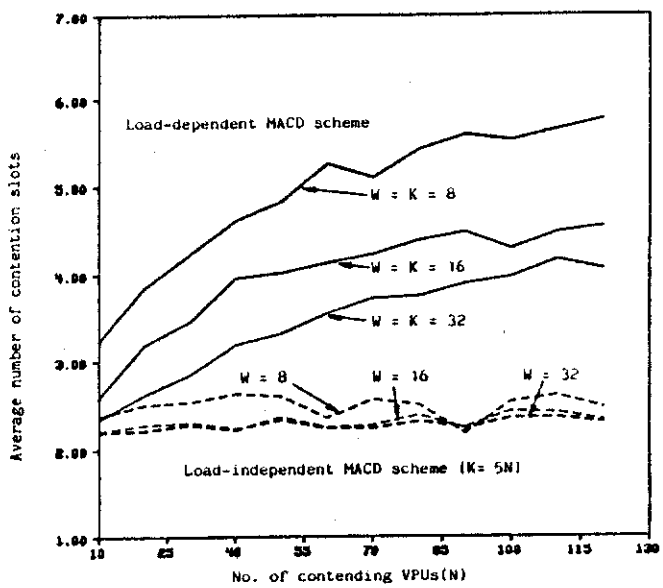

Figure 4—Average number of contention slots for resolving conflicts of bus requests using multiaccess code-deciphering scheme (K is the size of code space; W is the bus width, N is the number of contending VPUs)

$$N_t \to 1 \quad \text{as} \quad t \to \log_{W/2} N_0. \tag{8}$$

As shown in Figure 3, we can see that $P_{K,W,N} \to 1$ as $N < W$, and $P_{K,W,N} \to 0$ as $N \gg W$. This fact reveals that the contention process of MACD can approximately be divided into two phases. The effect of the first phase, that is, when $N_t > W$, is in reducing the number of contending VPUs. When the process enters the second phase, $N_t \le W$, contention can be resolved in about one step. The overall contention process will stop within an average of $\log_{W/2} N_0$ steps. Figure 4 shows the

simulation results that confirm our analysis. The number of contention slots shown includes the additional slots required for resolving hidden collisions. MACD performs better when the bus width is large.

LOAD-INDEPENDENT MACD BUS ARBITRATION SCHEME

As shown in Equation 8 and Figure 4, the scheme proposed in the last section is load-dependent and performs well when the bus width is large and the number of contending VPUs is small. Since the number of contention slots grows logarithmically with the number of contending VPUs, the scheme is inefficient when the number of contending VPUs is large or the bus width is small.

The cause for the load dependency is the fixed code space. In order to reduce the number of VPUs contending in a slot, th code space can be designed so that it is a function of the number of contending VPUs and the bus width. By choosing the size of the code space so that the number of VPUs contending in a slot is a relatively small constant as compared to the bus width, contention can be resolved in a time that is load-independent. We have studied a similar scheme for contention resolution on carrier-sense-multiple-access bus networks.[9,10]

The solution depends on choosing the size of the code space and estimating the number of contending VPUs. Suppose N can be estimated accurately, and a code is chosen so that K/N = r. The probability that contention is resolved in one step (refer to Equation 4) is:

$$P_{K,N,W} = \sum_{m=K-W+1}^{K} q(m \mid N = K/r, K, W) = N\left(\frac{r}{N}\right)^{N} \sum_{u=K-W}^{K-1} u^{N-1} \quad (9)$$

where $q(m \mid N = K/r, K, W)$ is defined in Equation 3. The value of $P_{K,N,W}$ is plotted in Figure 3. It is seen that the success probability is higher and load independent as a result of the increase in the code space size.

