Analysis of Bus Hierarchies for Multiprocessors *

Donald C. Winsor and Trevor N. Mudge

Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan

don@eecs.umich.edu
tnm@eecs.umich.edu

Abstract: *In order to build large shared-memory multiprocessor* systems that take advantage of current hardware-enforced cache coherence protocols, an interconnection network is needed that acts logically as a single bus while avoiding the electrical loading problems of a large bus. This paper develops models of bus delay and bus throughput to aid in optimizing the design of such a network. These models are used to derive a method for determining the maximum number of processors that can be supported by each of several bus organizations including conventional single-level buses, two-level bus hierarchies, and binary tree interconnections. An example based on a TTL bus is presented to illustrate the methods and to show that shared-memory multiprocessors with several dozen processors are feasible using a simple two-level bus hierarchy.

1 Introduction

Although there have been many proposals, some very complicated, for multiprocessor architectures, one of the most popular is also the simplest. It consists of a single shared bus that connects multiple processors to a shared main memory (Figure 1). Representative examples include the Encore Multimax [1] and the Sequent Balance series [2]. Its popularity is probably due to the fact that it is an evolutionary step from the familiar uniprocessor, and yet it can offer a performance increase for typical multiprogramming workloads that grows linearly with the number of processors, at least for the first dozen or so. The architecture of Figure 1 can also be used in a multitasking environment where single jobs can take control of all the processors and execute in parallel. This is a mode of operation which is infrequently used at present, so we will confine our discussion to a multiprogramming environment in which computational jobs form a single queue for the next available processor.

The maximum performance of these shared-memory systems is extremely sensitive to both bus bandwidth and memory access time. Since cache memories significantly improve both the bandwidth and the average access time, they are an essential component of this class of multiprocessor. When using a private cache memory for each processor (as shown in Figure 1), it is necessary to ensure that all valid copies of a given cache line are the same. (A cache line is the unit of data transfer between cache and main memory.) This requirement is called the *multicache consistency* or *cache coherence* problem. The most promising solutions to hardware-enforced cache coherence require that all processors share a common main memory bus (or the logical equivalent). Each cache monitors all bus activity to identify references to its lines by other caches in the system. This monitoring is called *snooping* on the bus. It has the advantage that coherence is managed by the hardware in a decentralized fashion, avoiding the bottleneck of a central directory. A detailed discussion of cache coherence protocols for snooping caches can be found in [3].

Until recently, the high cost of cache memories limited them to relatively small sizes. For example, the Sequent Balance multiprocessor system uses an 8 K-byte cache for each processor [2]. These small caches have high miss ratios, so a significant fraction of memory requests require service from the bus. The resulting high bus traffic limits these systems to a small number of processors. Advances in memory technology have substantially increased the maximum practical cache memory size. For example, the Berkeley SPUR multiprocessor workstation uses a 128 K-byte cache for each processor [4], and caches as large as 1024 K-bytes are considered for the Encore Ultramax in [5]. Using large caches, it is possible to reduce bus traffic due to individual processors, allowing systems with greater numbers of processors to be built. As the number of processors is increased, a point is reached where capacitive loading, driver current limitations, and transmission line propagation delays become the dominant factors limiting the maximum number of processors.

Interconnections such as multistage networks [6] do not have the bus loading problem of a single bus; however, the bus oriented cache coherence protocols will not work with them. To build very large systems that can benefit from the advantages of the bus-oriented cache coherence protocols, it is necessary to construct an interconnection network that preserves the logical structure of a single bus while avoiding the electrical implementation problems associated with physically attaching all of the processors directly to a single bus.

There are several practical ways to construct a network that logically acts as a shared bus connecting a large number of processors. Figure 2 shows an implementation that uses a two-level hierarchy of buses. If a single bus can support N processors with delay Δ, then this arrangement will handle N^2 processors with delay 3Δ. Figure 3 shows another implementation consisting of a binary tree structure of transceivers. This arrangement can connect N processors (where N is a power of 2) using $2N-2$ transceivers. The maximum delay through this network is $2\log_2 N$ times the delay of a single transceiver. Many other arrangements are possible. In order to select the optimal bus organization for a given implementation technology and a given number of processors, models of bus delay and bus interference are needed. In this paper, we construct these models and use them to derive the optimal organizations and sizes for single buses, two-level bus hierarchies, and binary tree interconnections. Simple relationships are developed that express the largest useful systems that can be constructed with a given processor, cache, and bus organization. We present an example based on a TTL bus to show that shared-memory multiprocessors with several dozen processors are feasible using a simple two-level bus hierarchy. To conclude our discussion of buses, a few remarks are made on the importance of good electrical design to system performance.

*This work was supported in part by DoD grant number MDA904-87-C-4136

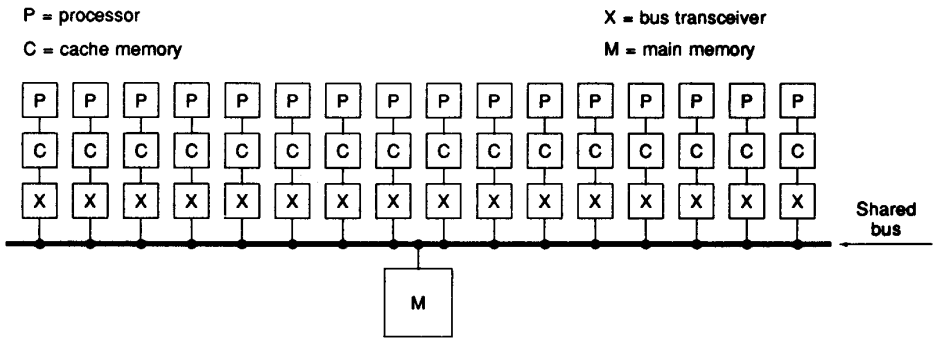


Figure 1: Single bus shared-memory multiprocessor.

