Although a technology may be familiar to most engineers, it may not find its way into their designs until long after it has been developed. Here is how several such technologies were used in one large computer

Building Today's Technologies Into a Large-Scale Time-Sharing System

Gerald F. Atterbury

Digital Equipment Corporation
Large Computer Group
Marlboro, Massachusetts

George F. Adams

Motorola Semiconductor Products Incorporated
Computer Applications Group
Lexington, Massachusetts

Several sophisticated computer-related technologies are receiving a relatively high degree of attention today. Although most have been known for several years, they have not been widely utilized by computer designers until recently. These include fast emitter-coupled logic (ECL), cache memory, and use of a separate console processor.

For example, the design of the KL10 central processor as part of the DECsystem 1080 was begun in late 1972 when no machine was employing commercially available ECL. As a relatively unknown performer in large-scale use, ECL [Fig. 1(a)] required tradeoffs to be made against the more popular Schottky transistor-transistor logic (TTL), as well as new interconnection rules. The latter were necessary for successfully implementing either ECL or Schottky TTL while retaining the techniques of a conventional production environment. System speeds for both Schottky and ECL are dependent on how these interconnections are made and how well each technology's capabilities are understood.

One particularly significant development in several recent machines—including the KL10—is the cache memory, which minimizes reference time. It is, in general, a small block of high speed memory—usually semiconductor—interposed between the processor and main

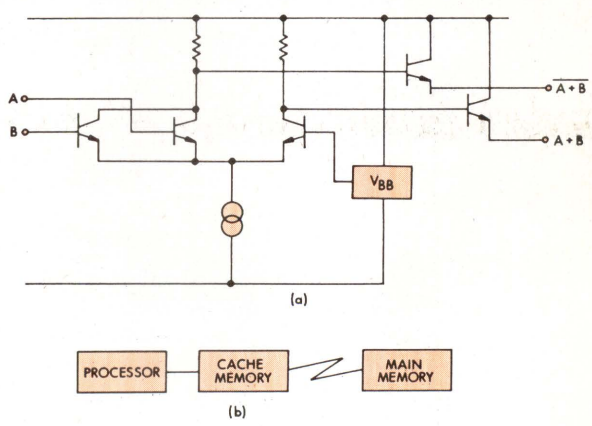


Fig. 1 Two technological advances. Among state-of-the-art technologies that boost system performance are (a) emitter-coupled-logic integrated cirtuits and (b) use of cache memory, especially when storing data in main memory

How Emitter-Coupled Logic Evolved

Various forms of ECL have been available for a number of years. Standard families have been obtainable since 1962, and custom circuits have been used in several large computer systems. IBM, for example, uses a custom family in its Systems 360 and 370 computers. RCA's former computer division (now part of Sperry Univac) uses other custom families in its Spectra 70 series.

ECL was among the first digital integrated circuit lines produced. One of these was Motorola's MECL I, introduced in 1962, having 8-ns gate propagation delays and 30-MHz flip-flop toggle rates, and dissipating 31 mW/gate. These characteristics placed it at the state-of-the-art for standard logic family performance at that time. However, most systems were not ready for 8-ns gate speeds, and several system-oriented features of MECL I prohibited its widespread use. For instance, it required a separately packaged bias voltage driver circuit, which in turn required extra circuit board wiring. Another limitation was its 10-pin package, in which only logic functions at the individual gate and flip-flop level could fit. In addition, it was not designed to operate with transmission lines, which were not necessary for ordinary logic interconnections when the circuits' rise and fall times were as long as 8 ns, but were required for long interconnecting lines, together with special drivers and receivers. In spite of these system constraints, MECL I still found usage in many specialized products and is still being produced.

The more advanced MECL II family, introduced in 1966, was also produced by other semiconductor manufacturers, including Signetics, Plessey, and Stewart-Warner. It included gates with 4-ns propagation delay and flip-flops that would toggle at over 70 MHz. Although its power levels were not lowered (30 to 35 mW/gate) and special drivers were still required for 50-Ω transmission lines, the bias generators for MECL II were internal, and more complex functions such as adders, multiplexers, and decoders became available.

