Computers & Elect. Engng Vol. 16, No. 3, pp. 139–158, 1990
Printed in Great Britain

0045-7906/90 $3.00 + 0.00
Pergamon Press plc

A DYNAMICALLY SEGMENTED BUS ARCHITECTURE

XU BAIQIANG and NATHAN IDA
Electrical Engineering Department, The University of Akron, Akron, OH 44325, U.S.A.

(*Received* 27 *February* 1989; *accepted in revised form* 23 *May* 1990)

Abstract—This work presents a dynamically segmented bus (DS-Bus) architecture for disturbed systems as an alternative to existing architectures, such as multi-bus, multi-stage network, or hyper-cube connection. Design considerations of the DS-Bus arbiter, as the key component on the DS-Bus, are discussed. A specific design of the arbiter is given with a resolution time proportional to the number of grants. Analysis of DS-Bus contention is carried out by analytic methods and by simulation. Formulas are derived for parameters such as accept rate, capacity, inter-communication delay, and bandwidth of the DS-Bus. As with other architectures, the performance of the DS-Bus is application sensitive. It will be shown, by comparison, that the DS-Bus architecture is a good approach to organize a large scale parallel computing machine. Finally, a large scale DS-Bus system design is proposed.

1. INTRODUCTION

A single bus is a potential bottleneck for multiprocessor systems and limits improvement of its system performance. However, most commercial multiprocessor systems employ this conventional bus structures. This approach results in systems which can only accommodate a few dozen processors. A recent study [1] showed that system performance can be improved considerably by increasing the capability of the bus. This paper explores a possible enhancement to conventional bus structures to support large numbers of processors.

In recent years, multistage network architectures were discussed extensively. One of the most serious problems with these architectures was observed by Pfister and Norton [2] in the form of so-called hot-spot contentions in shuffle-exchange networks. Their model shows that the effective bandwidth of the memory system falls off dramatically as references to a "hot" memory bank increase, or as the number of processors increases. This generic problem arises from the nature of shared memories, and therefore is intrinsic to multiprocessor systems. Among others, the indirect means of communication increases the utility of the interconnection network and memory banks since one communication needs two shared memory accesses. The sender and receiver compete with each other for the same path and the same memory banks during communication (Fig. 1).

In a distributed system, there are no shared memory banks and direct communication is established by either packet or circuit switching networks. The hyper-cube is one example. This kind of network can support full range of communication rates without saturation. The price for this is longer communication time and much more complicated hardware.

The DS-Bus architecture proposed here is an attempt to retain both short communication time and the simplicity of a bus structure while achieving some characteristics of hyper cube distributed systems, namely, high communication rate and direct communication.

2. STRUCTURE OF THE DS-BUS

Figure 2 shows the basic configuration of a DS-Bus system in which N, the number of processing elements, is equal to eight. The ordering of the PE's, switches and bus segments is arbitrary but the sequence in Fig. 2 will be used throughout this work.

The DS-Bus

Three groups of signals are defined on the DS-Bus as in Fig. 3(a). These are message, arbitration, and control signals. Message signals consist of Data and data-ID. The data is the information to be transferred; the data-ID is the identification of the data. Unlike a conventional bus, the data-ID

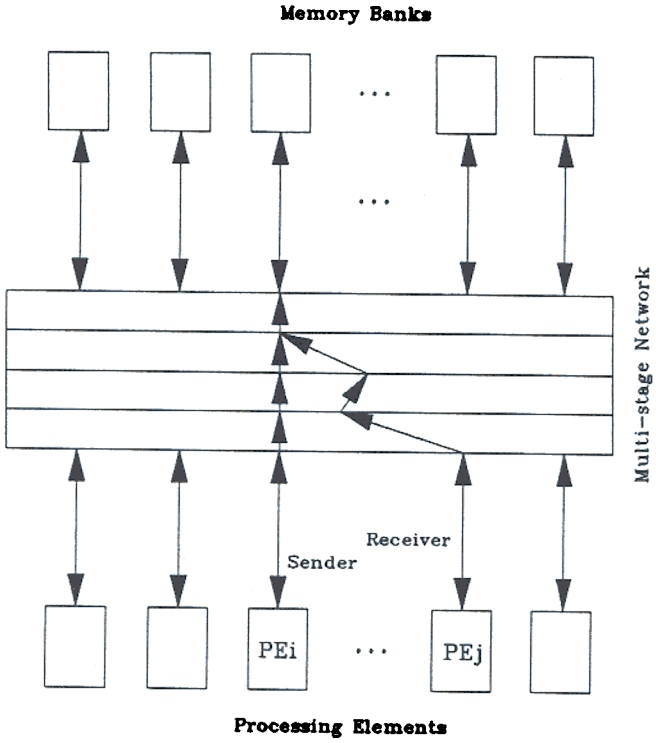


Fig. 1. Contention for network and memory banks between Sender and Receiver in a multi-stage architecture.

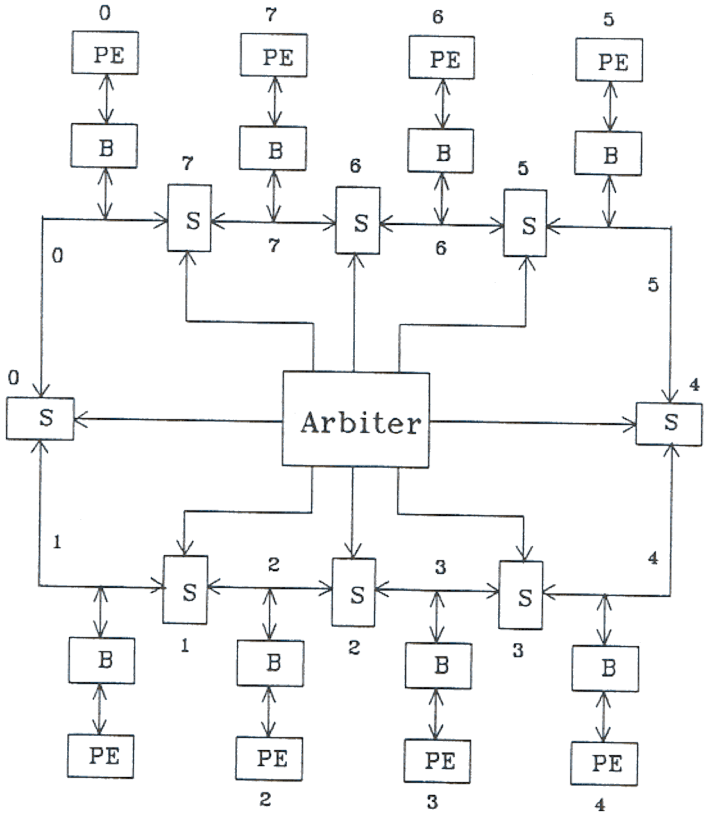
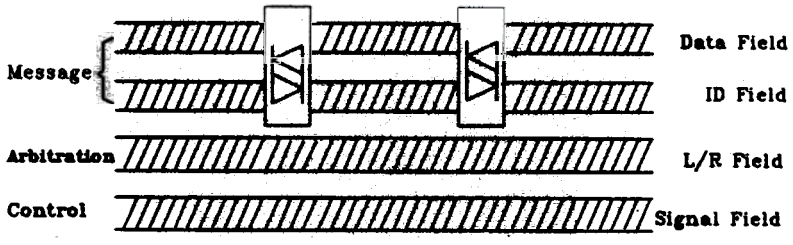
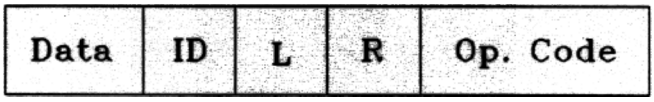


Fig. 2. The basic DS-Bus architecture ($N = 8$).



(a)



(b)

Fig. 3. The bus and Request format. (a) Bus layout. (b) Request format layout.

is necessary in order to have meaningful data. The arbitration signals have Left and Right which are used together to specify the range on which the communication takes place. Both are the relative locations on the bus. For instance, PE6 requests to send a message to PE11. Bus segments from 6 to 11 are needed in order to accomplish the communication. Consequently, Right is 11 and Left is 6.

Three operations are defined on the DS-Bus: Write, Read, and Broadcast. The corresponding signals belong to the control signal group. The Write operation transfers a message from initiator to recipient; the Read operation transfers a message in the opposite direction. Both are one to one operations. The Broadcast operation transfers a message from initiator to PE's in the range except itself. This is a one to many operation. It is obvious that broadcast is more efficient than write.

The operation of the DS-Bus has an arbitration and a transfer phase. An arbitration phase is the period in which the arbiter resolves the conflicts and assigns a bus partition to requests. In a transfer phase, the communications granted during the arbitration phase take place. Arbitration and transfer phases can be performed either in sequence or in pipeline. The later should be chosen whenever possible.

The request signals asserted on the bus by a PE have a format as defined in Fig. 3(b).