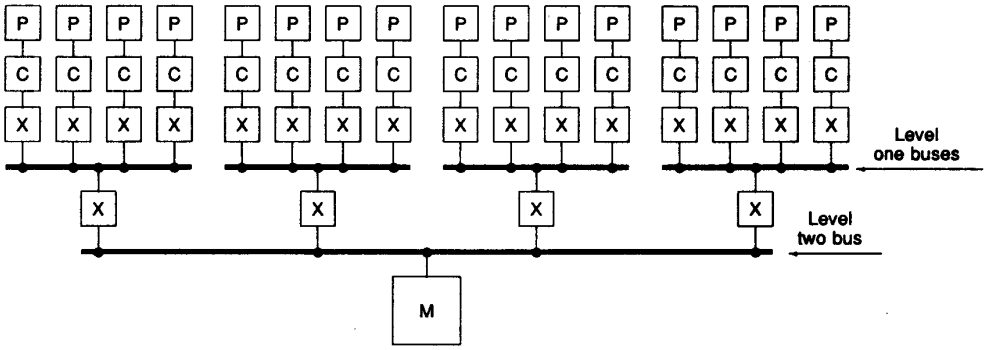


Figure 2: Interconnection using a two-level bus hierarchy.

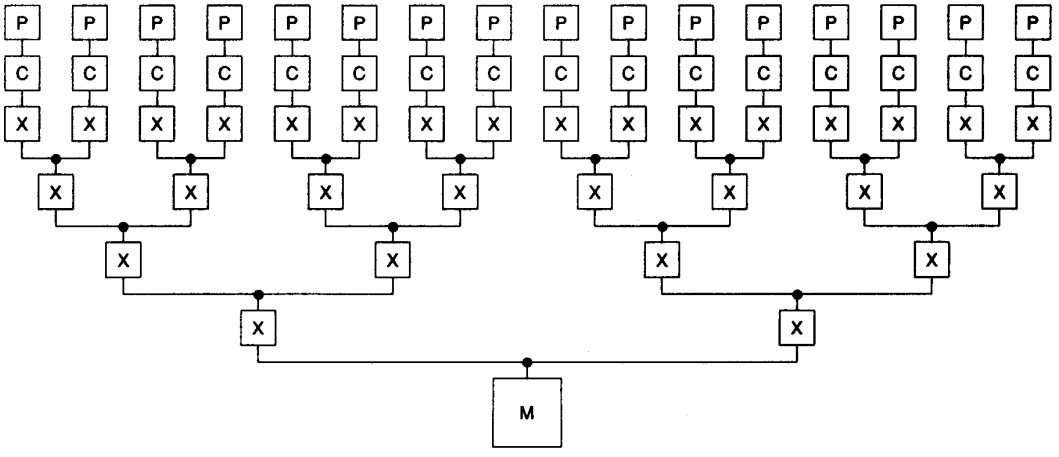


Figure 3: Interconnection using a binary tree.

2 Bus model

We define system throughput as the ratio of the total memory traffic in the system to the memory traffic of a single processor with a zero delay bus. This a useful measure of system performance since it is proportional to the total rate at which useful computations may be performed by the system for a given processor and cache design. In this section we develop a model for system throughput, T, as a function of the number of processors N, the mean time between shared-memory requests from a processor t_r, and the bus cycle time t_c.

2.1 Delay model

In general, the delay associated with a bus depends on the number of devices connected to it. In this section, we will use N to represent the number of devices connected to the bus under discussion. Based on their dependence on N, the delays in a bus can be classified into four general types: constant, logarithmic, linear, and quadratic. Constant delays are independent of N. The internal propagation delay of a bus transceiver is an example of constant delay. Logarithmic delays are proportional to $\log_2 N$. The delay through a binary tree interconnection network as shown in Figure 3 is an example of logarithmic delay. The delay of an optimized MOS driver driving a capacitive load where the total capacitance is proportional to N is another example of logarithmic delay [7]. Linear delays are proportional to N. The transmission line delay of a bus whose length is proportional to N is an example of linear delay. Another example is the delay of an RC circuit in which R (bus driver internal resistance) is fixed and C (bus receiver capacitance) is proportional to N. Finally, quadratic delays are proportional to N^2. The delay of an interconnection network on a VLSI or WSI chip in which both the resistance and the capacitance of the wiring are appreciable is an example of quadratic delay [8].

