LSI and minicomputer system architecture

by L. Seligman

Data General Corporation
Southboro, Massachusetts

INTRODUCTION

The direct impact of Large Scale Integration (LSI) on minicomputer system architecture has been and will continue to be evolutionary and incremental, not revolutionary.

LSI has been applied to minicomputers most effectively in the form of an incrementally improved technology. When well understood, it has offered predictable cost reductions and performance improvements. It has been most successful commercially when combined with other factors such as improved core memory technology and imaginative new approaches to minicomputer packaging. While the combination of these factors has been steadily driving prices down and performance and reliability up, we are not likely to see the bottom drop out of prices. There will not be a full scale minicomputer in every gas pump in the near future; nor will there be such dramatic improvements in minicomputer performance that minis will replace large System 360s one for one.

INDUSTRY MATURITY

Although technological advances, including increased use of LSI, have improved both price and performance, the recent maturing of the industry has had a more dramatic impact on the minicomputer market; thus, it has also had an indirect impact on the technological base of the industry.

A few years ago, there were many companies selling minis. The cost of entering the market was low, and many of these companies enjoyed some initial success. In the last few years, however, price competition has increased significantly, and the number of manufacturers has shrunk. Concurrently, the OEM minicomputer market has grown, such that, while there are many fewer companies in this business today, those who remain are each manufacturing thousands of units a year. These large minicomputer manufacturers are necessarily becoming very manufacturing-oriented. Their production methodology is growing to resemble that used in producing consumer electronics. Indeed, as Figure 1 shows, a modern mini with 32,000 16-bit words of memory and several peripheral controllers is about the same size as a stereo receiver.

Volume production of such a complex product has been facilitated by the improved reliability provided by the new technology. In fact, the greatly increased reliability of the modern mini, based in part on the use of LSI, has been a major factor in its proliferation, especially in real-time systems.

The basic cycle of a maturing industry—larger market, fewer manufacturers, better product, higher volume, increased manufacturing orientation—has naturally driven prices down and opened new application areas.

The indirect impact of LSI grows out of the business cycle as each of the minicomputer manufacturers remaining use many more integrated circuits. The successful manufacture of complex integrated circuitry requires high volume to obtain the high yields required to achieve profitability. Without large volumes, LSI manufacture is burdened with high prices, low yields, stalled development programs, and money-losing sales efforts. However, as minicomputers begin to use more LSI—more because the product design calls for more LSI per unit, and because more units are being made—large IC production develops. At this point the snowball effect begins to work beneficially. There are large circuits requirements, higher yield, lower costs, larger orders. This process is very much in keeping with the evolutionary character of changes in mini price/performance. That is, the manufacturers are increasing their use of LSI technology in predictable, evolutionary way, and not as a means to a breakthrough.

A third element in the growing maturity of the industry has been the diffusion of computer-oriented professionals into most industries. Many of the students
who had hands-on experience with minicomputers in
the university environment will apply this experience
to the development of minicomputer based systems for
specific applications that had not previously used com­
puters, and they contribute a wealth of ideas to the in­
dustry itself.

EFFECT OF LSI

While the result of these technological and business
developments has been an evolutionary improvement
in the price, performance, and reliability of minicom­
puters, the effect of the improvement has been revolu­
tionary in one important area—the basic architecture
of medium-scale computer-based systems.

There have been two elements in the revolution.
First, the minis are real and system designers can trust
them. They have become inexpensive, so inexpensive
that they are impossible to ignore; and they are re­
liable, largely because they contain far fewer com­
ponents and are built in long production runs. For those
applications with emphasis on logical decision making,
their performance is comparable to large-scale systems.¹

Thus, system designers are using minicomputers in
new applications such as front ends or preprocessors for
large business data processing installations; at various
levels in hierarchical communications systems; in small,
dedicated accounting systems; and in groups in real­
time control applications.

The second part of the revolution is more interesting
and is developed in detail in this paper. As the minis
have become faster, more reliable, and less expensive,
they have forced changes in the traditional perception
of a computer system. Today most minicomputer pro­
cessors comprise a relatively small part of the cost of
the system. Peripheral devices and their controllers,
the system hardware package, and the software to run
it all costs as much or more than the basic CPU/
memory package.