The expected number of VPUs that would survive a contention can also be derived similarly. In this case, the number of surviving VPUs is N if no station contends in the slot. That is, Equation 5 is changed to:

$$E(Z \mid m, N = K/r, W) = \begin{cases} \frac{N}{m} & K \le m \le K - W + 1 \\ N & 1 \le m \le K - W \end{cases} \quad (10)$$

The definition of $p(m \mid N, W)$ in Equation 6 remains true. The expected number of surviving VPUs in one contention is:

$$E(Z \mid N = K/r, W) = \sum_{m=1}^{K} E(Z \mid m, N = K/r, W) p(m \mid N = K/r, W)$$

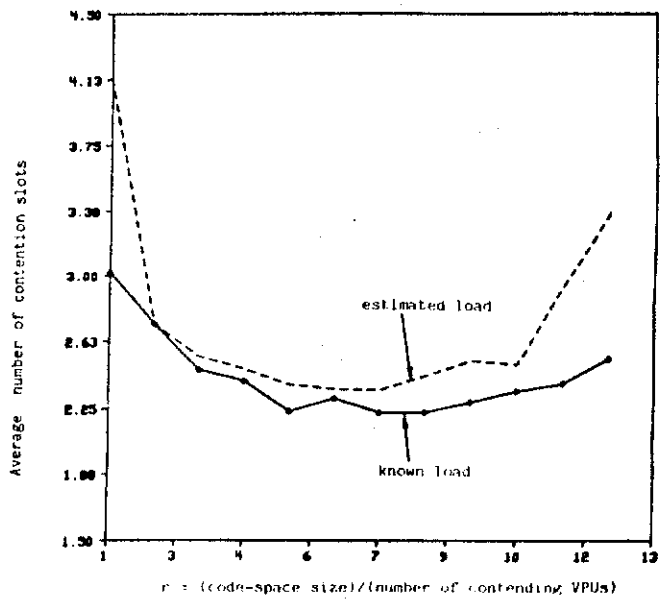

Figure 5—The optimal choice of the code-space size ($W = 16$, $N = 60$)

$$= \sum_{m=K-W+1}^{K} \frac{N}{m} \cdot p(m \mid N = K/r, W) + \sum_{m=1}^{K-W} N \cdot p(m \mid N = K/r, W)$$

$$\le \frac{2N}{K} + \left[\left(\frac{K-W}{K}\right)^{N} - \left(\frac{K-W-1}{K}\right)^{N}\right](K-W)N \quad (11)$$

Since K/N = r is a constant, $E(Z \mid N = K/r, W)$ is a constant independent of load (= N) if K is large as compared to W.

The correct choice of r is shown in Figure 5. There is an optimal choice of r so that the number of contention slots is minimum. The optimal value depends on the value of W and is load independent (assuming that N is known). The value is approximately five for the combinations of W and N tested. Using the optimal value of r, the performance of the load-independent MACD scheme is plotted in Figure 4. In generating these results, the size of the code space, K, is chosen to be W if r × N is smaller than W; that is, the scheme proposed earlier is used when the load is light. It is observed that the proposed scheme requires a small constant number of slots when the load is heavy.

The proposed scheme requires N to be known. In general, this is not possible due to the distributed control. One way is to estimate N based on information collected during the contentions. However, this information can indicate that one or more contending VPUs have generated the same code, but cannot reveal the exact number of contending VPUs. If the number of VPUs contending in a contention slot is small, a reasonable estimate of N can be obtained by using the number

Table I—Comparison of MACD with conventional bus arbitration schemes. (M = number of VPUs connected to the bus)

Comparison of MACD with Conventional Bus-Arbitration Schemes							
Attributes	Hardware Complexity		Contention Time		Reliability	Flexibility	Priority
Schemes	Control Logic	Control Line	light Load	Heavy Load	Failure Tolerance	Easy Reconfig.	Task Dependence
MACD	O(M)	1	~ 1	~ 1	Yes	Yes	Yes
Daisy-Chaining	O(M)	3	O(M)	~ 1	No	No	No
Polling	O(M)	$2+\log_2 M$	O(M)	~ 1	Yes	No	No
Independent Requests	O(M)	2M	$O(\log_2 M)$	$O(\log_2 M)$	Yes	No	No

of bits that are ones in a contention slot, B, as the number of VPUs contending in this slot. That is,

$$\hat{N} = \frac{B \times K}{W} \tag{12}$$

This will systematically underestimate the actual value of N, and some correction to the value of r used should be introduced. In Figure 5, the optimal value of r that should be used is slightly different when the estimate in Equation 12 is used. The number of contention slots required is slightly increased when N is estimated.