The total delay of a bus, Δ, can be modeled as the sum of these four components (some of which may be zero or negligible), i.e.,

$$\Delta = k_{\text{const}} + k_{\log}\log_2 N + k_{\text{lin}} N + k_{\text{quad}} N^2$$

The minimum bus cycle time is limited by the bus delay. It is typically equal to the bus delay for a bus protocol that requires no acknowledgment, and it is equal to twice the bus delay for a protocol that does require an acknowledgment. For the remainder of this paper, we will assume that no acknowledgment is required. Thus, the bus cycle time t_c can be expressed as,

$$t_c = k_{const} + k_{\log}\log_2 N + k_{\text{lin}} N + k_{\text{quad}} N^2 \qquad (1)$$

2.2 Interference model

To accurately model the bus performance when multiple processors share a single bus, the issue of bus interference must be considered. This occurs if two or more processors attempt to access the bus at the same time—only one can be serviced while the others must wait. Interference increases the mean time for servicing a memory request over the bus, and it causes the bus utilization for an N processor system to be less than N times that of a single processor system.

If the requests from different processors are independent, as would likely be the case when they are running separate processes in a multiprogramming system, then a Markov chain model of bus interference can be constructed [9,10,11,12]. This model may be used to estimate the effective memory access time and the bus utilization. In the model, states $\Pi_0, \ldots, \Pi_{N-1}$ correspond to the number of processors blocked (it is not possible to have all N processors blocked; one must be accessing memory). Let p be the probability that a processor requests a memory access in a bus cycle, and let q be $1 - p$ by definition. It is assumed that p is the same for all processors and that requests are independent. Let $\alpha_{i,j}$ be the probability that j processors generate new memory requests given that i processors are blocked. Since processors that are blocked cannot generate new requests, this probability is given by the binomial distribution:

$$\alpha_{i,j} = \begin{cases} \binom{N-i}{j} p^j q^{(N-i)-j} & j \le N-i \\ 0 & j > N-i \end{cases}$$

The state transition probabilities in the Markov chain model may be easily obtained from the α values. It is then possible to solve for the state probabilities (π_i), although it does not seem possible to give a closed form representation of the general solution. To compute the state probabilities π_i, define $w_i = \pi_i/\pi_0$. This results in a triangular system of linear equations. Thus, it is easier to solve for the w_i values than to solve for the π_i values directly. We will skip some steps for brevity, but it can be shown that,

$$w_0 = 1$$

$$w_1 = \frac{1 - (\alpha_{0,0} + \alpha_{0,1})}{\alpha_{1,0}} = \frac{1}{q^{N-1}} - (Np + q)$$

and, for $2 \le i < N$,

$$w_i = \frac{w_{i-1} - \sum_{j=0}^{i-1} \alpha_{j,i-j} w_j}{\alpha_{i,0}} = \frac{w_{i-1}}{q^{N-i}} - \sum_{j=0}^{i-1} w_j \binom{N-j}{i-j} p^{i-j}$$

The state probabilities π_i must sum to one, and $\pi_i = w_i \pi_0$, so

$$\sum_{i=0}^{N-1} \pi_i = \sum_{i=0}^{N-1} w_i \pi_0 = \pi_0 \sum_{i=0}^{N-1} w_i = 1$$

therefore,

$$\pi_0 = \frac{1}{\sum_{i=0}^{N-1} w_i}$$

The mean queue length L (the mean number of blocked processors) is given by,

$$L = \sum_{i=0}^{N-1} i\pi_i$$

The mean number of bus cycles needed to service a memory request, s, is one more than the mean queue length, therefore,

$$s = 1 + \sum_{i=0}^{N-1} i\pi_i \qquad (2)$$

We define t_m to be the mean time interval between the memory requests from a processor. This consists of bus queuing and transmission time, processor compute time, and memory access time. Let t_b be the bus time, and t_r be the processor and memory time, so $t_m = t_b + t_r$. Since t_c is the bus cycle time, and a memory request takes s cycles to service, the mean time required for the bus to service a memory request, t_b, is given by, $t_b = st_c$. The memory request probability p is equal to the ratio of the bus cycle time to the memory request time:

$$p = \frac{t_c}{t_m} = \frac{t_c}{t_b + t_r}$$

Substituting st_c for t_b yields,

$$p = \frac{1}{s + \frac{t_r}{t_c}}$$

Let r be given by,

$$r = \frac{t_r}{t_c} \qquad (3)$$

N	T	p	s
1	0.98	0.0196	1.00
2	1.94	0.0291	1.00
3	2.88	0.0385	1.00
4	3.79	0.0476	1.02
5	4.67	0.0565	1.04
6	5.49	0.0650	1.09
7	6.23	0.0731	1.18
8	6.84	0.0803	1.34
9	7.25	0.0863	1.59
10	7.42	0.0904	1.97
11	7.37	0.0927	2.46
12	7.16	0.0933	3.03
13	6.85	0.0927	3.65
14	6.51	0.0912	4.30
15	6.16	0.0893	4.95
16	5.84	0.0871	5.60
17	5.53	0.0847	6.25
18	5.25	0.0823	6.88
19	4.99	0.0799	7.51
20	4.76	0.0776	8.12

Table 1: T, p, and s as a function of N ($r_{\text{lin}} = 0.01$).

then,

$$p = \frac{1}{s + r} \tag{4}$$

The bus utilization is defined as the fraction of bus cycles that are used to service a memory request. This will be a number between zero and one. Since the bus is in use unless the system is in state zero and no processors generate new requests, the bus utilization U is given by,

$$U = 1 - \pi_0 \alpha_{0,0} = 1 - \pi_0 q^N$$

The system throughput T was defined previously as the ratio of the memory traffic for the multiprocessor system to the memory traffic for a single processor with a zero delay bus. This can be calculated from t_c, t_r, and U. The memory traffic for a single processor with $t_h = 0$ is $1/t_r$, and the memory traffic for the multiprocessor system is U/t_c, thus $T = Ut_r/t_c$. Substituting from (3) gives,

$$T = \left(1 - \pi_0 q^N\right) r \tag{5}$$

Solutions of the simultaneous equations (2) and (4) may be obtained for given values of N and r. Equation (5) may then be used to obtain T as a function of N and r.