The continuing development of ECL made possible an even faster family, with 1-ns delays and counter rates greater than 1 GHz. Announced in 1968 under the name MECL III, it still represents the fastest performance of any standard logic family. However, its fast rise and fall times required a transmission line environment for all but the smallest systems. For this reason, all MECL III outputs were designed to drive transmission lines, while internal 50-kΩ pulldown resistors were included on the circuit inputs so that unused inputs would not have to be connected to ground. Although the power levels of MECL III (60 mW/Gate) and its fast rise and fall times made its use in large systems difficult, it was widely used in fast instrumentation and communications equipment.

Nevertheless, in spite of their speed and system-oriented improvements, MECL II and III did not find widespread usage in large systems. In the late 1960s, standard TTL logic families met the speed requirements of most systems. By 1970, the TTL price war made the cost of MECL II and III seem relatively high; this, together with these families' system constraints (tighter than standard TTL), contributed to their apparent unpopularity. Then, Texas Instruments introduced Schottky TTL, the gate speeds of which were comparable to those of MECL II. Its direct compatibility with standard TTL made it the choice for many system designers who required 4-ns gates.

By 1971, however, trends in large systems demanded a logic family that was faster than Schottky TTL or MECL II, yet not subject to the system constraints imposed by the speed of MECL III. These circuits, of which MECL 10,000 is an example, were comparatively easy to use, because rise and fall times were slowed to 3.5 ns, while gate propagation delay was held at 2 ns. These rise and fall times permitted the use of standard circuit board packaging techniques, but the circuits could also drive transmission lines where that level of performance was needed. These circuits also dissipated less power than other ECL families—typically 25 mW/gate. This, together with advanced circuit design techniques, led to a new level of complexity in large-scale integrated ECL circuits, ie, inclusion of multiplexers, 4-bit arithmetic units, random-access or programmable read-only memories, 4-bit-by-2-bit multiplier, and a 4-bit slice (coming soon). This family's logic levels were directly compatible with those of MECL III, permitting redesign of critical timing chains in systems that used MECL 10,000 to upgrade them to higher performance levels of later generations of the system.

The circuit family is available from several major semiconductor houses, including TI, Fairchild, and Signetics. It has been used in numerous successful systems including the DECsystem 1080, which is state-of-the-art in machine architecture as well as in its logic family.

memory [Fig. 1(b)]. It holds selected information from main memory, chosen so that the instruction and data which the processor needs are usually in the cache. It boosts system performance because its speed exceeds that of the main memory and because (in large systems, at least) it is usually packaged in close proximity to the processor and therefore is not subject to delays that can occur in a long cable extending between a processor and its main memory when widely separated.

Another important idea is the use of a small console processor, which, in addition to running the console, handles a selection of peripherals and serves as a powerful diagnostic tool.

Performance Tradeoffs

An initial performance goal for any new machine can often be attained with either of two logic families: ECL or Schottky TTL. The choice between them is based on other considerations; for example, if ECL is used, its faster basic gate speed suggests that less complicated logic is possible.

Instruction-time calculations for the most commonly used instructions show that on the shorter ones, such as add or jump, performance is limited by the length of time required to fetch information from memory, while longer instructions such as multiply, divide, or floating-

point arithmetic are limited by logic circuit speed and are performed at a considerably higher speed on the ECL machine.

Although the cache improves performance of both Schottky and ECL machines, the degree of improvement is higher for the ECL because the basic speed of the logic is not masked by the memory. With the scientific Gibson mix of instructions, an overall comparison shows that, relative to the speed of an earlier reference machine (the KI10 processor of the DECsystem 10), the Schottky operates 1.75 times faster, the ECL 3.2 times faster. Thus the ECL machine operates at 1.8 times the speed of the Schottky—in itself a substantial improvement. Without a cache memory, however, the Schottky machine has a performance level of only 0.91, while the ECL's is 1.0—the same as that of the reference machine.

Choice of Technology

Selecting a logic family involves careful analysis of tradeoffs concerning maximum performance and minimum cost as well as usage constraints of each logic type. To minimize cost, the choice should be an industry-standard logic family that is readily available and can be used with established packaging techniques, although performance goals may dictate a medium to high speed logic family. High speed logic turned out to be a better choice in the KL10 because medium speed would have required certain functions to be performed in parallel in order to achieve the performance goals, and parallelism would have required a greater number of integrated circuit packages and a physically larger machine. Most likely candidates among present high speed families are ECL and Schottky TTL.