The switches

The bus is segmented by N switching devices (S). There is no direct connection between any two bus segments. Two neighboring bus segments can be connected only if the switch between them is on. Since each PE is connected to the corresponding bus segment only, connection among PE's is the same as connection among bus segments, and therefore, is controlled by switches. In the extreme, all PE's can work separately (all switches are off); or all PE's are coupled tightly by one common bus (all switches are on). However, in most cases, these switches should configure the DS-Bus into multiple bus sections, to support concurrent communications. Thus, the switches establish and decouple paths among PE's as the communication requests change dynamically. Also, it is interesting to note that the DS-Bus is a message path, shared by multiple PE's. The physical structure of the path is similar with a conventional bus except for its segmented form. Thus the terminology: "Dynamically Segmented Bus".

The arbiter

The arbiter role is to make the switches intelligent enough to accomplish the functions specified above. It finds a set of non-conflicting requests and sets the switches accordingly. Since the arbiter is required to resolve multiple conflicts among multiple communication requests, it is necessary to examine the conflicts on the DS-Bus before discussing the arbiter any further.

2.1. DS-Bus contentions and resolution

On the DS-Bus, a PE requests a section of the bus in order to communicate with another PE or a group of PE's. A conflict exists in the case two bus section requests at the same time are overlapping. For instance, PE5 requests bus section from segment 2 to 7. If, at this time, PE8 requests bus section from segment 6 to 8, a conflict arises due to common segments 6 and 7. It is obvious that these two communications can not be accomplished in one transfer phase. Thus, some simultaneous requests may not be granted due to overlapping bus sections requested and should be deferred.

The contentions on the DS-Bus put constraints on the arbiter. In general, the constraints are:

(1) Multiple selections;
(2) Assignment of a bus partition to winners;
(3) Reasonable assignment;
(4) Fair arbitration.

The first constraint is fundamental. The DS-Bus can support multiple simultaneous communications only if the arbiter is capable of selecting multiple requests. The second emphasizes the condition of bus segment assignment: one bus segment can be assigned to only one PE in one resolution phase. This raises the two-side arbitration problem. Imagining the DS-Bus as a loop, the right side of a section is the side which can be reached from the left side counter-clockwise along the section. With the definition of Right and Left in mind, a request can conflict with other requests at both right and left sides because of the sectional nature of the requests. Both sides of a request, to be granted, can not overlap with the bus section or sections which have been granted.

Generally, the resolution can be accomplished as following. Let S be the union of all segments granted, R_i be the set that includes the segments which PE_i is requesting, R the set of all PE's that are requesting bus sections, but have not been granted yet. The order in R is significant. Assume R preserves the ordering of the following:

$$(s, s+1, \quad N, \quad 2, \ . \ , s-1).$$

Also, let H be a subset of R, such that for any i in H, the intersection of S and R_i is the empty set Q. Then,

R:=($P1, P2, \ldots, Pn$)
; PE's with numbers $P1$, $P2$, are requesting
H:=R
S:=R_k, $k = P1$
; choose s by criterion 1
Assign bus section R_k to PE_k
R:=$R - (k)$
While $H \neq Q$ do
Update H, Assign bus section R_j to PE_j
; choose j from H by criterion 2
S:=$S + R_j$
R:=$R - R_j$
End of while.

Criteria 1 and 2 above are not specified yet, and are subject to change to achieve optimal performance. They can be used together to meet constraints 3 and 4. However, for the sake of simplicity, criterion 1 is chosen to make the resolution fair. Criterion 2 is chosen to optimize the performance of the arbiter. Some of the policies for criterion 1 are:

(1) Fixed: let s:=C. Typically $C = 0$;
(2) Rotational: let s:=$(k + C)$ MOD N at the kth arbitration phase;
(3) Random: let s:=C_k at the kth arbitration phase. Where C and C_k are arbitrary integers between 0 and $N - 1$, where N is the number of PE's on the DS-Bus.

Some of the policies for criterion 2 are:

(1) Priority: let j:=max(P(**H**)). Where symbol P is the transformation from a vector to another vector if **H** is treated as a vector.
(2) Random: let j be any element in **H**;

However, there are two optimal criteria for the DS-Bus:

(1) Maximum utility criterion: choose a bus partition such that $|S|$ is maximized at the end of the resolution cycle (maximum DS-Bus utility).
(2) Maximum response criterion: choose a bus partition such that $|R|$ is maximized at the end of the resolution cycle (greatest number of PE's can be granted).

$|\bullet|$ is a operation taking the size of the set.

To illustrate the arbitration process, an iterative arbitration is described below as an example. In such a process, resolution may begin from one particular PE, say PE_8, which is assigned the highest priority by the arbiter. Then the process scans over the entire bus counter-clockwise in an iterative fashion. The following is an algorithm which formally defines steps that may be involved in resolution.

(1) Chose a starting point by criterion 1, and take the Left value and the Right value of the first request encountered during the process as the left boundary and the right boundary of the granted bus section. The granted bus section refers to the section on the DS-Bus which accommodates all requests granted so far. Go to step 4;
(2) Check the Left and Right: compare Left and Right of all requests, which have not been granted yet, against the right and left boundary of the granted bus section. Figure 4 shows various situations between requests and the granted bus section. In all these cases, any request with Right less than the left boundary and Left greater than the right boundary can be granted for communication. The only exception is in case 1 in which either Left is greater than the right boundary or Right is less than the left boundary. This also satisfies the requirement for grant.
(3) Resolution: Adopt a criterion to select one request from those that are successful in step 2, and extend the granted bus section to include the bus section just resolved. If no matching is found in step 2, go to step 5;
(4) Set the setting register: set the bits of the setting register within the bus section just resolved except on the two sides. The information stored in the setting register defines the switch setting for the next transfer phase. If a bit is set, the corresponding switch will be on. Repeat from step 2;
(5) End of resolution phase.

2.2. Arbiter design

It is clear that the performance of the arbiter can greatly influence the performance of the system. Serial comparison involved in resolution may considerably increase the period of the bus cycle even though it can be partially overlapped with data transfer.

There are several trade-offs available during the design process. The main options are:

(1) Central or distributed organization;
(2) Synchronous or asynchronous access;
(3) Fixed priority, dynamic priority, or optimal policy.

Accordingly, there are 12 distinct kinds of arbiters. Because of the complexity and difficulty in layout of the arbiter, distributing it into many modules is a good organization to consider. Each granted request can be carried out as soon as the requested bus section is free and the beginning of a transfer activates the resolution to the bus section in asynchronous access mode. This requires an even more complex arbiter while asynchronous access may not be necessary for optimal performance. Once again, criterion 2 mentioned in Section 2.1 is critical in the trade-off between complexity and performance of the arbiter.

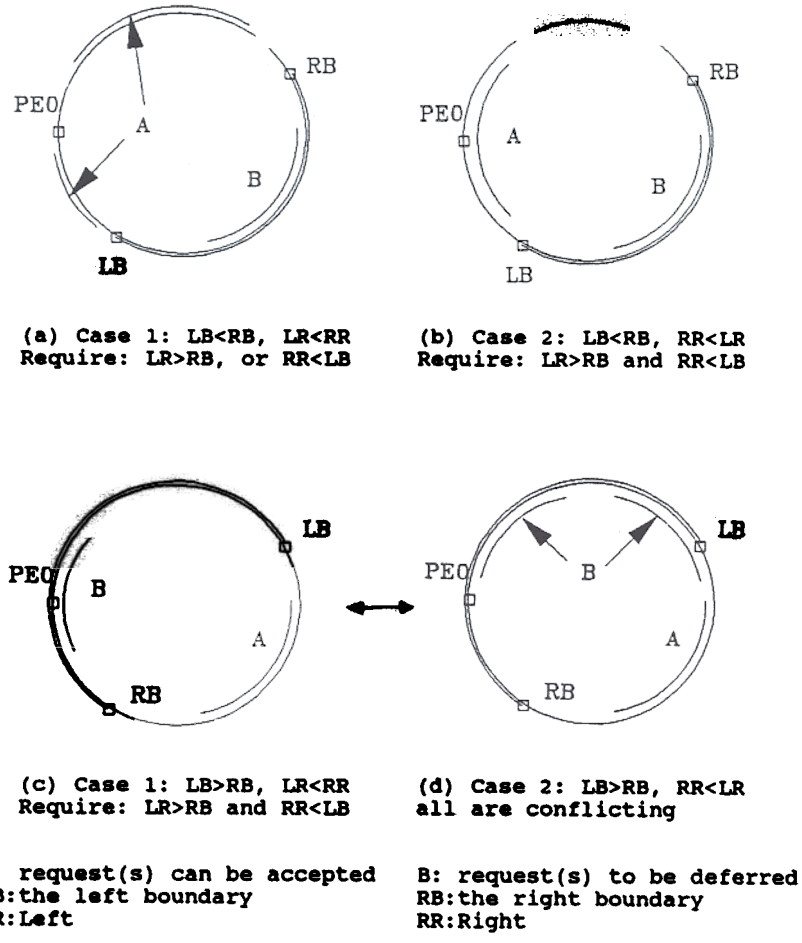
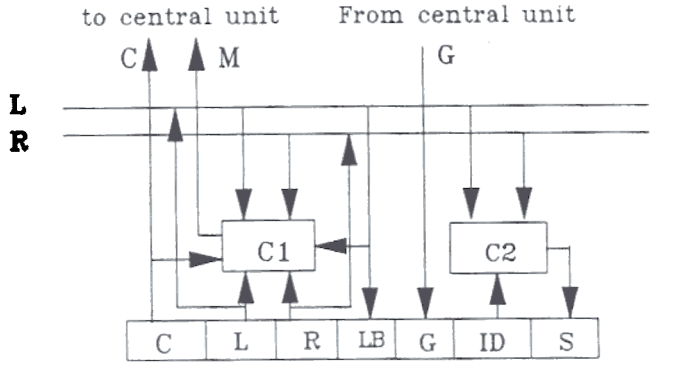


Fig. 4. Variant cases of requests and the granted Bus section.

