Interconnecting off-the-shelf microprocessors

by HUMOUD B. AL-SADOUN, O. A. OLUKOTUN, and T. N. MUDGE
University of Michigan
Ann Arbor, Michigan

ABSTRACT

This paper outlines the design and analysis of a crossbar-type interconnection network that can be used with off-the-shelf microprocessor components to construct a four-processor/eight-memory module multiprocessor system. The only custom component is an interconnection (ICN) chip presently being fabricated by the authors. The ICN chip integrates 4 × 8 crosspoints for a 3-bit bus slice together with related priority circuitry into a single component. System design using the ICN chip is illustrated with the Intel 8086 family of microprocessor components. An analysis of the resulting system is presented. The degradation of performance resulting from the memory interference associated with multiprocessors is shown to be small compared to that resulting from the setup time required by the interconnection logic.

INTRODUCTION

The use of multiprocessors in appropriate situations can improve the performance of a wide variety of computing tasks, particularly with respect to speed and reliability. Several currently available microprocessor families provide VLSI components that support the construction of multiprocessor systems. Typically, these components are intended for shared-bus MIMD multiprocessors. Such systems are composed of a number of processors which share a common memory by means of a single shared bus and which may possibly have local memory. The shared bus can be a bottleneck that offsets the advantage of having multiple processors if their combined request rate to the shared memory exceeds the bus bandwidth. In this paper we outline the design of an integrated circuit, the ICN (interconnection) chip, that simplifies the construction of multiprocessors that are not constrained to share memory through a single bus. Specifically, the ICN chip together with off-the-shelf microprocessor support chips makes possible the construction of a system in which four processors share eight memory modules through a crossbar interconnection. Individual copies of the chip integrate the necessary logic for a 3-bit bus slice together with related priority circuitry. The ICN chip is presently being fabricated by the authors and is the successor to two earlier prototypes.[1,2] In principle the concept can easily be extended to systems with N processors sharing M memories, subject only to the constraint of pin limitations imposed by the technology used to package the ICN chip. By organizing the interconnection logic as a stack of slices, there can be considerable replication of priority logic and crosspoint selection logic; however, reducing the number of components, rather than gates, and making the components easy to use are more important considerations from a systems design viewpoint. This is evident from the typical replication of address logic that results from organizing memories from slice components.

If a high bandwidth connection to memory is required in a multiprocessor, there are two major advantages to using a crossbar organization compared to multistage alternatives such as Delta networks, cube networks or Banyan networks.[3] First, there is the relative ease with which crossbars can be controlled in MIMD mode. Second, there is the absence of intra-network interference that arises with multistage networks. These advantages are bought at the expense of gate complexity. In particular, the gate complexity of the crossbar is $O(n^2)$ if $N = M = n$, and that of the Delta, cube, or Banyan is $O(n \log_2 n)$; however, in terms of VLSI layout the space complexity is closer to $O(n^2)$ in both cases if logic gates are discounted,[4,5] and higher if priority and related logic are considered.[1] Thus, for systems with less than a few dozen processors operating in MIMD mode the advantages of a crossbar connection outweigh its disadvantages.

The remainder of this paper is organized as follows. The next section describes the operation of a crossbar constructed from ICN chips and Intel support chips. The section following the next section develops an analytical model for the performance of multiprocessor systems constructed using ICN chips. Closing remarks are presented in the conclusion.

A CROSSBAR-BASED MULTIPROCESSOR

As noted, ICN chips can be used to construct a crossbar to interconnect four processors and eight memories. The crossbar allows simultaneous connections between the processors and the memories. The resulting multiprocessor is diagrammed in Figure 1. The crossbar is designed so that it interfaces directly with Intel 8086 microprocessors augmented by several support chips; in particular, Intel 8289 bus arbiters are used to multiplex processors onto multimaster system buses and avoid contention problems between bus masters. Each memory port can be regarded as a multimaster system bus, and each processor can be regarded as a bus master. The crossbar requires four 8289s to control the access of the four processors to the eight memories, and since each multimaster system bus is made up of 40 signal lines, the crossbar requires 14 ICN slices.