3 Maximum throughput for a linear bus

In this section, we consider a bus in which the linear component of the bus delay, given by k_{lin} is large compared with the other components of bus delay. We assume that the bus has $N + 1$ connections, N processors and a single memory controller. The bus cycle time t_c for this bus is $k_{\text{lin}}(N + 1)$. To obtain an expression for p, we define the ratio:

$$r_{\text{lin}} = \frac{k_{\text{lin}}}{t_r}$$

Since k_{lin} and t_r are both independent of N, r_{lin} is also independent of N. Substituting $t_c = k_{\text{lin}}(N+1) = t_r r_{\text{lin}}(N+1)$ into (3), (4), and (5) gives:

$$r = \frac{1}{r_{\text{lin}}(N + 1)}$$

$$p = \frac{1}{s + \frac{1}{r_{\text{lin}}(N+1)}} \tag{6}$$

N	maximum r_{lin}	T	p	s
2	0.192	1.11	0.346	1.15
4	0.0536	2.64	0.196	1.38
8	0.0146	5.92	0.107	1.73
16	0.00384	12.82	0.0569	2.28
18	0.00305	14.58	0.0509	2.39
32	0.000985	27.18	0.0295	3.09
64	0.000249	56.79	0.0151	4.28
72	0.000197	64.29	0.0135	4.53
128	0.0000622	117.35	0.00766	6.00
256	0.0000155	240.44	0.00386	8.45
288	0.0000123	271.43	0.00343	8.96
512	0.00000387	489.47	0.00194	11.94
1024	0.000000964	991.58	0.000972	16.88
1152	0.000000761	1117.53	0.000864	17.90

Table 2: Maximum value of r_{lin} as a function of N for a linear bus.

$$T = \frac{1 - \pi_0 q^N}{r_{\text{lin}}(N + 1)} \tag{7}$$

By solving the simultaneous equations (2), (6), and (7), the throughput T may be obtained as a function of the number of processors N and the ratio r_{lin}.

For a given value of r_{lin}, if the total system throughput is plotted as a function of the number of processors, it will be found to increase up to a certain number of processors and then decrease again when the bus becomes overloaded. This occurs because once the bus utilization U gets close to one, further increases in the number of processors N will not significantly increase U, but the bus cycle time t_c will increase due to increased bus loading. Thus, the throughput $T = Ut_r/t_c$ will decrease.

The effects of bus loading are illustrated in Table 1 where throughput is shown as a function of N for a particular r_{lin}. It can be seen that the throughput increases as N increases until $N = 10$ and then it decreases again.

Using the results of the bus interference model described above, a solution was obtained for the maximum number of useful processors as a function of r_{lin}. Since the number of processors is constrained to be an integer, the results are presented to show the maximum value of r_{lin} for which it is reasonable to use N processors. We define the maximum r_{lin} value to be that value of r_{lin} at which the throughput of a system with $N + 1$ processors is equal to that of a system with N processors. Table 2 shows this for a range of values of N. The request probability p, the mean number of bus cycles for service s, and the throughput T for these values of N and r_{lin} are also shown. From Table 2, it can be shown that for large N, the following approximation holds,