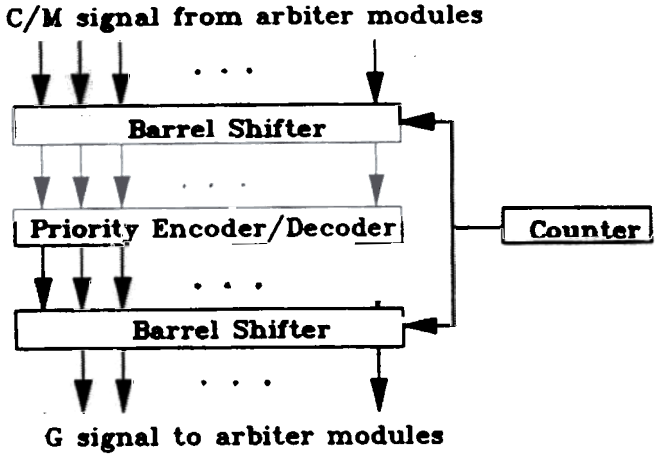
Instead of describing each kind of arbiter in detail, one practical implementation of the arbiter is proposed. Figure 5 shows a particular distributed design in synchronous access mode. The resolution process of the arbiter follows the algorithm described in the previous section. In particular, rotation policy is adopted for criterion 1; priority policy for criterion 2.

The arbiter in Fig. 5 is distributed into N individual arbiter modules, one for each PE, and a central control unit. Each module consists of C, L, R, LB, ID, G, and S registers as in Fig. 5(a). They are used to store request (C), Left (L), Right (R), left boundary (LB), ordering number (ID), Grant (G), and Switch setting (S). There are two comparators: C_1 for resolution compare, C_2 for switch setting compare. The control unit consists of a counter and a priority logic.

When an arbitration cycle starts, the control unit first asserts a G signal to the arbiter module with highest request priority. Module s is chosen in a rotational fashion. In this case, the specialization of criterion 1 can be easily implemented by a counter in the control unit. The first G signal can be generated from the C signal as illustrated in Fig. 5(b). At the next clock cycle, the module, which has received a G signal at the last cycle, places the Left and Right on the L/R fields for both resolution and switch setting compare. The resolution compare generates a matching signal (M in Fig. 5) at each C_1 in a way defined in step 2 of the resolution algorithm by using its R and L; the left (in its LB register) and right boundary (the value of R on ths bus is exactly the same as the right boundary). The value of the left boundary is latched into the LB registers by a signal asserted by the control unit (omitted in Fig. 5). The M signals asserted by modules are led to the control unit. In the control unit, the M signals are processed by a barrel shifter, a priority encoder, a decoder, and another barrel shifter (opposite direction). Finally, a G signal is generated and fed back to the proper arbiter module and stored in the G register. The corresponding PE will gain a section of the DS-Bus in the next transfer phase. In the meantime, each module compares



(a)



(b)

Fig. 5. Arbiter design (register level). (a) The Arbiter module, (b) the central unit.

its ID with values from L and R fields at its C_2. When $R > ID \geqslant L$, the corresponding S register is set; otherwise, there is no response. At the end of the resolution phase, the contents of the S registers determine the switch setting for the next transfer phase, and so on.

This implementation has a non-deterministic maximum resolution time, which is linearly proportional to the number of grants. Each grant, except the first, takes one clock cycle. In each cycle, the longest path is releasing the R and L on the DS-Bus, resolution comparison and priority selection. If each operation takes 3Δ, where Δ is a gate delay, the minimum period of the clock is 9Δ. The total resolution time would be approximately $9\Delta \cdot M$, where M is the number of grants. Since the resolution time depends heavily on the number of grants, the time interval for a bus cycle may be determined on a cycle-by-cycle basis.

3. DS-BUS MODELING

In order to study the performance of the DS-Bus, analytic models will be developed while computer simulation is employed as reference for comparison.

3.1. The basic model

The following conditions are assumed for the basic model:

(1) PE's generate independent communication requests;
(2) Requests at any bus cycle are independent of requests at previous bus cycles;
(3) Request rates are uniform for all PE's and have a value r;

(4) Compared to a bus section requested by a PE, the total length of the DS-Bus is much larger.
(5) Each communication requests a bus section of length $L + 1$, which is equally distributed on either side of each PE.

With these assumptions, it is safe to study a particular PE rather than all PE's on the DS-Bus at the same time. (In the appendix the constraint in 5 is removed.) As an example, if PE_i requests communication (Fig. 6), and a request within distance (defined as the smaller number of bus segments separating the two PE's) L on the left side of PE_i (the range is indicated by FR) has been accepted by the arbiter, then the current request will be deferred until the next resolution cycle [Fig. 6(b)]. On the other hand, if there is no request in FR, or requests in FR are deferred due to conflicts, the request will be granted. Requests more than a distance L from PE_i have no influence on the current request [see Fig. 6(a)].

If Pr denotes the probability of accepting a request (accept rate), then the event that a PE in FR will access the bus has a probability of $r \cdot Pr$. Altogether there are L such probabilities. One such occurrence will deny the request of PE_i. Thus

$$Pr = 1 - L \cdot r \cdot Pr \tag{1}$$

is the probability of the DS-Bus accepting the request. Rearranging equation (1)

$$Pr = \frac{1}{1 + L \cdot r}. \tag{2}$$

Equation (2) explicitly describes the accept rate for each PE. This is then the basic model. Figure 7 shows the curve of Pr against r for $L = 2, 4, 8$ and $N = 32$. As expected, the influence of the requested bus section size upon the accept rate is rather strong. The Pr decreases very sharply as L increases. In Fig. 8, a comparison is made for equation (2) against simulation results derived under the same assumptions. Surprisingly, the analytic model of Pr fits the simulation results quite well although it does not take the total length of the DS-Bus into account. The simulation results in Fig. 8 show that the accept rate is insensitive to the number of PE's when N exceeds 10 for $L = 4$. This helps clarify assumption 4. By "much larger", the assumption really says that the length effect can be neglected if the DS-Bus is about three times as long as the bus sections requested.

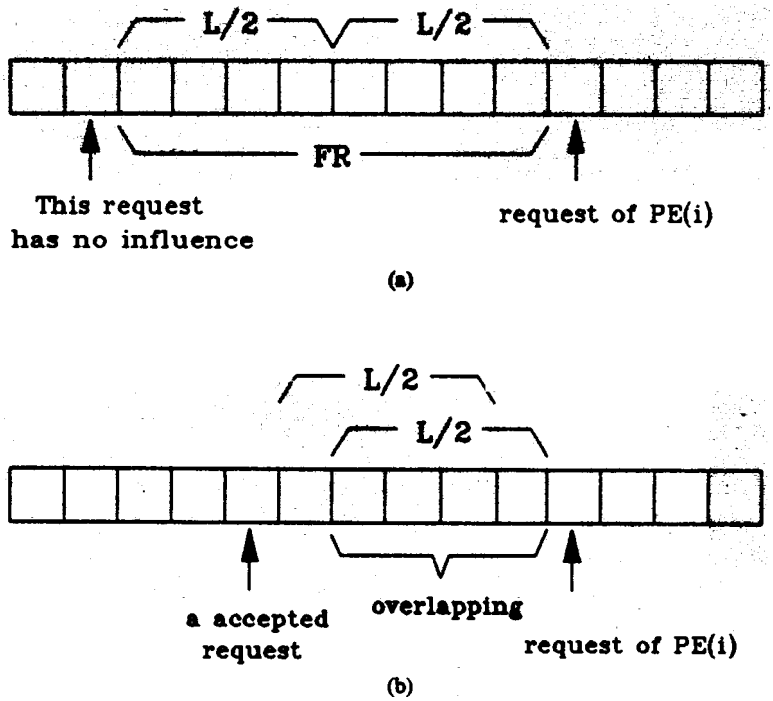


Fig. 6. Analysis of acceptance.