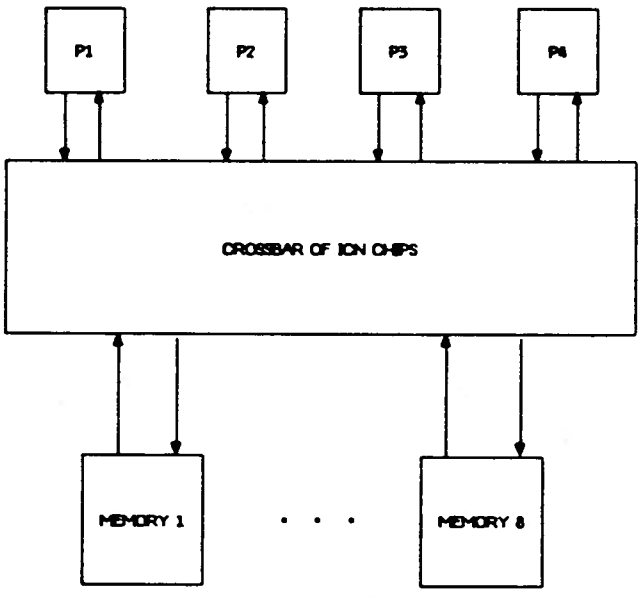

Figure 1—Crossbar-based multiprocessor

Interface between Processor and Crossbar

In common with most microprocessors, the 8086 lacks the capability of requesting bus access and recognizing bus grants from a multimaster bus. Therefore it is necessary to have extra logic to generate the necessary signals for sending bus requests and receiving bus grants. This extra logic comes in the form of an Intel 8289 bus arbiter, one of which is associated with every bus master (processor) in the multiprocessor. These bus arbiters are all synchronized by BLCK, the common bus clock.[6] (See Figure 2 for further details of each of the four processors of Figure 1.)

The 8086 is unaware of its attached arbiter's existence and therefore issues commands as though it had exclusive use of the system memory. The 8289 monitors its 8086's status lines (S2–S0) to detect the beginning of a bus cycle. At the beginning of the bus cycle the bus controller (8288), which also monitors the 8086's status lines, generates an ALE (address latch enable) signal to latch the address information from the 8086. If the processor is in control of the bus (i.e., it has requested and received a memory), it enables the outputs of the bus controller (8288) and the address latches (8283), and the bus cycle continues as usual. If the processor does not have control of the bus, the arbiter forces the outputs of the 8288 and the address latches into their high impedance state. The 8288 in turn forces the outputs of the data transceivers (8287) into their high impedance state. At the same time the clock generator (8284) is prevented from sending a ready signal to the 8086, which forces the 8086 to enter its wait state after T3 of the bus cycle in progress (bus cycles have four subcycles T1, T2, T3 and T4). Once the arbiter is granted bus access, it enables the outputs of the 8288 and the address latches. The addressed memory port returns an acknowledge signal to the clock generator when the data transfer is complete. This acknowledge signal causes the clock generator to send a ready signal to the 8086. The 8086 then exits its wait state and completes the bus cycle in progress.

When an arbiter detects the beginning of a bus cycle and does not have control of the bus, it proceeds to request the bus by activating its BREQ (bus request) line. The BREQ line from each arbiter is fed into the crossbar. Our application makes use of the three highest bits of address to determine which of the eight memories the processor is requesting. Therefore, it is necessary that the address be latched by the crossbar's internal latches and that the connections be established within the crossbar before BREQ is activated by the arbiter (see Figure 3). This sequence of events is ensured by setting the arbiter in resident bus mode (RESB pin high). This