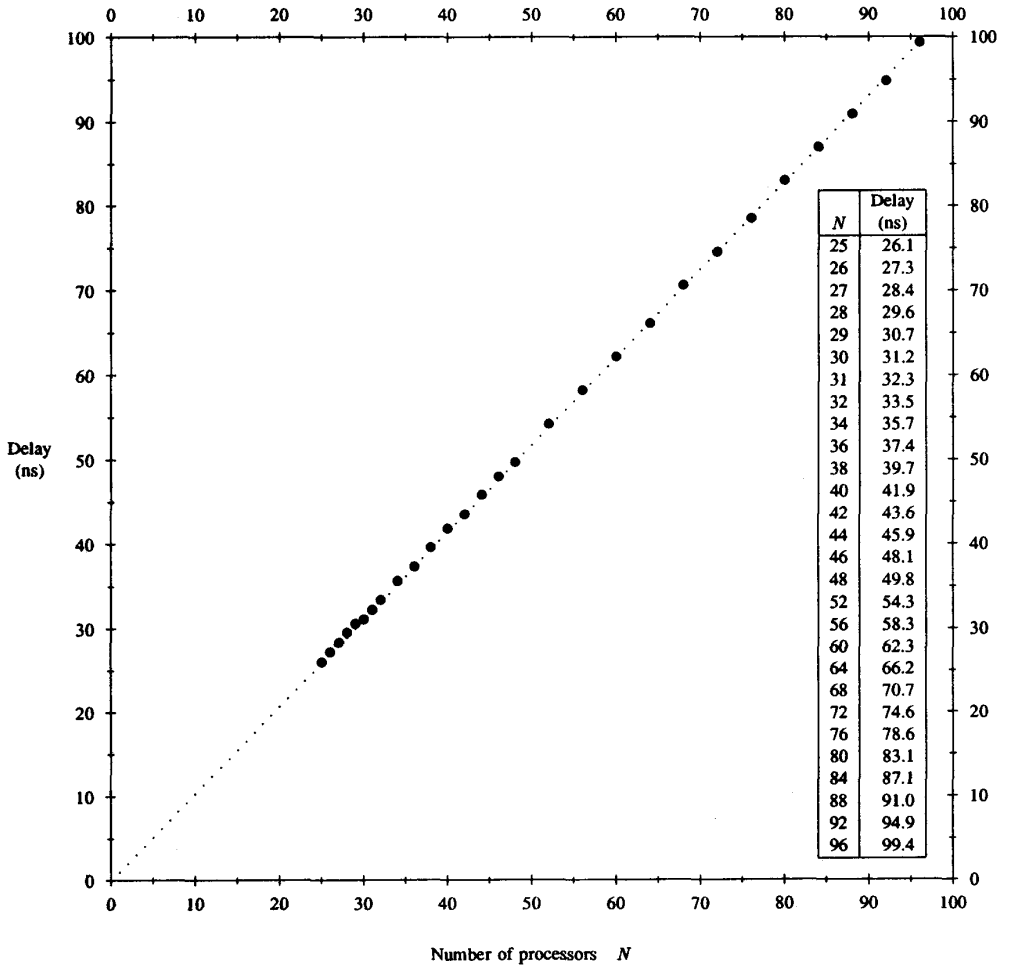
$$\text{maximum } r_{\text{lin}} \approx N^{-2}$$

Since r_{lin} is specified by the processor, cache, and bus characteristics, this relationship gives an approximate idea of the largest useful system that can be constructed with these components.

3.1 TTL bus example

In this section, we will illustrate the linear bus case with a system using standard TTL components for its bus transceivers. First, we show that a TTL bus is, in fact, a linear bus.

The delay of a TTL bus consists primarily of the time required for the driver to charge the transceiver capacitances. The total capacitance which must be charged by the driver is proportional to the number of transceivers connected to the bus. Since the driver acts as a nearly constant current source, the delay is nearly proportional to the number of transceivers connected to the bus. Thus, the k_{lin} term will be dominant in the bus delay model.



N	Delay (ns)
25	26.1
26	27.3
27	28.4
28	29.6
29	30.7
30	31.2
31	32.3
32	33.5
34	35.7
36	37.4
38	39.7
40	41.9
42	43.6
44	45.9
46	48.1
48	49.8
52	54.3
56	58.3
60	62.3
64	66.2
68	70.7
72	74.6
76	78.6
80	83.1
84	87.1
88	91.0
92	94.9
96	99.4

Figure 4: Bus delay as calculated from ODEPACK simulations.

An estimate of the bus delay for a large bus may be made as follows. We assume a 64 ma FAST Schottky TTL bus transceiver. Typical characteristics for this component are 12 pf output capacitance, 64 ma driver current, and 2 volt logic swing [13]. Assuming 1 pf per connection of additional capacitance from the bus wiring gives a total of 13 pf per connection. With terminations of 64 ohms to 2.5 volts, and a logic low level of 0.6 volts, approximately 30 ma of driver current flows through the terminations, leaving 34 ma to charge the capacitances. From these figures, we get,

$$k_{\text{lin}} = \frac{\Delta}{N} \approx \frac{(13\,\text{pf})\,(2\,\text{V})}{(34\,\text{ma})} = 0.76\,\text{ns}.$$

To obtain a more precise estimate of the delay of a TTL bus, a circuit model was constructed in which the driving transceiver was modeled as a nonlinear current source in parallel with a capacitor, and the receiving transceivers were modeled as nonlinear resistors in parallel with capacitors. The bus was modeled with lumped inductances and capacitances. Using this model, the bus was represented by a system of nonlinear ordinary differential equations. Parameters for this model were obtained from [13]. The ODEPACK (Ordinary Differential Equation Package) software package was used to obtain a solution to this system, from which the bus delay was determined. The results for buses with 25 to 96 connections on them are shown in Figure 4. For smaller N, effects due to transmission line reflections dominate, and the delay model tends to be somewhat unpredictable. For larger N, the computation time required for the simulations becomes prohibitive.

From Figure 4, it can be seen that for large N ($N \geq 25$), the delay is essentially linear in the number of bus connections, with $k_{\text{lin}} \approx 1.04$ ns. Thus, a linear bus model may be used for a TTL bus. The value for k_{lin} from Figure 4 is slightly larger than the value estimated from the transceiver capacitance and drive current and it represents the typical time to reach correct logic values everywhere on the bus. A practical implementation must use a time significantly greater than this figure to allow for worst-case variations, clock skew, arbitration time, etc. To account for these in our examples, we consider existing TTL bus designs. Several TTL based buses are described in [14]. The maximum bandwidth claimed for any of these buses is 57 megabytes per second with 21 bus

slots. This bus is four bytes wide, so we have

$$k_{\text{lin}} = \frac{(4 \text{ bytes})}{(57 \times 10^6 \text{ bytes/sec})(21)} = 3.34 \text{ ns} \quad (8)$$