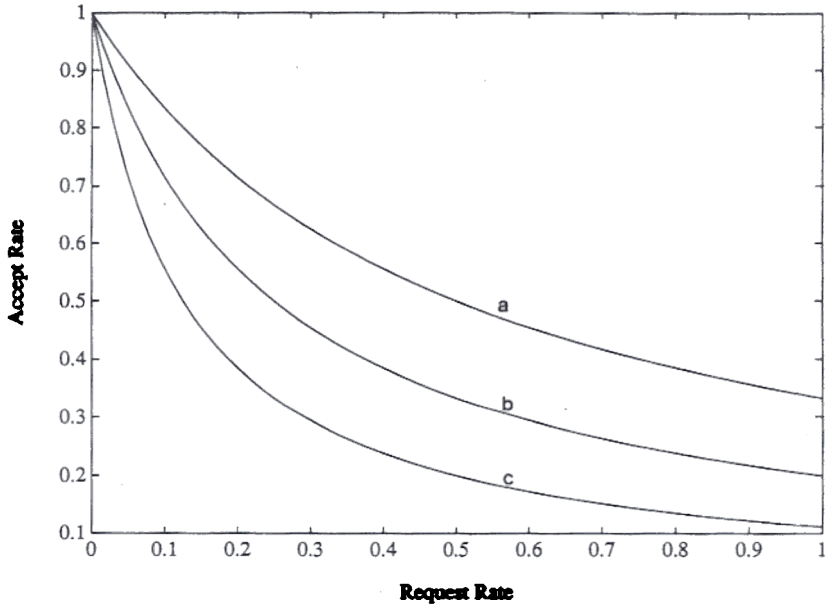


Fig. 7. Influence of L on Pr. (a) $L = 2$, (b) $L = 4$ and (c) $L = 8$.

When the request rate r is large, especially when it is greater than $1/(1 + L)$ (capacity of the DS-Bus), the dependence between two successive bus cycles can not be ignored. In other words, the basic model will produce large errors in case of relatively large r.

3.2. Relaxed models

If assumption 2 is relaxed, the request rate r in equation (2) should be modified because some requests were deferred from earlier bus cycles. Now $r(i)$ is employed to denote the instant request rate, defined as the ratio of the number of requests to the number of PE's at the ith bus cycle. The changes in $r(i)$ over time are:

At first cycle, $r(0) = r$;

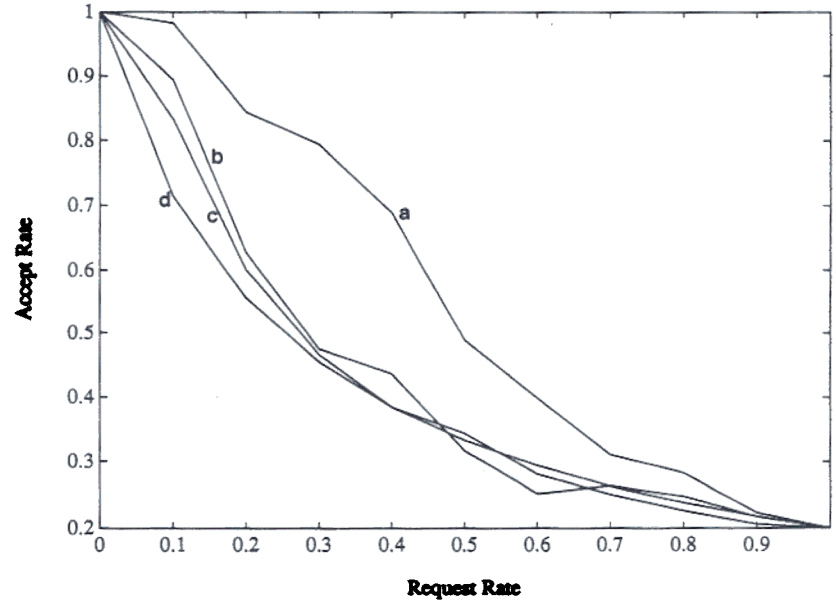


Fig. 8. Simulation with different number of processing elements (N) and comparison with equation (2). (a) $N = 5$, (b) $N = 10$, (c) $N = 20$ and (d) equation (2).

at second cycle,

$$r(1) = M(((1 - Pr(0)) \cdot r \cdot N + r \cdot N)/N)$$
$$= M(r + r(0) \cdot (1 - Pr(0)));$$

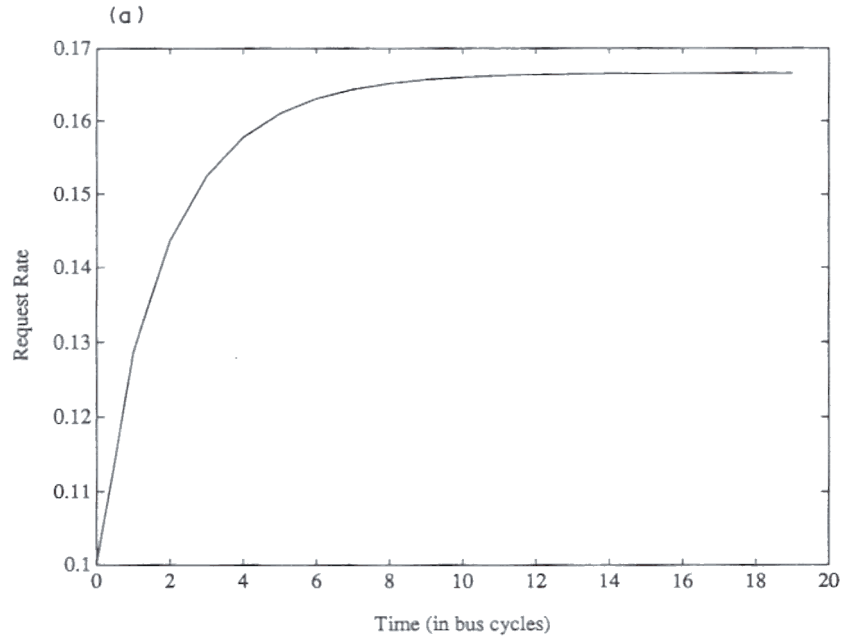
... ...

at $(k + 1)$th cycle,

$$r(k) = M(r + r(k - 1) \cdot (1 - Pr(k - 1))); \quad (3)$$

where

$$M(x) = \begin{cases} x & \text{for } 0 \leqslant x < 1 \\ 1 & \text{for } x \geqslant 1 \end{cases}$$



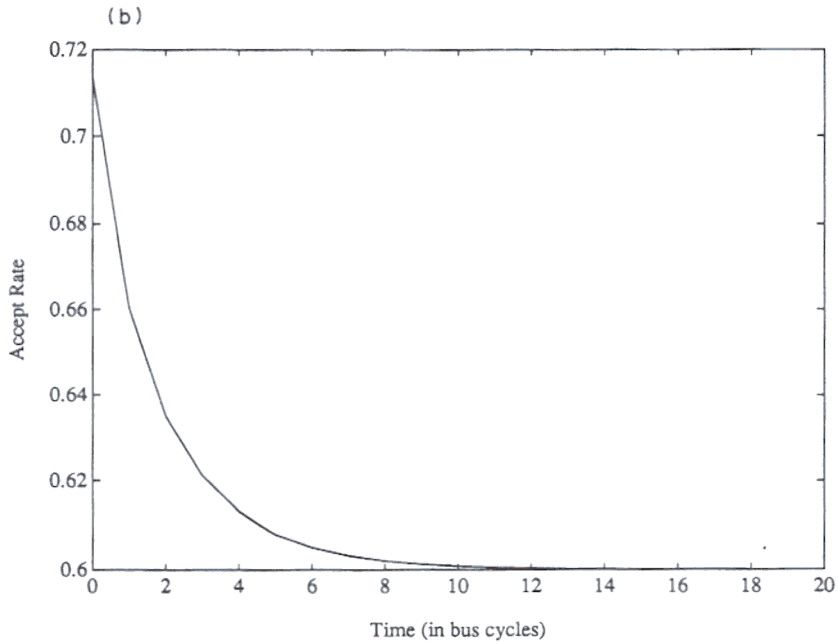


Fig. 9. (a,b) *Caption on facing page.*

and $Pr(i)$ is the corresponding accept rate for $r(i)$. For the modified r, the basic model is still valid, i.e.

$$Pr(k) = \frac{1}{1 + L \cdot r(k)}. \tag{4}$$

Equations (3) and (4) describe the dynamic behavior of the DS-Bus recursively. Figure 9 shows how $r(k)$ and $Pr(k)$ change for $r = 0.1$ and $r = 0.2$. Figure 9(b) clearly shows that $r(k)$ goes up to 1.0 and $Pr(k)$ drops to 0.2 when $r = 0.2$ [equals exactly $1/(1 + L)$].