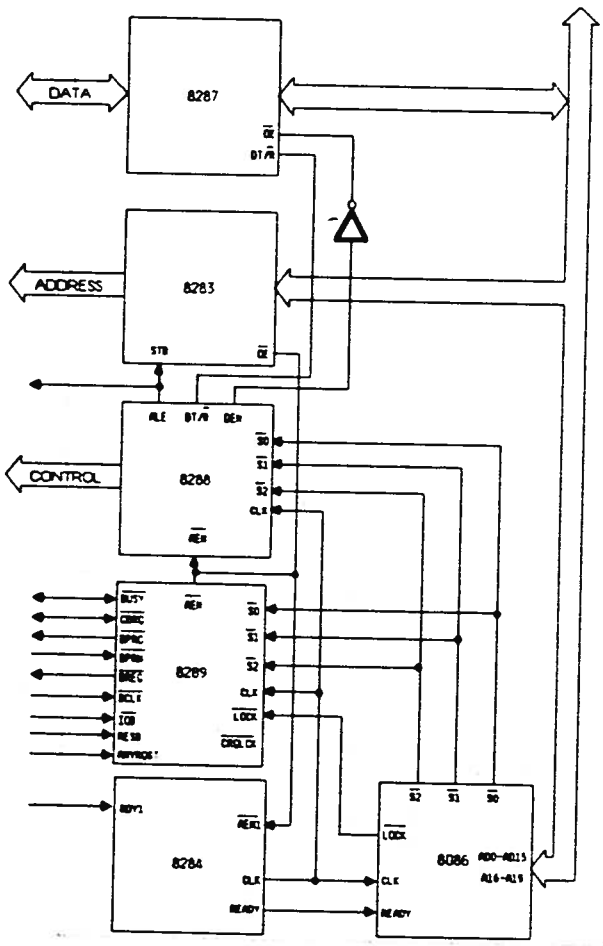

Figure 2—Processor details

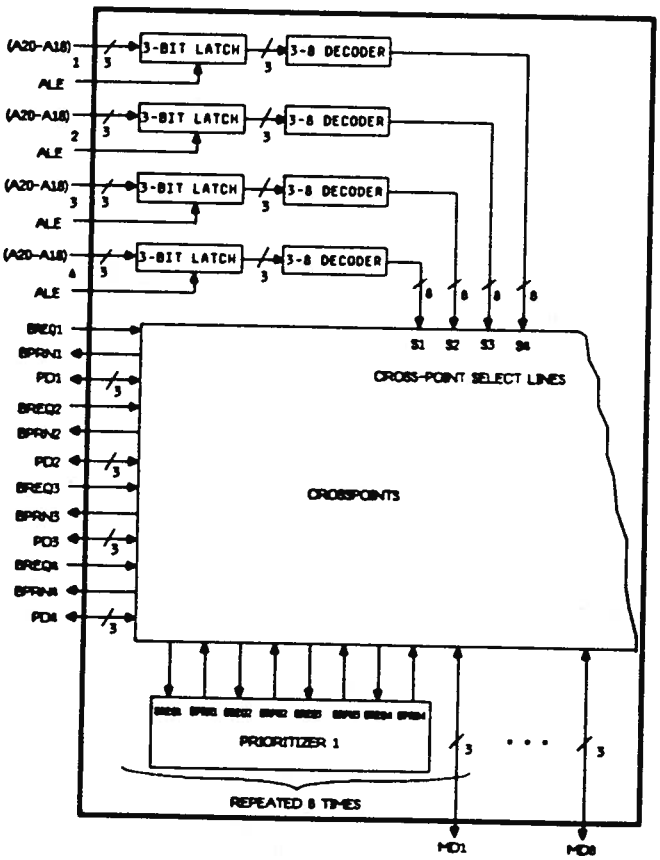

Figure 3—ICN chip

mode ensures the BREQ will not be activated until after the ALE (address latch enable) line is activated. Once the crosspoints have been set, the BREQ and BPRN (bus priority in) lines connect to the prioritizing logic for the memory requested. At the same time the BUSY and data lines (one set of bidirectional lines PD1, . . . , PD4) are also connected to those of the memory selected. A parallel priority resolving technique is used to decide which processor should have a memory in the event of a simultaneous request. The BREQ and BPRN lines of each arbiter requesting a given memory simultaneously are connected via the crossbar to the appropriate prioritizing logic for that memory. When the BREQ lines are activated, the prioritizer will return a BPRN active signal to the arbiter with the highest priority. Its associated processor will then take control of the memory as soon as it is no longer busy (BUSY line inactive). BUSY is an active low OR-tied signal that is connected via the crossbar to every arbiter in the system. Every memory has a BUSY line associated with it. When the BUSY line becomes inactive, the arbiter with priority takes control of the memory by activating BUSY to prevent any other arbiter from taking the memory. The arbiter maintains its BREQ active throughout the time it is connected to a memory.