It no longer makes sense to hold a strictly processor
centered view nor does it make economic sense to base
all design decisions on a requirement to keep the central
processor busy all the time. The low total costs of
modern minis make multiprocessor minicomputer sys­
tems a practical alternative to large multitask mono­
processors, especially in real-time systems which can be
partitioned into individual functional tasks.

MULTIPROCESSOR CONFIGURATIONS

The use of multiple minicomputers, each performing
some relatively independent function as part of an
overall system, is a natural extension of the concept of
the dedicated realtime minicomputer that has evolved
over the past several years. This technique is in sharp
contrast to the traditional approach to a real-time ap­
plication, which consists of dropping the entire time­
critical package onto a single processor. The alternative
approach outlined here substitutes an individual mini­
computer for each of the natural functional subsystems
of a large-scale application, interconnecting the mini­
processors through an adequate communications path.
The “traditional” or general multiprocessor system
is a cross-connected network of processors, memory,
and I/O controllers (channels); this configuration

![Figure 2a—Traditional multiprocessor](From the collection of the Computer History Museum (www.computerhistory.org))
shown in Figure 2, is widely described in the literature and is most often proposed for the multitask environment.

A number of problems have arisen in the completion of the development programs for these systems. Very high utilization is considered imperative for these systems, and a sophisticated multiprogramming executive is required to keep the processor busy. Use of a large executive control program is acceptable so long as the costs of resource allocation and optimization do not exceed the savings which can be attributed to its use. This goal has not always been realized, although a good deal of research has been done on "balanced" resource allocation. Minicomputer multiprocessor systems will inherit similar problems if configured in the traditional patterns. In addition, these problems would have to be solved in light of the particular constraints imposed by minicomputers, especially the amount of software development resources which can be applied economically.

While there are a number of these traditional multiprocessor systems in use today, many more multiprocessor systems are organized in the form of a main processor and one or more smaller support processors. The processor cost in each system is often sufficiently low compared to the cost of the peripherals required to support the database that it is economical to increase capacity with additional processors.

This associated support processor concept is less elegant, but it has been more successful commercially, mainly because the program development efforts required to make it operational are within the resources of the various manufacturers. The support processors function primarily in the I/O area, freeing the main processor to perform computational functions, while communications, file maintenance, and most I/O activities are handled elsewhere. Even without an explicitly programmed support processor, a typical large scale system is a multiprocessor in that the data channels and some of the peripheral subsystems (especially the disk) contain a substantial instruction processing capacity.

The increased use of LSI imposes some new economic constraints, especially in regards to multiprocessor configurations, on minicomputer systems engineering. The cost of the central processing or memory units are so low that one is forced to view them as modules. In fact, in the case of the Nova 1200 series, as shown in Figure 3, the processor is a single replaceable printed circuit card as is an 8192-word memory module. Other modules, for example a magnetic tape or disk controller, are about equally complex as a processor. Since the
manufacturing volume of the processor module is higher than the others, its sales price can be lower. Thus, the use of multiple processors need not add significantly to the system cost.

Once it is decided that a multiple processor approach is the most suitable for a particular application, the intramodule connection facility must be selected. Due to the small physical size and low cost of the modules, the interconnection facility and the mechanical package(s) must be low in cost or much of the potential economic advantage of LSI is lost. Yet, the interconnection facility must support the extremely high combined data rates found at the processor-memory interfaces.

The generalized crossbar interconnection scheme used in the traditional multiprocessor does not meet these needs. Nor does there seem to be a general solution to the multiprocessor system design problem. One does not simply replace a single $200,000 large scale processor with 50 or fewer $4,000 minis, however tempting. However, a number of specialized minicomputer systems have been developed with the concept of the mini as a system element, and some general principles can be abstracted from these designs. As implemented on the minis, none are multiprocessing systems but by necessity they would have been if they had been implemented on a one or two processor large scale machine. They are all organized in a form similar to the associative support processor concept. A processor and sufficient storage to hold its application program and data buffers form a module. Several of these modules are connected via a high speed interconnection bus, so that data and control information can be passed among the several functional programs.

THE PROCESSOR/STORAGE MODULE

The processor/storage module approach of Figure 4 (PSM) calls for the loose interconnection of a number of such modules into a single system according to a partition of the overall application into functional tasks. The modules operate independently but are tied together via an interconnection bus that allows a number of concurrent intermodule logical communications links to be established. Processing is thereby distributed across the system rather than concentrated at a central point.