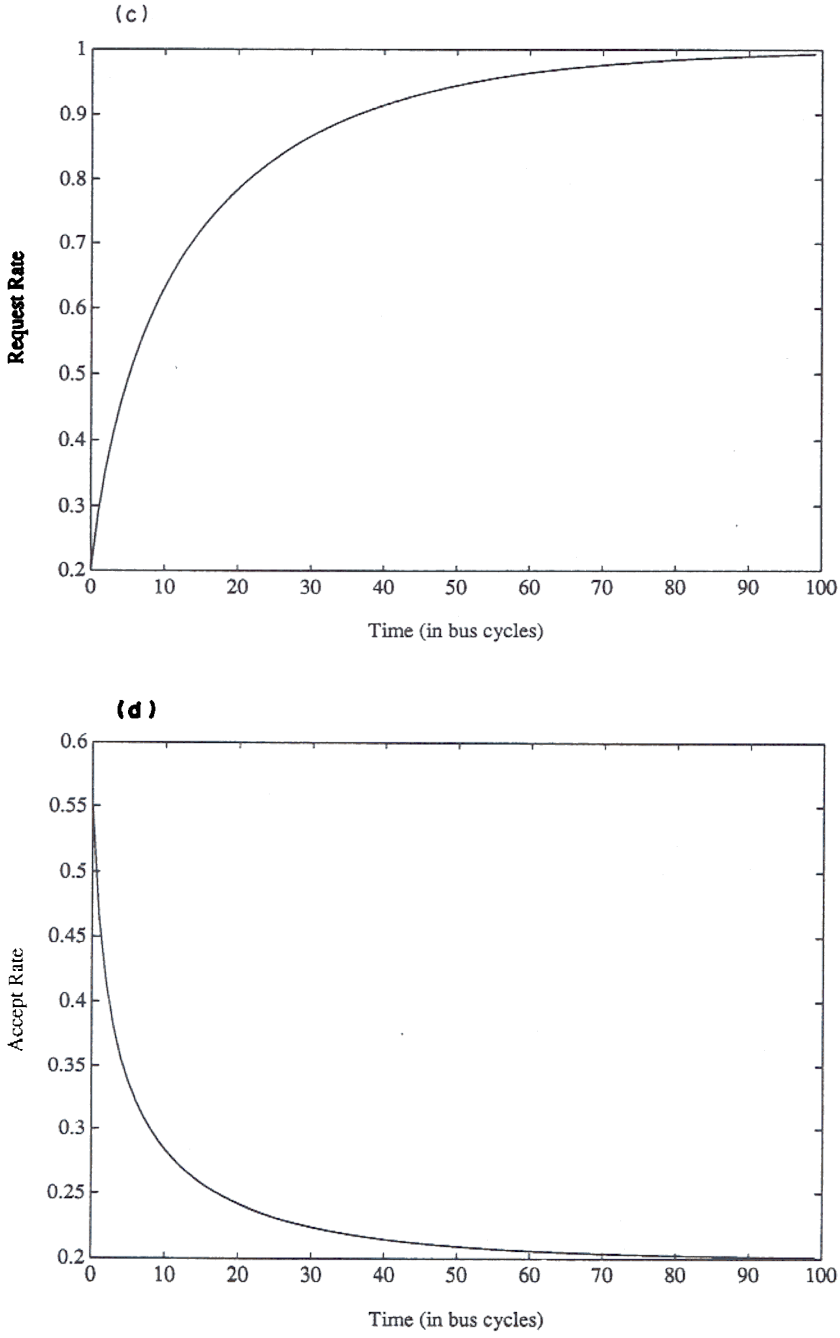


Fig. 9 (c and d).

Fig. 9. Request and accept rate versus time. (a) Request rate increases with time ($r = 0.1$), (b) accept rate decreases with time ($r = 0.1$), (c) request rate increases with time ($r = 0.2$) and (d) accept rate decreases with time ($r = 0.2$).

In the static state, each $Pr(i)$ in equation (3) should be the same for the finite terms with lower order. Ps is used for $Pr(i)$, Rs for $r(i)$. Thus

$$\begin{aligned} Rs &= [r(k);\ k \to \infty] \\ &= M(r \cdot (1 + (1 - Ps) + (1 - Ps)^2 + (1 - Ps)^3 + \cdots)) \\ &= M\left(\frac{r}{1 - (1 - Ps)}\right) = M(r/Ps) \end{aligned} \tag{5}$$

and

$$Ps = \frac{1}{1 + L \cdot M(r/Ps)}$$

i.e.

$$Ps = \begin{cases} 1 - L \cdot r & \text{for } r \leqslant Ps; \\ \dfrac{1}{1+L} & \text{for } r > Ps. \end{cases} \tag{6}$$

The curve of Ps is shown in Fig. 10, this is quite different from that of equation (2) shown in Fig. 8 (redrawn in Fig.10 for comparison). Ps drops much faster than Pr while Pr degrades smoothly as r increases. The largest error occurs at $r = 0.2$. However, both tend to 0.2 (minimum value of the accept rate for $L = 4$).

3.3.Model discussion

Ps is the static value of the accept rate and is also equal to 0.2 for a requested bus section of 5 and the request rate r is equal to 0.2. Since the dynamic request rate $r(i)$ is approaching 1 (Fig. 9), the accept rate must be 0.2 although requests experience long delays. This example clearly illustrates that accept rate alone is not sufficient to describe the DS-Bus.

The region to the left of C is called W, and the region to the right of C is called S (see Fig. 10). Since the DS-Bus usually works in the region W, it is convenient to use the mean request interval instead of the request rate. For instance, the relationship between the request interval and the accept rate in the static state can be derived from equation (6)

$$Ps = \begin{cases} 1 - L/c, & \text{for } c \geqslant 1 + L; \\ 1/(1 + L), & \text{for } c < 1 + L. \end{cases} \tag{7}$$

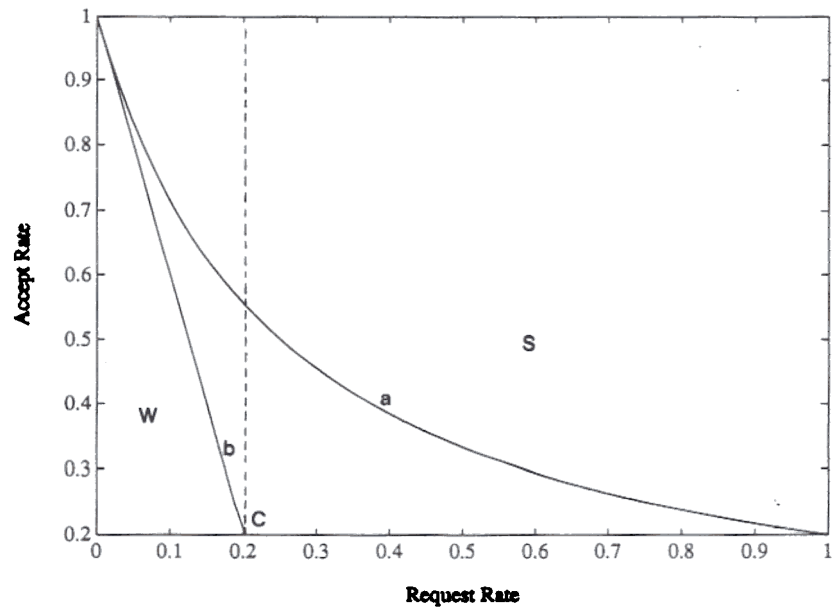


Fig. 10. A relaxed model. (a) From equation (2) and (b) the relaxed model from equation (6).

where c is used to denote the request interval. Figure 11 shows the accept rate against the request interval of equation (7). Simulation results are also plotted in the same figure. The experiments were done for both uniform and Poisson distributions of the request interval. One conclusion drawn from Fig. 11, is that equations (6) and (7) are quite accurate estimates of the accept rate. In some cases (usually when requests have large variance), it overestimates the accept rate. In other cases (usually when requests have small variance), it underestimates. The error in Fig. 11 is tolerable. Models described by equations (4), (6), and (7) can be used in most cases without undue risk.

Special attention should be paid to point C in Fig. 10. The value at C is the capacity of the DS-Bus (denoted by CDSB). Physically, the capacity of the DS-Bus is the maximum request rate at which no bus saturation occurs. With $Ps = r$, the following formula can be derived from equation (6)

$$\text{CDSB} = \frac{1}{\quad} \tag{8}$$

When the request rate r goes beyond CDSB, the accept rate remains the same. This is an indication of bus saturation caused by mismatch between the accept and request rates. In this state, buffers between the DS-Bus and the PE's accumulate the requests from the processing element. The buffers may be filled up if requests in the buffer are not accepted by the DS-Bus at a sufficiently high rate. Eventually the PE's slow down because of long delay of messages needed during computation, and the system performance is degraded. Ideally, an application should not drive the DS-Bus into the region beyond, or too close to the capacity of the DS-Bus.

3.4. Delay and bandwidth

Delay, or waiting time, is the mean time difference between the instance at which a communication request is submitted and the instance at which the request is accepted. Obviously, the delay should be as small as possible.