Memory Surrender Conditions

It is possible for a processor to switch between memory modules from one bus cycle to the next. If we allow an arbiter to keep a memory between bus cycles, the possibility exists that a processor could access a busy memory in a subsequent bus cycle and cause a conflict. This problem could arise if the location of the present bus cycle address is in a different memory from the memory addressed during the previous bus cycle. The arbiter would assume it has control of the memory and instruct its attached bus controller and 8086 to proceed with the bus cycle as usual. A conflict would arise if this new memory is already busy. We have therefore configured our system so that an arbiter gives up the memory under its control after completing a bus cycle regardless of whether another processor is requesting that memory or not (packet switched). This is done by strapping ANYRQST (any request) high and CBRQ (common bus request) low. CBRQ is an open collector signal of the arbiter which can function either as an output or an input, depending on whether its associated arbiter has control of a memory. As an output it is sent by a lower priority arbiter to request a memory from a higher priority arbiter. As an input it instructs the arbiter presently controlling a memory that a lower priority arbiter would like the memory. ANYRQST is a signal of the arbiter that when strapped high signifies that the arbiter should release the memory after the end of the current bus cycle. A provision could be made to control CRQLCK (common request lock) via an I/O port from the processor. Activating CRQLCK prevents the 8289 from releasing the memory to any processor having a lower priority. In this way a processor could retain a memory as long as no other processor with higher priority requested it (circuit switched). One would have to make sure, when using such a provision, that while CRQLCK is activated all instructions and data reside in one memory.

ANALYTICAL MODEL

The behavior of the multiprocessor system described above can be approximated by a stochastic process, given the following assumptions about its operation. At the beginning of the bus cycle a processor selects a memory module at random with probability $1/M$ (there are M memory modules) and makes a request to access that module with probability r (≤ 1). If more than one processor requests the same memory module, the arbitration logic will choose the processor with higher priority. The other processors will continue to request the memory module until they are allowed access. The time it takes to perform this arbitration, i.e., to set up the processor-crossbar interface logic, is two bus cycles. The connection time between the processors and the memory modules will last for C bus cycles. A processor has at most one request waiting to be serviced at any time. During operation the behavior of each of the processors is considered to be independent but statistically identical. The memory access priority assigned to each of the processors by the prioritizing logic in the ICNs is as follows: the first processor has the highest priority in the first two memories, the second highest priority in the second two memories, the third highest priority in the third two memories, and the fourth highest priority in the fourth two memories; the second processor has the first highest priority in the second two memories, the second highest priority in the third two memories, the third highest priority in the fourth two memories, and the fourth highest priority in the first two memories; a similar pattern is repeated for the third and fourth processors.

The behavior of the multiprocessor system, under the operation assumptions stated above, can be described by using a discrete-time Markov chain. However, such a chain has an unmanageably large state space.[7] To avoid this, an approximate model can be used. In this study, we will use the equivalent rate model.[8] This model assumes that a memory module that receives more than one request selects any one of the requesting processors equiprobably, i.e., all the processors are modeled as having the same priority. Furthermore, the model assumes that the steady-state flow of the processors to the memory modules is equal to the flow of the processors from the memory modules: the equivalent rate model is an extension of the steady-state flow model.[9] The derivation of the model proceeds as follows: the rate of requesting a multiple bus cycle connection, r, is transformed to the equivalent rate of requesting a single bus cycle connection, r_{eq}. The quantity r_{eq} represents the connection time as a fraction of the total average processor cycle which contains both the think time and the connection time. The think time of a processor, T, is the time elapsed between releasing a memory module and making the next request for memory connection. Hence, r_{eq} is expressed as follows:

$$r_{eq} = \frac{C'}{T + C'}$$

where $T = 1/r - 1$ and $C' = C + 2$. The term C' includes the two bus cycles that are needed to set up the processor-crossbar interface logic. In our case where the crossbar is operating in

packet switched mode, $C = 4$ bus cycles. A measure of the effectiveness of the crossbar is its bandwidth, BW, which can be expressed as follows:

$$BW = N U_p r_{eq}$$

where U_p is the processor utilization, i.e., the probability that a processor is thinking or accessing a memory module. Hence, $N U_p$ is the expected number of processors thinking or accessing; and $N U_p r_{eq}$ is the expected number of processors accessing, i.e., the crossbar bandwidth. Furthermore, it can be shown[9] that

$$BW = M\left[1 - \left(1 - \frac{U_p r_{eq}}{M}\right)^N \left(1 - \frac{1}{M}\left(1 - \left(1 - \frac{1 - U_p}{M}\right)^N\right)\right)^M\right].$$