When designing a large system with high speed logic, interconnection lines exceeding 6 to 8 inches in length must be treated as transmission lines. Standard packaging techniques are consistent with a transmission line environment if the line impedance does not exceed about 80 Ω. However, impedance of etched lines on a printed circuit board depends heavily on their widths and the thickness of the board; these factors prohibit controlled impedances greater than 80 Ω when standard manufacturing methods and tolerances are used. Therefore, the logic family used for any large system design should be compatible with such low impedance transmission environments.

ECL circuit structure is well suited for such a transmission line environment. The devices' specifications are given for driving 50-Ω lines, while the open-emitter outputs provide a low output impedance (7 Ω in either high or low state) and current sufficient for the low impedance lines. Their input impedance is high, producing very little loading on interconnection lines. In contrast, Schottky TTL devices have relatively high output impedances (14 Ω in the low state, 54 Ω in high), which can create system problems.

In either logic family, the output impedance forms a voltage divider with the line impedance, so that the voltage step that is created when the circuit turns on is somewhat lower that its ultimate value. Because the output impedance of the Schottky TTL circuit is high, this voltage divider action creates a relatively small step—lower than the threshold of nearby loads. However, the step travels down along the transmission line and encounters an impedance mismatch at the far end; this mismatch causes a positive reflection which travels back to the point of origin. When it arrives, the full transition at the output of that circuit is achieved.

On the other hand, with a low output impedance, as in the case of ECL, the voltage divider makes a higher step at the output—high enough to switch these nearby loads without waiting for the reflection. Of course, the reflection comes eventually, unless the line is terminated in its characteristic impedance; but by the time it arrives, the transition that initiated it is long past.

The effect this has on system performance can be shown for a typical logic interconnection (Fig. 2). Typically, the length of time needed to drive this circuit configuration (which occurs frequently in systems) including a 12-in., 68-Ω line with an ECL driver, is 4.7 ns, whereas a Schottky gate requires 13.7 ns. Thus the ECL delay is only slightly over one-third that of the equivalent Schottky delay—better than predicted on data sheets, which specify 50% less delay. This difference is more pronounced on longer signal lines.

System noise immunity is a parameter of major importance. Both the susceptibility of logic to noise and the generation of noise by the logic—particularly when switching—must be considered. The dc noise margin depends on both power supply voltage and temperature differences between driving and receiving gates. ECL's dc noise margin is smaller than that of Schottky TTL, but dc conditions alone are not sufficient for successful system design; noise margin is also affected by the rate of change of signal edges and the magnitude of signal swing. System noise performance of ECL is superior in the worst case to that of Schottky TTL.

Logic power dissipation affects both system reliability and cost. Of course, reducing ambient temperatures on logic modules reduces the failure rate of active devices, and the logic module temperature is directly related to logic power dissipation. Coping with high power dissipation means increases in power supply, power busing, power line bypassing, and cooling—all of which cost more than they would for a logic family with lower power dissipation—other factors being equal.

Dissipation of a logic gate depends on dc current drain of the gate itself, dc and ac loading on device outputs, and transient currents through the device during switching. In comparing power dissipation figures, ECL power levels must include power dissipated in the line termination resistors. Partly for that reason, the average dc power dissipation of a Schottky gate, which depends on the output logic level, is about one-third less than that of ECL, but increases at higher switching frequencies because of an effect called overlap switching, or totem-pole spiking. Every gate has two transistors in series at its output, in a "totem-pole" configuration that drives current in the output line in both high and low states. In the dc state, only one of these transistors conducts at a time, but as the gate output switches, both transistors conduct simultaneously for a few nanoseconds. Thus Schottky gate power dissipation increases with frequency. At approximately 32 MHz, it equals that of the resistively terminated ECL gate and, at even higher frequencies, the Schottky gate dissipates more power than the ECL gate and its terminator do. The exact crossover frequency depends on the resistor value.