A closer look at a buffer between the DS-Bus and a PE is in order. Each buffer is very much like a service queue model (see Fig. 12). Assuming that the request is Poisson-distributed, the service time distribution is unknown at this point but can be safely assumed to be exponentially distributed. An exponential distribution has large variance, therefore, this assumption is justified. Thus, the buffer behaves like an $M/M/1$ queue. For an $M/M/1$ queue with customer arrival rate γ and service

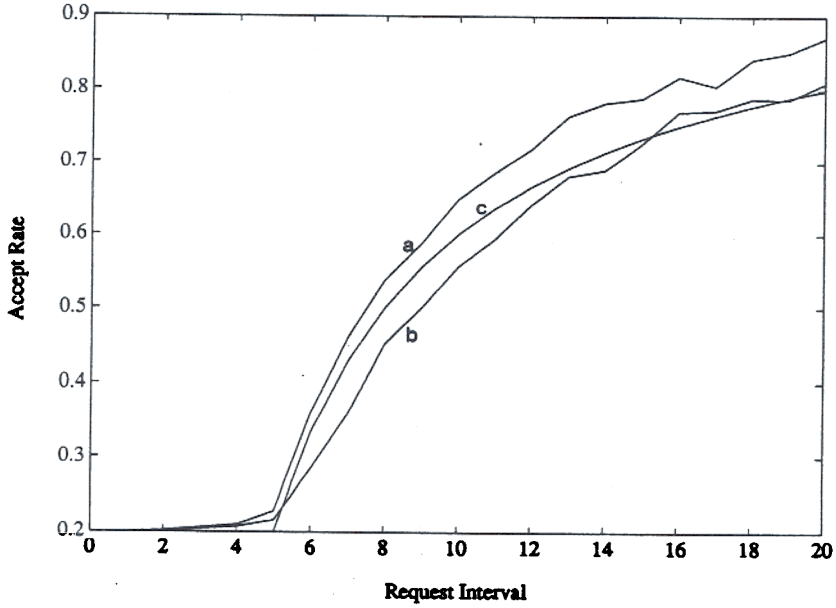


Fig. 11. Comparison of simulation with equation (7). (a) Uniform distribution, (b) poisson distribution and (c) equation (7).

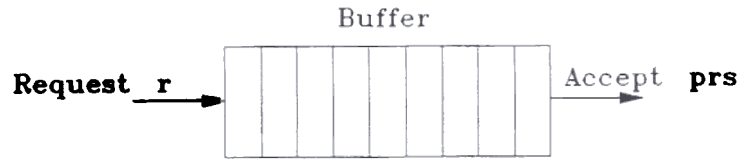


Fig. 12. A buffer considered as an $M/M/1$ queue

rate μ the average number of customers $E(q)$ in the queue is

$$E(q) = \frac{\gamma^2}{\mu^2 - \gamma \cdot \mu}. \tag{9}$$

Thus, a buffer has mean requests waiting for acceptance

$$E(b) = \frac{r^2}{Ps \cdot (Ps - r)}.$$

Using Little's formula, $L = \gamma \cdot W$, the delay can be written as

$$d = E(b)/r = \frac{r}{Ps \cdot (Ps - r)}.$$

Substituting r with c, and Ps with equation (7), equation (11) becomes

$$d = \frac{c}{(c - L)(c - L - 1)} \quad \text{for} \quad c \geqslant 1 + L.$$

This is illustrated in Fig. 13 for $L = 4$, 5, 6 and 7.

As an example, consider PE's to be implemented with 8086-8087 pairs. When the PE's are executing the following loop

```
DO I = 1 to N
  ADD AX, [BX + DI]; operation
  OUT [DX], AX; output to buffer
END DO
```

they can generate the highest reasonable output rate. If DX points to communication buffers, they generate a shortest communication interval of $(8 + 9) + 8 = 25$ clock cycles. This interval

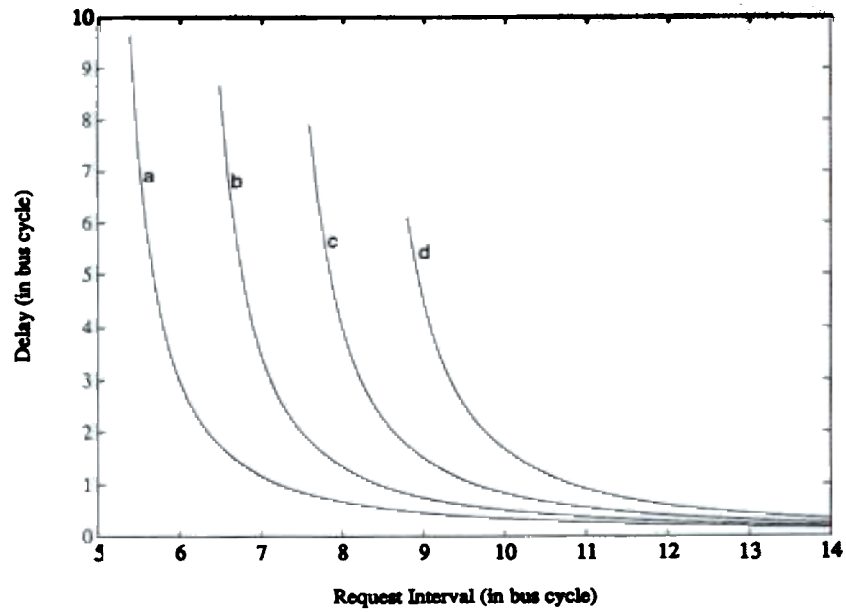


Fig. 13. Delay (d) vs request interval (c). (a) $L = 4$, (b) $L = 5$, (c) $L = 6$ and (d) $L = 7$.

corresponds to 6.25 bus cycles for DS-Buses with four clock cycles per bus cycle. The delay inserted by a DS-Bus is about 2 bus cycles (see Fig. 13). At this point, the system speed-up at maximum request rate can be determined as

$$SU_i = N \cdot 25/(25 + 8 + 4) = 0.67 \cdot N.$$

This SU_i is for integer computation. If an application deals with floating point numerical computation, the shortest request interval corresponding to the above loop would be much longer. The ADD instruction takes about 84 clock cycles. If load and store time is neglected, the request interval is 21 bus cycles. The bus delay (from Fig. 13) is less than 0.5 bus cycle. The speed-up in this case is

$$SU_f = N \cdot 21/(21 + 0.5 + 2) = 0.9 \cdot N$$

This is a very promising result.

Bandwidth (BW) is defined as the average number of requests accepted in each bus cycle, i.e. it is a measure of the concurrent degree of the bus. BW can be derived from equation (7) as follows

$$BW = Rs \cdot N \cdot Ps = N \cdot M(r/Ps) \cdot Ps$$

i.e.

$$BW = N/c \quad \text{for} \quad c \geqslant 1 + L. \tag{13}$$

For $L = 4$, $c = 6$ and $N = 32$, BW equals 5.3. Equation (13) is relevant when trying to compare the DS-Bus scheme with other schemes during evaluation. One contradiction should be noticed. In order to make maximum use of the DS-Bus, c in equation (13) should be decreased. However, this is contrary to the demand of equation (12) which says that, the larger c, the smaller d. Biasing towards equation (12) is necessary during system design, tuning and application.

4. EVALUATION OF THE DS-BUS ARCHITECTURE

The DS-Bus and multiple bus networks evolved from bus structures. On the other hand, a DS-Bus works as interprocessor networks similar to a hyper-cube. Therefore, multiple bus and hyper-cube networks will be used for comparison to evaluate a DS-Bus network.

From a hardware point of view, there are five commonly used parameters: Complexity, Modularity, Expandability, Fault Tolerance and Reconfiguration. Complexity is a measure of cost; Modularity the ability to construct the system by repetition; Expandability the easiness to expand the system; Fault tolerance a measure of the performance in case a fault occurs; Reconfiguration the flexibility of the system to adapt to changes of demand. The attributes of the three types of networks to these parameters are listed in Table 1. The DS-Bus is favorable on all aspects except fault tolerance due to the central control logic and the segmented bus.

Performance is the most important parameter for evaluation of a network. The performance of a network usually is application dependent. For proper evaluation a particular type of work-load must be chosen.

Consider first the summation of N numbers $A(0 \ldots N-1)$. Assume there are N PE's and that $A(i)$ is already stored in the local memory of the ith PE, and $N = 2^n$ for some n. The summing can be done by sending $A(2k-1)$ to the $2k$th PE and summing $A(2k-1)$ and $A(2k)$ in the $2k$th PE, for $k = 1, \ldots, N/2$ in the first cycle. At the ith cycle, the $2^{i-1}(2k-1)$th PE sends the partial sum to the $2^i k$th PE, where it forms a new partial sum (for $k = 1, \ldots, N/2^i$). Finally, the summation is formed by the Nth PE in the $(\log_2 N)$th cycle. During the process, there are no conflicting requests submitted. Thus, there is no waiting involved during summation.