The above two equations for BW can be solved by iterating on U_p, with $N = 4$ and $M = 8$ (two or three iterations are usually sufficient). The resulting value for BW is the memory bandwidth of the system, assuming that the connection time is C' bus cycles rather than C bus cycles. Therefore, the true memory bandwidth of the multiprocessor system described above, BW_{true}, is given by

$$BW_{true} = \frac{C}{C'} BW.$$

To test the validity of the approximate model described above, the results of the model are compared to those obtained by simulation. A SIMSCRIPT II.5 simulation program has been used to simulate the multiprocessor system. Table I shows the simulation and the model results for the memory bandwidth, given different values of the request rate, r. In addition, the simulated memory bandwidth is compared to the ideal memory bandwidth of the system, BW_{ideal}, which is defined as follows:

$$BW_{ideal} = 4 \times \frac{C}{T + C}$$

In other words, BW_{ideal} is the memory bandwidth obtained assuming that each of the four processors operates independently without interference during memory accesses. Hence, BW_{ideal} can be thought of as the maximum potential memory bandwidth for the multiprocessor system. The percentage difference between the simulated BW and BW_{ideal} is also shown in Table I. *%Diff* is defined as follows:

$$\%Diff = \frac{BW_{ideal} - Simulated\ BW}{BW_{ideal}} \times 100$$

The simulation used priorities determined by the prioritizer logic in the ICN's, nevertheless the model, in which all priorities are equal, produces similar results to simulation. This arises because the queueing discipline for memory modules does not affect BW. The results presented in Table I show, among other things, that the performance of the multiprocessor system is degraded in the region where r is small. Since the memory conflicts are minimal in this case this degradation is almost entirely the result of the setup time. By comparing with cases where setup is fixed at zero, it can be shown that the effects of the memory conflicts become more critical as r increases. However, even in the case $r = 1$ the degradation due to interference never rises above 12%, i.e., about 33% of the degradation is due solely to setup (see last entry in Table I). Finally, measurements performed on a number of 8086 systems indicate that $r = 0.4$ is a frequent operating point.

TABLE I—Comparisons between the simulation and the model results

r	Simulated BW	BW_{true}	%Diff
0.1	1.04230	1.03091	15.31
0.2	1.49357	1.47896	25.32
0.3	1.72489	1.71427	31.72
0.4	1.86074	1.85618	36.04
0.5	1.95659	1.95026	38.86
0.6	2.02529	2.01692	40.39
0.7	2.07624	2.06653	42.53
0.8	2.12421	2.10483	43.58
0.9	2.14873	2.13527	44.79
1.0	2.17244	2.16003	45.69

CONCLUSION

In this paper we have presented a technique for constructing MIMD multiprocessors using a 3 bit slice component, the ICN chip. A performance model was presented that showed close agreement with simulation, and thus would make a useful design tool for estimating crossbar performance. The main measure of crossbar efficiency, BW, was shown to be dependent mainly on the setup time in the case of a four-processor/eight-memory system. In the case where processors have local memory accessed through a resident bus the values for r (the request rate to the shared memory) are likely to be very small if the hit rate to local memory is high. In such systems a concentrator ($N > M$) version of the ICN chip would be more appropriate.

ACKNOWLEDGMENT

The work described in this paper was supported in part by Air Force Contract No. F49620-82-C-0089.

REFERENCES

1. Makrucki, B. A., and T. N. Mudge. "VLSI Design of a Crossbar Switch." SEL Report No. 149, Department of Electrical and Computer Engineering, University of Michigan, January 1981.

2. McFarling, S., J. L. Turney, and T. N. Mudge. "VLSI Crossbar Design Version Two." CRL Report CRL-TR-8-1982, Department of Electrical and Computer Engineering, University of Michigan, February 1982.
3. Wu, C. W. C. "Interconnection Networks," *Computer*, 14 (1981), 12, 8–9.
4. Kruskal, C. P., and M. Snir. "The Importance of Being Square." *Proceedings of the 11th Annual International Symposium on Computer Architecture*, IEEE Computer Society, Ann Arbor, MI, 1984, pp. 91–98.
5. Franklin, M. A. "VLSI Performance Comparison of Banyan and Crossbar Connection Networks." *Proceedings of Workshop on Interconnection Networks*, 1980, pp. 20–28.
6. *iAPX 86,88 User's Manual*. Santa Clara: Intel Corporation, July 1981, pp. A.111–A.134.
7. Skinner, C. E., and J. R. Asher. "Effects of Storage Contention on System Performance." *IBM Systems Journal, 8* (1969), 4, pp. 319–333.
8. Mudge, T. N., and H. B. Al-Sadoun. "Memory Interference Models with Variable Interconnection Time." *IEEE Transactions on Computers, C-33* (1984), pp. 1033.
9. Yen, D. W. L., J. H. Patel, and E. S. Davidson. "Memory Interference in Synchronous Multiprocessor Systems." *IEEE Transactions on Computers, C-31* (1982), pp. 1116–1121.