A maximum-likelihood estimate of N also can be derived. However, the complexity of such a scheme is high and cannot be used in real-time applications.

CONCLUSION

In this paper, we have studied the problem of bus contentions in VLSI-densed shared-bus systems. Data traffic generated by VPUs in such systems are characterized by three distinct features: large population, bursty transmissions, and task-dependent priority accesses. A bus arbitration protocol is necessary to resolve access conflicts when several VPUs are trying to access the bus simultaneously. Conventional schemes such as daisy chaining, polling, and independent requests are shown to be inadequate because of the large overhead or the high complexity of implementation.

The load-dependent MACD scheme presented in this paper can resolve contention of N VPUs in an average time of $O(\log_{W/2} N)$ steps where W is the width of the bus. For bursty traffic in a system with a parallel bus, N is usually relatively small as compared to W. Nearly perfect bus scheduling is achievable. An extended scheme is proposed that estimates the value of N and uses a code space of variable size depending on N. It is found that contentions can be resolved in a time that is load-independent.

The proposed MACD scheme can support task-dependent priority accesses that cannot be supported by conventional bus arbitration schemes. Comparisons between the MACD and the conventional bus arbitration schemes are summarized in Table I. These comparisons clearly indicate that the MACD scheme is superior in almost every respect.

ACKNOWLEDGMENT

This research was partially supported by National Science Foundation Grant ECS80-16580 and by CIDMAC, a research unit of Purdue University, sponsored by Purdue, Cincinnati Milicron Corporation, Control Data Corporation, Cummins Engine Company, Ransburg Corporation, and TRW.

REFERENCES

1. Briggs, F. A., K. S. Fu, K. Hwang, and B. W. Wah. "PUMPS Architecture for Pattern Analysis and Image Database Management." *IEEE Transactions on Computers*, C-31 (1982), pp. 969–983.
2. Kung, H. T. et al. *VLSI Systems and Computations*. Rockville, Md.: Computer Science Press, 1981.
3. Control Data Corporation. "Technical Information: Control Data Cyberplus." News Release and Fact Sheet, Oct. 1983.
4. Wah, B. W., "A Comparative Study of Distributed Resource Sharing on Multiprocessors." *Proceedings of 10th Annual International Symposium on Computer Architecture*. Stockholm, Sweden: ACM, 1983, pp. 300–308c.
5. Baer, J. *Computer Systems Architecture*, Rockville, Md.: Computer Science Press, 1980.
6. Hayes, J. P. *Computer Architecture and Organization*. New York: McGraw-Hill, 1978.
7. Thurber, K. J. et al. "A Systematic Approach to the Design of Digital Bussing Structures." *AFIPS, Proceedings of the National Computer Conference* (Vol. 41), 1972, pp. 719–740.
8. Tobagi, F. A. "Multiaccess Protocols in Packet Communication Systems." *IEEE Transactions on Communications*, COM-28 (1980), pp. 468–488.
9. Wah, B. W. and Y. Y. Juang, "Load Balancing on Local Multiaccess Networks." *Proceedings of 8th Conference on Local Computer Networks*, Minneapolis, Minn.: IEEE, 1983, pp. 55–61.
10. Juang, J. Y., and B. W. Wah. "Unified Window Protocol for Local Multiaccess Networks." *Proceedings of Third Annual Joint Conference of the IEEE Computer and Communications Societies*. San Francisco, California: IEEE, 1984, pp. 97–104.
11. Metcalf, R. M., and D. R. Boggs. "Ethernet: Distributed Packet Switching for Local Computer Networks." *Communications of the ACM*, 19 (1976), pp. 395–404.