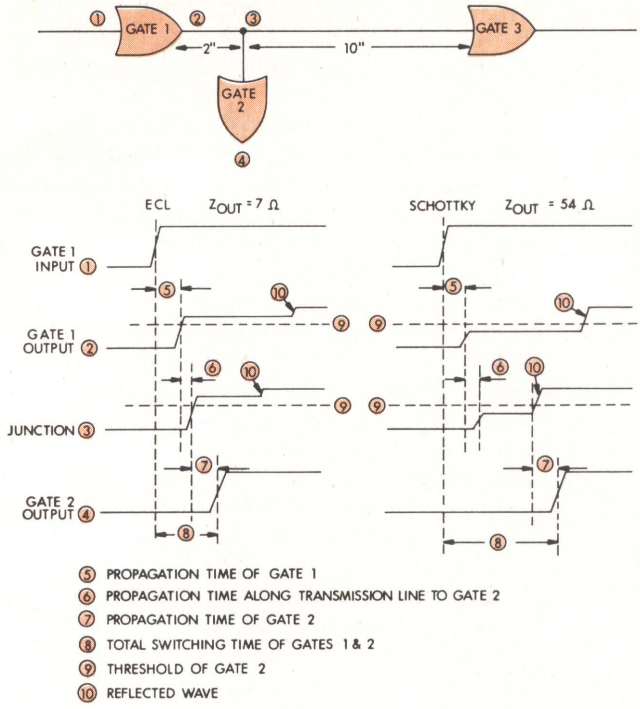


Fig. 2 How output impedance affects switching time. An output voltage step always propagates to the end of the line and is then reflected toward its source. If the source's output impedance is too high, output amplitude may not be sufficient to switch nearby gates before the reflection returns—increasing switching time unacceptably

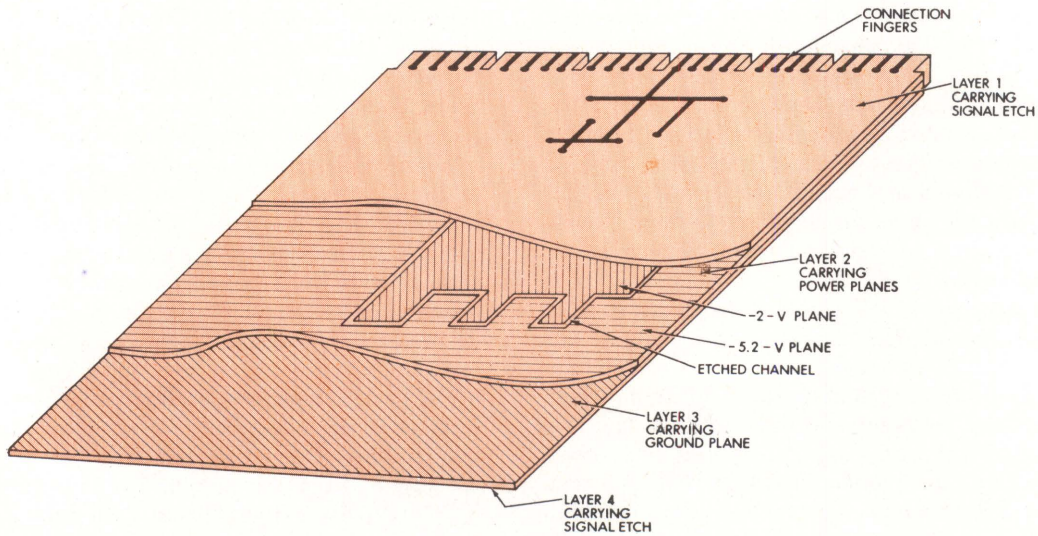


Fig. 3 Four-layer board. Signal lines are routed on the outer surfaces, while power and ground are distributed on the inner layers. Power at two voltages is distributed by dividing one of the two inner layers into two sections

System Packaging

Today's most advanced ECL circuits exhibit typical propagation delays of about 2 ns and rise times of 1 to 3 ns. Application of such high speed logic requires careful attention to crosstalk, reflections, and noise on the power supply and ground lines.

DEC's ECL processor was designed on 4-layer multilayer boards, shown diagrammatically in Fig. 3. These boards permit better component packing density than do double-layer boards because they route only signal lines on the board surfaces and consign power and ground lines to the inner layers. In addition, the impedance of signal lines can be kept close to the selected design value (68 Ω for the KL10—a choice based on the tolerance of the characteristic impedance of the etched lines and the pull-down current that these lines would carry to their termination resistors).