We will use this figure for k_{lin} for our examples that follow, and assume a linear model as Figure 4 would suggest. Consider a system with the following characteristics:

- 16.67 MHz Motorola MC68020 processor
- 64 K-byte cache
- 8 byte bus data path
- 16 byte cache line size
- 3 bus cycles needed to fetch data for a cache miss
- 3 bus cycles needed to write a dirty line back from the cache
- 260 ns main memory access time
- 7.0 ns bus transceiver propagation delay

The following 16.67 MHz 68020 performance figures are from [15]:

- 2.52 million instructions per second
- 1.201 memory references per instruction
- 52.1% of memory references are data references

The following cache performance figures are from [16] and [17]:

- Cache miss ratio = 0.030
- Half of data misses require writing a dirty line back from the cache

With this data, an estimate may be obtained for t_r:

$$t_r = \frac{\frac{1}{(2.52 \times 10^6/s)(1.201)(0.030)} + 2.60 \times 10^{-7} \text{ s} + (2)(7.0 \times 10^{-9} \text{ s})}{3 + (3)(0.521)(0.5)} = 2.98 \, \mu\text{s}$$

The first term in the numerator gives the mean processor compute time between references (this is the reciprocal of instructions per second × references per instruction × miss ratio), the second term is the fixed memory access time, and the last term is the fixed round-trip bus transceiver delay. The denominator is the mean number of bus cycles for a single memory reference and is the sum of the number of bus cycles needed to service a cache miss (three) and the average number of cycles to write back dirty lines. Three bus cycles are needed to fetch data or write a dirty line because one is required to issue the address and two are needed to transfer the data. From the figure obtained earlier for k_{lin} in (8), we get $r_{\text{lin}} = k_{\text{lin}}/t_r = 0.00112$. With this value of r_{lin}, a maximum throughput of 25.4 can be achieved using 30 processors.

4 Optimization of a two-level bus hierarchy

We now consider a bus that is implemented with two physical levels, as in the design of Figure 2. Using a delay model as described previously to model each level, it is possible to select the number of interconnections that should be made at each level in order to minimize the total delay. For example, to build a 256 processor system, the following are plausible approaches: give each processor its own interface to a 256 slot bus, connect pairs of processors to a 128 slot bus, or connect groups of 16 processors to a 16 slot bus. In this section, a method will be developed for selecting the organization with the minimum delay.

If we let N be the total number of devices connected through both levels and B be the number of devices connected together at the first level, then N/B devices must be connected together at the second level. The optimization problem is to choose B to minimize the total delay, Δ, for a given N. The longest path in this hierarchy is from a processor through its level one bus, down through the level two bus, and back up

	Level one constant	Level one logarithmic	Level one linear	Level one quadratic
Level two constant	Constant delay; B doesn't matter	1	1	1
Level two logarithmic	N	1 if $2k_{\log_1} > k_{\log_2}$; N if $2k_{\log_1} < k_{\log_2}$	$\frac{k_{\log_2}}{2k_{\text{lin}_1}}$	$\sqrt{\frac{k_{\log_2}}{4k_{\text{quad}_1}}}$
Level two linear	N	$N\frac{k_{\text{lin}_2}}{2k_{\log_1}}$	$\sqrt{N\frac{k_{\text{lin}_2}}{2k_{\text{lin}_1}}}$	$\sqrt[3]{N\frac{k_{\text{lin}_2}}{4k_{\text{quad}_1}}}$
Level two quadratic	N	$N\sqrt{\frac{k_{\text{quad}_2}}{k_{\log_1}}}$	$\sqrt[3]{N^2\frac{k_{\text{quad}_2}}{k_{\text{lin}_1}}}$	$\sqrt{N\sqrt{\frac{k_{\text{quad}_2}}{2k_{\text{quad}_1}}}}$

Table 3: Value of B for minimum delay in a two-level bus hierarchy.

to a different level one bus (see Figure 2). It is necessary to include this last delay to allow the processors on that bus to perform their snooping operations. Thus, the delay of the two-level hierarchy is twice the level one delay plus the level two delay, i.e.,

$$\Delta = 2\Delta_1 + \Delta_2$$

where,

$$\Delta_1 = k_{\text{const}_1} + k_{\log_1} \log_2 B + k_{\text{lin}_1} B + k_{\text{quad}_1} B^2$$

and,

$$\Delta_2 = k_{\text{const}_2} + k_{\log_2} \log_2 \left(\frac{N}{B}\right) + k_{\text{lin}_2} \left(\frac{N}{B}\right) + k_{\text{quad}_2} \left(\frac{N}{B}\right)^2$$

therefore,

$$\Delta = 2k_{\text{const}_1} + 2k_{\log_1} \log_2 B + 2k_{\text{lin}_1} B + 2k_{\text{quad}_1} B^2 + k_{\text{const}_2} + k_{\log_2} \log_2 \left(\frac{N}{B}\right) + k_{\text{lin}_2} \left(\frac{N}{B}\right) + k_{\text{quad}_2} \left(\frac{N}{B}\right)^2$$

Taking the derivative of Δ with respect to B and setting the result to zero gives,

$$4k_{\text{quad}_1} B^4 + 2k_{\text{lin}_1} B^3 + (2k_{\log_1} - k_{\log_2}) B^2 - k_{\text{lin}_2} NB - 2k_{\text{quad}_2} N^2 = 0$$

Although a closed form solution of this quartic equation is possible, it is complex and gives little additional insight into the problem. However, it is useful to consider the cases in which a particular type of delay is dominant. In these cases, simpler approximations are possible. The value of B to minimize the total delay is shown in Table 3 for all combinations of level one and level two delays. The situation in which the dominant delays in level one and level two are different may occur when the levels are implemented in different technologies. For example, one level may be implemented entirely within a single VLSI chip, while another level may consist of connections between the chips.