The logical basis of the PSM approach is “locality”, which is the observed phenomenon that the memory address patterns generated by a processor are not random but exhibit a strong clustering. This principle has led to the development of many storage hierarchy systems ranging from cache memories in modern large scale machines to various overlay systems like the chaining feature in Fortran to demand paging timesharing systems. The success or failure of these various approaches depends on the relationship between the amount of locality encountered and the overhead incurred on nonlocal references. The general problem is difficult to model mathematically, although in recent years several approaches show promise.

Locality requires, especially in real time systems, that certain portions of the program be resident in primary (low access time, executable) memory; other portions can be brought in as needed. Minicomputer economics are such that each memory partition might just as well have a processor associated with it. One would then think of the memory as defining a function, specializing a processor for a task just as read-only control memory has been used to define the function performed by microprogrammable controllers.

The interconnected processor/storage module organization described here constrains the programs written to a form that is highly local. Like chaining in Fortran, intermodule references are quite explicit and, hence, visible to the individual programmer who most optimally partition the program.

Not all of the individual minicomputer subprograms need come directly from a partitioning of the task. A PSM can be used as a flexible, programmable controller to enhance the performance of peripheral equipment. In the message switching system example below, the disk control program and associated processor, together with a simplified hardware controller, perform many of the more sophisticated key search functions that would, in a large scale system, be performed in part by the file controller itself. The major advantage of the mini as a controller is that it can be programmed by the user for its particular task rather than micro-programmed at

![Figure 4—Processor/storage module system organization](From the collection of the Computer History Museum (www.computerhistory.org))
the factory for a generalized task. It is far easier to modify a factory supplied operating system than to alter firmware.

In addition, programs can be swapped in and out of a physical memory through the inter-module communication facility. In such a situation, one module would have secondary storage associated with it and would distribute programs on request (or by some scheduling algorithm) to other modules. In essence, it would act as a (micro-) programmable multi-port disk controller serving many processors and would perform certain system level executive functions.

The hardware actually used to interconnect the processor is termed a multiprocessor communications adapter (MCA). One MCA is attached to the I/O bus of each computer in the system, and the adapters are connected together by a common communication bus which is time-division multiplexed among the adapters. Although a single circuit module, an MCA actually contains independent receiver and transmitter subsections, allowing simultaneous reception and transmission of data. Each interface is connected separately to the data channel. The program need only set up an interface for receiving or sending, and all transfers to and from memory are then handled automatically by the channel hardware. A processor with an adapter can establish a link between its transmitter and any receiver it designates, provided that the receiver adapter has been initialized for reception.

A number of logical links concurrently share the single communications bus using time-partitioning multiplexer circuitry built into the adapters. If there are N logical links established and communications is proceeding on all, each link receives 1/N of the communications bus time. The bandwidth of the bus is 500 KHz (1 million bytes per second). However, this rate will only be obtained when a large number of links are active concurrently as data rates are primarily determined by the processor’s channel facilities. The typical data rate on a single link for a pair of Nova-line computers with high speed data channel feature is 150 KHz.

Each adapter has a unique identifying number assigned to it. These codes are used to specify the number of a second adapter to which the first is to be logically connected. Upon receipt of the first data word from some transmitting adapter, the identifying number of the transmitter is set into the receiver adapter status register; the receiver will subsequently accept further data only from the transmitter, i.e., it “locks” to that transmitter. It must be explicitly unlocked by the receiving processor’s program.

A number of the specifications of the MCA arise from the need for the whole system to remain functional despite a hardware or software failure. The transmitter registers must be initialized by the transmitting processor; and the receiver registers must be initialized by the receiving processor. Thus, a transmitter cannot force data into a receiving processor at addresses not specified by the receiver. Transmission of a block terminates when the number of words transferred satisfies the transmitter or receiver with the shorter word count. In addition, a receiving processor is not only protected against a failing transmitter as described above, but the hardware is arranged so that any of the interconnected computers can be stopped or have their power switched off without affecting the other computers still in operation. If a processor adapter attempts to transmit data to an unavailable receiver adapter, a timeout interrupt will occur after a delay of approximately 10 milliseconds. If the receiver is unavailable because it is linked to another transmitter, the transmitter may be restarted for further attempts or the data may be routed to a different receiver.

The size and nature of the data transmission can follow any convention established by the user; no particular structure is forced by the hardware design. In a relatively simple system in which the size and nature of the data block to be transferred is always known in advance, the receiver can simply initialize itself to accept the next block at the completion of the previous transmission.