Table 1. Comparison of hardware for various networks

	Multiple bus	Hyper-cube	DS-Bus
Complexity			
Modularity			
Expandability			
Fault tolerance			
Reconfiguration			

As a second example, consider the solution of a system of linear equations. The system is of the form [A] × [X] = [B], where [A] is an $N \times N$ banded matrix, with a bandwidth of $L + 1$, and [B] and [X] are vectors. There are two methods to solve this problem: direct and iterative methods. Iterative methods seem to be more efficient, in particular, for large sparse matrices and on MIMD computers [3,4]. Thus, the classical Jacobi method is considered here by mapping the ith equation to the ith processor. Under these conditions, most processors carry out the computation in the following form

$$x(i) = \sum_{k=i-l}^{i+l} a(i,k) \cdot x(k),$$

where l is the half bandwidth of the system (i.e. $L = 21$). In this case, the request interval will be approximately $96 \times 5 \div 4 = 120$ bus cycles. The delay at this request interval is negligible compared to the computation time (see Fig. 13).

However, if an FFT program is going to be partitioned and executed on a DS-Bus machine, the system would not perform so well.

In the following, a DS-Bus will be compared with multiple bus and hyper-cube in terms of bandwidth and delay by characterizing workloads with request rate of PE's.

Assume that the number of processors N is 64 for all three cases. The number of memory banks M is 32, the number of buses B is either 7 or 8 for the multiple bus case. The hyper-cube is a direct binary network. Also, the reference pattern is assumed to be harmonic so that it is relevant to the characteristics of the DS-Bus. Harmonic locality defines a reference pattern such that a PE generates a message of hopcount i with a probability inversely proportional to i. Figures 14 and 15 show the bandwidth and delay [5,6].

Although the hyper-cube is capable of supporting any value of request rate, the penalty is long message delays and a costly system. On the other hand, the multiple buses yield much shorter delays if they are not saturated. The maximum request rate is 0.11 for $B = 7$ and 0.13 for $B = 8$. The figures show that the DS-Bus has a bandwidth between $B = 7$ and $B = 8$ of the multiple buses, and that the delay is between that of the multiple buses and that of the hyper-cube when the request rate is below 0.095. Most of the time, it is close to the multiple buses. Recall that 0.16 is the highest request rate for integer operation, and 0.04 the highest request rate for real operation. If two operations are equally likely to happen, the highest rate would be 0.1. Thus, by a specially structured "bus", the DS-Bus is a good compromise between cost and performance.

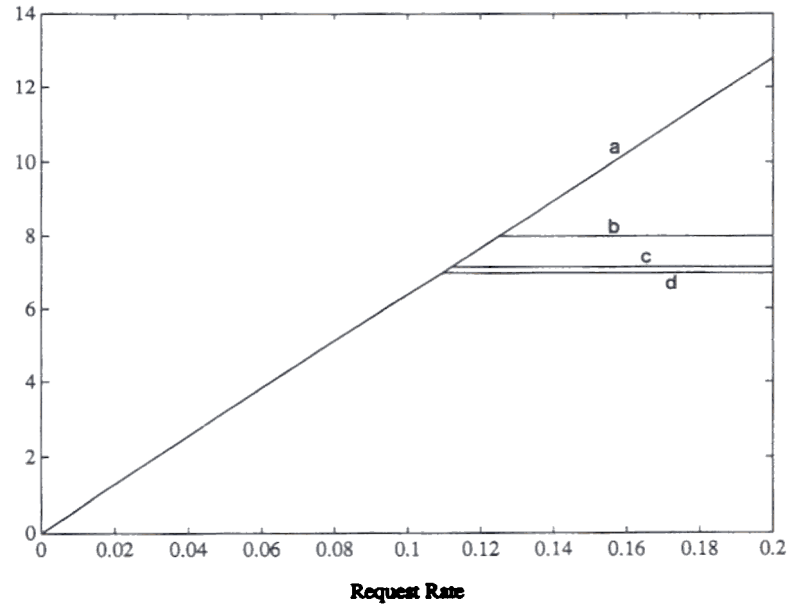


Fig. 14. Comparison of Bandwidth on various networks. (a) Hyper cube, (b) multiple bus ($B = 8$), (c) DS-Bus and (d) multiple bus ($B = 7$).

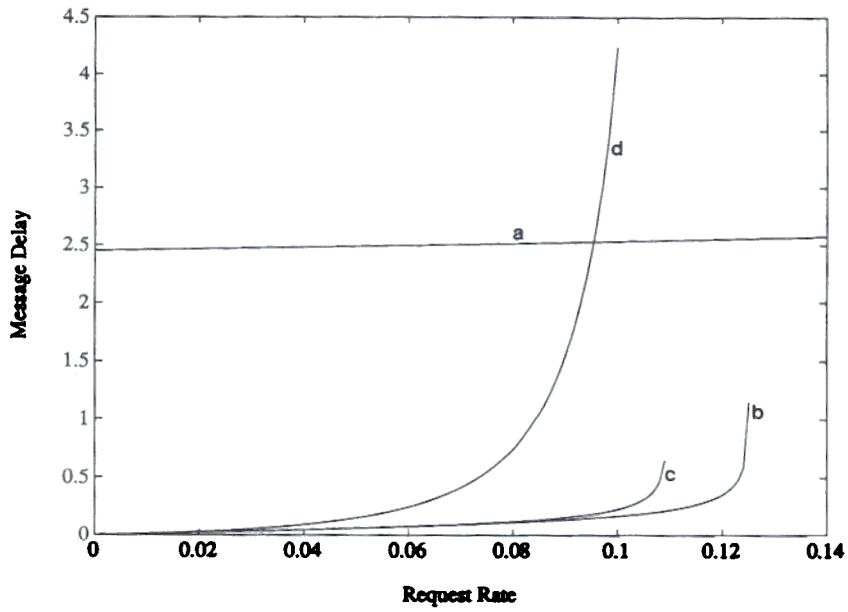


Fig. 15. Comparison of message delay on various networks. (a) Hyper-cube, (b) multiple bus ($B = 8$), (c) multiple bus ($B = 7$) and (d) DS-Bus.

5. AN EXAMPLE OF THE DS-BUS ARCHITECTURE

Figure 16 is an example of a DS-Bus architecture. It illustrates a network implemented with 66 individual DS-Buses. Each DS-Bus is connected with others through two DS-Buses. If one DS-Bus can support 64 processors, then a total of 3904 processing elements (excluding the three communication processors (CP) per DS-Bus) can be accommodated in this system. All communication processors are connected to the front-end processor by a common bus.

Previous analyses indicate that PE's should be implemented with microprocessors of moderate performance with respect to the bus speed. Otherwise, mismatching between computing power and communication resource causes poor utilization of system hardware and results in low speed up. Thus, standard microprocessors (with numerical co-processor) would be a good choice to implement PE's in the system.

Implementing the interconnection network by a set of DS-Buses, instead of one DS-Bus has two advantages: recall that the resolution time is linearly proportional to the number of grants (Section 2.2). This implies that the shorter a DS-Bus, the shorter the resolution time. Consequently, the design given in Section 2.2 can be implemented in practice. Secondly, the requested bus section by read and write will be effectively cut down. The longest requested bus section in such a system is 80 bus segments. For a single DS-Bus system with same number of PE's the longest requested bus section is 2k.

The connection between DS-Buses is accomplished by two additional DS-Buses (called global DS-Bus), as well as the common PE's on neighboring DS-Buses which should greatly improve the communication of the neighboring DS-Buses. CP's on each DS-Bus are located uniformly around DS-Buses, 32 bus segments apart from each other, and such that two are on neighboring DS-Buses, the other two on global DS-Buses.

The front-end processor is mainly for interfacing between the system and the outside world, and/or for partitioning and assigning jobs. It connects with all CP's through a common bus, and, therefore, may access the rest of the system via the CP's. Notice the configuration flexibility of the DS-Bus architecture. This can be illustrated in two aspects. From a hardware point of view, the PE modules can be replaced with other modules, such as memory or I/O modules, that is, the system can be easily reconfigured. On the other hand, the front-end processor can treat PE's in the system like a pool of resources. PE's can be individually used for small scale computation, can be used set by set for larger scale computation, or all PE's can be assigned to a very large scale engineering problem. Furthermore, each PE is capable of sharing workloads, such as housekeeping, management, scheduling, and even I/O operations, with the front-end processor. Therefore, the