4.1 Maximum throughput for a two-level bus hierarchy

Using Table 3, we can determine the maximum number of processors when the two-level bus hierarchy of Figure 2 is used instead of a single

N	maximum r_{lin}	T	p	s
8	0.0130	5.66	0.113	1.85
18	0.00418	13.97	0.0537	2.68
32	0.00182	26.32	0.0308	3.50
72	0.000551	63.05	0.0138	5.13
128	0.000235	115.79	0.00780	6.76
288	0.0000705	269.28	0.00347	10.02
512	0.0000299	486.78	0.00195	13.27
1152	0.00000893	1113.78	0.000868	19.77

Table 4: Maximum value of r_{lin} as a function of N for a two-level bus hierarchy.

bus. We will assume the electrical characteristics of the two levels are identical and linear. Then, the bus cycle time at each level is approximately k_{lin} times the number of devices (processors or clusters of processors) connected at that level. Furthermore, the optimal organization for N processors is to connect them as $\sqrt{2N}$ clusters with $\sqrt{N/2}$ processors in each cluster. The level one buses will have $\sqrt{N/2}$ processor connections and one connection to the level two bus for a total of $\sqrt{N/2}+1$ connections. Similarly, the level two bus will have $\sqrt{2N}$ connections to the level one buses and one connection to the main memory for a total of $\sqrt{2N}+1$ connections. The bus cycle time is twice the level one delay plus the level two delay, so $t_c = k_{\text{lin}}(\sqrt{8N}+3)$. Substituting $t_c = k_{\text{lin}}(\sqrt{8N}+3) = t_r r_{\text{lin}}(\sqrt{8N}+3)$ into (3), (4), and (5) gives:

$$r = \frac{1}{r_{\text{lin}}\left(\sqrt{8N}+3\right)}$$

$$p = \frac{1}{s+\frac{1}{r_{\text{lin}}\left(\sqrt{8N}+3\right)}} \tag{9}$$

$$T = \frac{1-\pi_0 q^N}{r_{\text{lin}}\left(\sqrt{8N}+3\right)} \tag{10}$$

By solving the simultaneous equations (2), (9), and (10), the throughput, T, may be obtained as a function of the number of processors N and the ratio r_{lin}. As was done for a single level bus, the maximum value of r_{lin} was determined for each value of N. Table 4 shows these results, along with the request probability p, the mean number of bus cycles for service s, and the throughput T for these values of N and r_{lin}.

It can be seen that for large N, the following approximation may be used for the maximum value of r_{lin}:

$$\text{maximum } r_{\text{lin}} \approx (2N)^{-\frac{3}{2}}$$

As befo.e, this relation gives a simple approximation for the largest useful system that may be constructed with a given processor, cache, and two-level bus.

For moderate to large N, the results for throughput are much better than can be obtained with a single level. For small N, however, the improvement is not significant, and for $N < 12$, throughput is actually worse for a two-level bus hierarchy than for a single bus. With the value of r_{lin} from the previous example, $r_{\text{lin}} = 0.00112$, a maximum throughput of 37.8 can be achieved using 50 processors. This represents a 49% improvement in maximum throughput over a single level bus.

5 Maximum throughput using a binary tree interconnection network

Finally, in this section, we determine the maximum number of processors when the binary tree interconnection network shown in Figure 3 is used. In this case, the logarithmic component of the bus delay, given by $k_{\log}$, is large compared with the other components of bus delay. For this system $t_c = k_{\log}\log_2 N$. Defining $r_{\log} = k_{\log}/t_r$ gives:

$$t_c = t_r r_{\log}\log_2 N$$

$$r = \frac{1}{r_{\log}\log_2 N}$$

$$p = \frac{1}{s+\frac{1}{r_{\log}\log_2 N}} \tag{11}$$

$$T = \frac{1-\pi_0 q^N}{r_{\log}\log_2 N} \tag{12}$$

The solution for this case is shown in Table 5, and the approximation for large N is:

$$\text{maximum } r_{\log} \approx \frac{1}{N log_2 N}$$

N	maximum $r_{\log}$	T	p	s
2	0.307	1.46	0.231	1.06
4	0.111	2.89	0.173	1.28
8	0.0401	6.18	0.101	1.62
16	0.0150	13.34	0.0534	2.03
32	0.00586	28.42	0.0273	2.50
64	0.00240	59.40	0.0138	3.01
128	0.00102	122.31	0.00698	3.57
256	0.000446	249.17	0.00352	4.15
512	0.000198	504.01	0.00177	4.73
1024	0.0000896	1014.86	0.000892	5.31

Table 5: Maximum value of $r_{\log}$ as a function of N for a binary tree interconnection.