If the exact size and nature of the data blocks is determined dynamically, a control block specifying the nature of a transfer can be transmitted before the actual data block. With such a convention, the receiver initializes itself to accept a control block of standard format and unlocks itself. The first word transferred to the receiver locks it to the sending transmitter by setting the transmitter’s code into its status register, locking it to that adapter transmitter until explicitly unlocked by the program. Thus, once the first word in a control block is accepted, the receiver is locked to that transmitter and can be initialized to accept subsequent data blocks from the appropriate transmitter.

Alternatively, the control block from adapter A to adapter B can be a request for data. Adapter B’s transmitter can be started sending the desired data while its receiver is reinitialized to accept a new control block. The hardware itself does not distinguish between a data and a control block.

The first sample system using the PSM approach is shown in Figure 5. The function of the system is small scale message switching and it uses the separate control and data block technique. Separate processors handle: (1) interfacing to synchronous and asynchronous lines, (2) disk queuing, and (3) executive and journal func-
The journal and executive/monitoring functions are traditional. As the disk is basically organized as a ring buffer, older messages can be copied from it onto tape during the system's idle periods in order to keep as much space as possible available on disk. A second tape keeps a permanent record, including sequence numbers, terminal ID's, etc., of system activity. The executive function is primarily active in case of error, rebuilding the system after error through journal tapes; it also responds to system level messages, such as requests for retransmission of an earlier message.

The significant point is that new functions and applications modules are not limited by total computational capacity, because additional processor modules can be added as necessary.

A second example of the PSM approach is the experimental multiprocessor small computer system developed at the Plessey Laboratories for the control of a small telephone exchange. The design is inherently extensible and allows for better performance for both voice and data users since the processing capability is distributed among the several processor/memory modules, the number of which can be varied according to the size of the switching network being controlled. The modules are interconnected via a "ring highway" data transmission path similar to the previously described multiprocessor communications adapter.

The organization of the control programs for the multiprocessor complex is almost identical to the organization used in the (essentially) single processor telephone exchange control developed at Bell Laboratories.

In both cases, large tables stored in memory define the state of the switching network itself, and individual, table-driven functional programs are run to implement the various stages in the completion of a call. The major difference between the two approaches is that a spatial distribution of programs across several processors is used in the former, while a time division of the processor is used in the Bell System No. 1 ESS. The amount of interprogram data movement is proportional to the number of calls that can be completed per second. This number is relatively small, since it is limited by the electromechanical nature of the switching equipment.

A detailed description of the functions performed is beyond the scope of this paper, but such tasks as dial pules accumulation, network computation, billing, and trunk signaling are typical.

A second important distinction between the Plessey and Bell systems results from different goals influencing the tradeoffs between the costs and benefits of redundancy and error detection. The micro-synchronism of the two computers used in No. 1 ESS allows for the
immediate detection of hardware errors at the cost of doubling the amount of hardware required. The PSM approach allows as few as one additional module to be used for lower cost redundancy but does not provide as good error detection. Dual micro-synchronous processors could, of course, be used as modules in the PSM approach which is probably less susceptible to software errors. In addition, many of the real-time problems encountered in a time-division, traditional multitask system can be avoided because modules can be added as the traffic load increases. In fact, as in the message switching system, the modular design facilitates the duplication of a module or application program when use of that program exceeds the available subsystem capacity.

SUMMARY

A number of technical and economic arguments have been presented for the use of multiprocessor minicomputer systems. It is hoped that the examples given will spur the development of a methodology for system design.

REFERENCES

1 A KAY
Computer structures: Past, present and future
AFIPS 1971 Fall Joint Computer Conference Vol 39
pp 395-396
2 F J CORBATO V A VYSSOTSKY
Introduction and overview of the multics system
AFIPS 1965 Fall Joint Computer Conference Vol 27
pp 185-196
3 R J KAY
The management of large scale software development projects
AFIPS 1969 Spring Joint Computer Conference Vol 34
pp 425-433
4 P J DENNING
Resource allocation in multiprocess computer systems
PhD thesis Massachusetts Institute of Technology 1968
5 C J CONTI
Concepts for buffer storage
6 J R POLLARD
Putting the computer into the telephone exchange
Computer Decisions June 1970 pp 34-37