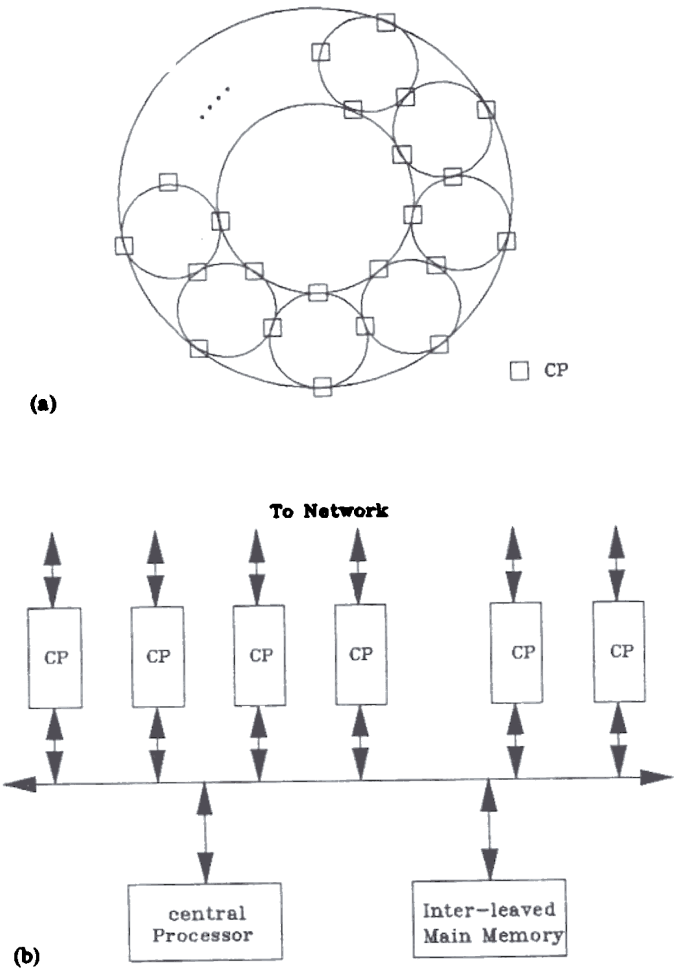


Fig. 16. An example of DS-Bus architecture. (a) Interconnection network and (b) the front-end processor and CP's.

front-end processor does not necessarily have to be a fast computer, and may be omitted from the system by replacing it with any PE.

6. CONCLUSIONS

A number of aspects of the DS-Bus architecture have been studied in some detail. The study leads to the following conclusions:

(1) The DS-Bus offers another approach for interconnection networks in parallel computer systems. The architecture provides a shared message path to combine short delay in a multiprocessor system with low communication demand in a distributed system.
(2) The DS-Bus gains capacity by using the bus efficiently. Since the accept rate and delay are degraded quite rapidly as the requested bus section increases, the way to make use of the potential of the DS-Bus is to limit communication between processing elements which are far from each other. This implies that the performance of the DS-Bus can be application sensitive.
(3) The arbiter is the most sophisticated module in the DS-Bus architecture. The resolution time of the current design is linearly proportional to the number of grants. In order to broaden the application and accommodate more processing elements on one DS-Bus, the design needs further improvements.
(4) Flexibility is another advantage of the DS-Bus architecture. This, seems, can be explored to overcome the bound on both computation and I/O.

REFERENCES

1. J. Archibald and J. L. Baer, Cache coherence protocols: evaluation using a multiprocessor simulation model. *ACM Trans Computer Syst.* pp. 273–298, Nov. (1986).
2. G. Pfister *et al.*, The IBM research parallel prototype (RP3): introduction and architecture, *Proc. 1985 Int. Conf. on Parallel Processing* (1985).
3. R. J. Melosh and Senol Utku, Direct finite element equation solving algorithms. *Computers Struct.* **20**, 99–105 (1985).
4. O. Axelsson, A class of iterative methods for finite element equations. *Computer Meth. Appl. Mech. Engng*, pp. 123–137, No. 9 (1976).
5. T. N. Mudge, J. P. Hayes *et al.*, Analysis of multiple-bus interconnection networks. *J. Parallel Distrib. Computing* **3**, 328–343 (1986).
6. S. Abraham and K. Padmanabhan, Performance of direct binary n-cube network for multiprocessors. *IEEE Trans. Comput.* **38**, 1000–1011 (1989).
7. A. Gottlieb *et al.*, The NYU Ultracomputer—designing an MIMD shared-memory parallel computer. *IEEE Trans. Comput.* **C-32**, 175–189 (1983).
8. T. Lang and M. Valero, M-users B-servers arbiter for multiple-buses multiprocessors. *Microprocessing Microprogr.* **10**, 11–18 (1982).
9. H. F. Jordan and P. L. Sawyer, A Multi-processor system for finite element structural analysis. *Computer Struct.* **10**, 21–29 (1979).
10. T. N. Mudge, J. P. Hayes and D. C. Winsor, Multiple bus architectures. *Computer*, pp. 42–48, June (1987).
11. D. A. Reed and D. C. Grunwold, The performance of multicomputer interconnection networks. *Computer*, pp. 63–73, June (1987).
12. K. Hwang and F. A. Briggs, *Computer Architecture and Parallel Processing*. McGraw-Hill, New York (1984).
13. M. Schwartz, *Telecommunication Network: Protocols, Modeling and Analysis*. Addison-Wesley, Reading, Mass. (1987).
14. P. J. Denning, The operational analysis of queueing network models. *Computing Survey* **10**, No. 3 (1978).
15. J. M. Kurtzberg, On the memory conflict problem in multiprocessor system. *IEEE Trans. Comput.* **C-23**, No. 3 (1974).
16. Kwok-Tung Fung and H. C. Torng, On the analysis of memory conflicts and bus contentions in a multiple-microprocessor system. *IEEE Trans. Comput.* **C-27**, No. 1 (1979).
17. H. S. Stone, *High-Performance Computer Architecture*. Addison-Wesley, Reading, Mass. (1987).
18. H. J. Siegel, *Interconnection Networks for Large Scale Parallel Processing: Theory and Case Studies*. Lexington, Mass. (1985).
19. G. A. Anderson and E. D. Jensen, Computer interconnection structures: taxonomy, characteristic, and examples. *ACM Comput. Surv.* **7**, No. 4 (1975).

APPENDIX

It is possible to derive a formula for an accept rate in case of varying L if some approximations are allowed. To do this, Fig. 6 is used again. Assuming that, at a particular bus cycle, the ith PE is requesting a bus section of length $L(i)+1$ with a probability of r. Then, in order to accept this request, possible requests, at a distance $L(i)/2$ or less from the PE on the left side, should be deferred. This event has a probability of $(1-r\cdot Pm)^{L(i)/2}$ provided that Pm is the mean accept rate and allowing substitution of individual acceptance Pa with Pm. Furthermore, outside of the region above on the left side, from right to left, the first possible request should be deferred; the second possible request may access the DS-Bus with section length of 3 bus segments or should be deferred; ...; the kth possible request may access a section of length $2k-1$ or less, or should be deferred, and so on. This event has a probability of

$$\prod_{k=0}^{\max}[(1-r\cdot Pm)+r\cdot Pm\cdot P_k] \tag{14}$$

where P_k is the probability of the requested bus section being of length $2k+1$ or less. Keeping terms up to the order $r\cdot Pm\cdot P_k$, we expand equation (14) to get

$$1-r\cdot Pm\sum_{k=0}^{\max}1-P_k=1-r\cdot Pm\cdot Lm/2 \tag{15}$$

where Lm is the mean value of L. When we recognize that $1-P_k$ is the probability of the section length being $2k+3$ or greater, the right hand of (15) is quite obvious. Thus, the probability of accepting the request of PE(i) is

$$\begin{aligned} Pa(i) &= (1-r\cdot Pm)^{L(i)/2}\cdot(1-r\cdot Pm\cdot Lm/2)\\ &\approx (1-r\cdot Pm\cdot L(i)/2)(1-r\cdot Pm\cdot Lm/2). \end{aligned} \tag{16}$$

Here, an approximation is made by neglecting high orders of $r\cdot Pm$. From (16), the accept rate is derived as

$$Pr=\sum_{i=0}^{N-1}Pa(i)/N=(1-r\cdot Pm\cdot Lm/2)\left(N-r\cdot Pm\sum_{i=0}^{N-1}L(i)/2\right)\Big/N. \tag{17}$$

Applying the central limit theorem to $\Sigma L(i)$, the distribution of $\Sigma L(i)$ is Gaussian with mean of $N\cdot Lm$ regardless of the distribution of L provided the L's are independent. In this case, the mean of Pr is

$$Pm=(1-r\cdot Pm\cdot Lm/2)^2. \tag{18}$$

When $r\cdot Pm$ is relatively small with respect to 1 (it usually is), equation (18) is reduced to

$$\begin{aligned} Pm &= (1-r\cdot Pm)^{Lm}\\ &= 1-Lm\cdot r\cdot Pm. \end{aligned} \tag{19}$$

Comparing equation (1) with equation (19), L has been replaced by Lm and Pr by Pm. Consequently, the models developed thereafter also apply to the case with varying L if these substitutions are made.

AUTHORS' BIOGRAPHIES

Xu BaiQiang—Mr Xu BaiQiang is a Ph.D. candidate at the University of Akron. His main interests are digital circuits, computer architecture, digital signal processing, numerical analysis, applied mathematics, and inverse problems. He received his B.Sc.E.E. from Chengdu Institute of Telecommunication, P.R. China, and his M.Sc.E.E. from the University of Akron.

Nathan Ida—Dr Ida is associate professor of electrical engineering at The University of Akron. His current research interests are in the areas of parallel and vector computation, numerical modeling of electromagnetic fields, electromagnetic wave propagation and nondestructive testing of materials. Dr Ida received his B.Sc.E.E. and M.Sc.E.E. from the Ben Gurion University in Israel and his Ph.D. from Colorado State University.