We now consider our example system with $t_r = 2.98\,\mu$s using this bus structure. The delay constant $k_{\log}$ is equal to twice the propagation delay of the transceivers used, which is approximately 7.0ns for our TTL-based example. Thus, $r_{\log} = k_{\log}/t_r = 0.00470$ in this case. For this value of $r_{\log}$, a maximum throughput of 35.4 can be obtained by using 64 processors. This is a 39% improvement in maximum throughput over the single level bus, but it is poorer than the two-level bus hierarchy.

6 Discussion and concluding remarks

We have presented a model of bus delay and shown how it may be used to analyze three bus organizations assuming a TTL implementation. In our examples the two-level organization yielded the greatest throughput. This is a consequence of the technology chosen for the examples and may not, in general, be the case.

The TTL technology we have used to illustrate our examples is adequate but not the best choice for implementing a bus system (we used it because its parameters were easily obtained). In particular, the bus delay component, k_{lin}, may be decreased with better bus transceiver technology. A new transceiver technology, BTL (backplane transceiver logic), has significantly better performance than TTL for bus applications [18]. It achieves this by having a lower capacitance and a smaller swing between logic levels. Good electrical design of the bus is as important as clever cacheing strategies. This can be appreciated if we note that halving the bus capacitance allows a $\sqrt{2}$ (i.e. 41%) increase in N and T for the single-level case, and $\sqrt[3]{4}$ (i.e. 59%) for the two-level case.

Finally, if better transceivers and larger caches are not sufficient for achieving the desired throughput, then techniques for distributing the processor to memory traffic over several buses such as those described in [19] and [20] may be used. Combinations of these techniques should allow shared memory systems with several hundred processors to be constructed in the near future.

References

[1] *Multimax Technical Summary*, Encore Computer Corporation, May, 1985.

[2] *Balance Technical Summary*, Sequent Computer Systems, Inc., November 19, 1986.

[3] James Archibald and Jean-Loup Baer, "Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model", *ACM Transactions on Computer Systems*, volume 4, number 4, November, 1986, pages 273–298.

[4] Mark Hill, Susan Eggers, Jim Larus, George Taylor, Glenn Adams, B. K. Bose, Garth Gibson, Paul Hansen, Jon Keller, Shing Kong, Corinna Lee, Daebum Lee, Joan Pendleton, Scott Ritchie, David Wood, Ben Zorn, Paul Hilfinger, Dave Hodges, Randy Katz, John Ousterhout, and Dave Patterson, "Design Decisions in SPUR", *Computer*, November, 1986, pages 8–22.

[5] Andrew W. Wilson Jr., "Hierarchical Cache / Bus Architecture for Shared Memory Multiprocessors", In *The 14th Annual International Symposium on Computer Architecture Conference Proceedings*, June 2–5, 1987, pages 244–252.

[6] H. J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing: Theory and Case Studies*, Lexington Books, Lexington, Massachusetts, 1985.

[7] Carver Mead and Lynn Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980, pages 12–14.

[8] Abhiram G. Ranade and S. Lennart Johnsson, "The Communication Efficiency of Meshes, Boolean Cubes and Cube Connected Cycles for Wafer Scale Integration", In *Proceedings of the 1987 International Conference on Parallel Processing*, August 17–21, 1987, pages 479–482.

[9] C. E. Skinner and J. R. Asher, "Effects of storage contention on system performance", *IBM Systems Journal*, volume 8, number 4, 1969, pages 319–333.

[10] D. P. Bhandarkar and S. H. Fuller, "Markov Chain Models for Analyzing Memory Interference in Multiprocessor Computer Systems", In *Proceedings of the 1st Annual Symposium on Computer Architecture*, December, 1973, pages 1–6.

[11] F. Baskett and A. J. Smith, "Interference in multiprocessor computer systems with interleaved memory", *Communications of the ACM*, volume 19, number 6, June, 1976, pages 327–334.

[12] Trevor N. Mudge and Humoud B. Al-Sadoun, "Memory Interference Models with Variable Connection Time", *IEEE Transactions on Computers*, volume C-33, number 11, November, 1984, pages 1033–1038.

[13] *Motorola Schottky TTL Data*, Motorola Inc., 1983.

[14] Paul L. Borrill, "MicroStandards Special Feature: A Comparison of 32-Bit Buses", *IEEE Micro*, volume 5, number 6, December, 1985, pages 71–79.

[15] Doug MacGregor and Jon Rubinstein, "A Performance Analysis of MC68020-based Systems", *IEEE Micro*, volume 5, number 6, December, 1985, pages 50–70.

[16] Alan Jay Smith "Cache Evaluation and the Impact of Workload Choice", In *The 12th Annual International Symposium on Computer Architecture Conference Proceedings*, June 17–19, 1985, pages 64–73.

[17] Alan Jay Smith, "Line (Block) Size Choice for CPU Cache Memories", *IEEE Transactions on Computers*, volume C-36, number 9, September, 1987, pages 1063–1075.

[18] Balu Balakrishnan, "32-Bit System Buses — A Physical Layer Comparison", *Computing*, August 27, 1987, pages 18–19 and September 10, 1987, pages 32–33.

[19] Donald C. Winsor and Trevor N. Mudge, "Crosspoint Cache Architectures", In *Proceedings of the 1987 International Conference on Parallel Processing*, August 17–21, 1987, pages 266–269.

[20] Trevor N. Mudge, John P. Hayes, and Donald C. Winsor, "Multiple Bus Architectures", *Computer*, June, 1987, pages 42–48.