Of the two inner layers, one is a ground layer, while the other is etched into two interdigitated sections to distribute both −5.2 and −2 V on one layer. Since it maximizes the surface area available for signal lines, this is a very practical voltage distribution method, although it does require care in board layout.

Bunching

Because of the way programs are structured, usually any required group of information tends to be in memory locations that are close to information just used. Contributing to this "bunching" tendency is another tendency, ie, for any particular instruction or data word to be reused. As a result, when a word is brought from main memory to cache memory, and then to the processor, a number of words in nearby locations of the main memory can also be brought to the cache and kept there. Once this is done, there is a strong likelihood that it will be called for within a short period of time—at least once and possibly several times; from the cache, this information is available to the processor much more quickly than from main memory. Furthermore, each such reference to information already in the cache saves one reference to main memory.

The result is that information is stored at a cost per bit that exceeds that of main memory only slightly, but with an average access time almost as short as that of the cache. For this to be true, a fairly high degree of success in finding information in the cache is necessary. This success is measured by the hit rate, ie, the number of processor requests satisfied by the cache divided by the total number of processor requests, including those that go to the main memory. Hit rates of 95% or better are achievable.

In the KL10 the cache holds 2048 words. When it becomes necessary to bring new data from main memory to the cache when the cache is full, the least recently used block of words is supplanted by the new data. The old block is usually destroyed; the exceptions, when they are moved back to main memory, are discussed later.

Cycle time of main memories used on the DECsystem 1080 is 1 μs, while the cache memory has an access time of 160 ns. Since the required information is found in the cache in about 95% of the memory references made by the processor, average access time for the cache/main memory combination is $(0.95)(160 \times 10^{-9}) + (0.05)(1 \times 10^{-6}) = 202 \times 10^{-9}$, or 202 ns—an impressive improvement over the straight 1-μs memories.

A New Use for Cache

In the KL10 the cache memory is used in a manner which is different from how it is used in any other machine announced to date. The KL10 both reads and writes in the cache; it does not update the main memory until space is needed in the cache. In other machines using the cache concept, the cache benefits only the reading of data; when new data are stored by the processor, it updates the contents of the addressed location in both cache and main memory. This is not a serious disadvantage, because in most applications, read cycles outnumber writes by 3 or 4 to 1. Thus in the KL10, write cycles benefit from the cache as much as read cycles do.

This innovation is based on the operation of the entire memory system. Main memory is divided logically into "pages" of 512 words, with each page organized as 128 lines of four words. In the user's program, an address occupies one-half word or 18 bits; this is a virtual address assigned by the programmer and subject to modification in order to obtain access to real memory. Modification takes place in the paging logic, which transforms the nine high-order bits of the virtual address into 13 bits that address a page (Fig. 4), while the other nine bits of the virtual address point to a word within a page.

These nine low-order bits address the data in the cache, which is divided into four sections (Fig. 5). Each section contains 128 locations, each holding four words, or one quadword. Thus total capacity of the cache is four sections by 128 locations by four words, or 2048 words of 36 bits each. Each of the four words also has two status bits appended to it. One, called the write bit, shows whether the word was put in the cache by the processor (in which case the original word in main memory is invalid). The other shows whether the word in the main memory has been changed since it was copied into the cache. If it has, the word in the cache is invalid; thus the second status bit is called the valid bit.

Of the nine bits of the unpaged address, seven address one of the 128 locations in each section, while two bits address one of the four words at that location. Because there are four sections, four words—one in each section—are selected by the nine bits.

One section, and therefore one word, is selected by the cache directory, which contains a number of 13-bit address registers. These in turn contain the main-memory page number from which came each quadword stored in the cache. A 2-bit use count for each quadword shows which of them is the least recently addressed and therefore subject to replacement by current data.

When the processor issues a read request, seven bits of the 18-bit virtual address select four of the 128 locations—one in each section of the cache—together with four registers similarly located in the cache directory. Two bits select one word in each of the four cache sections. The remaining nine bits of the virtual address go to the paging logic, which converts them into a 13-bit

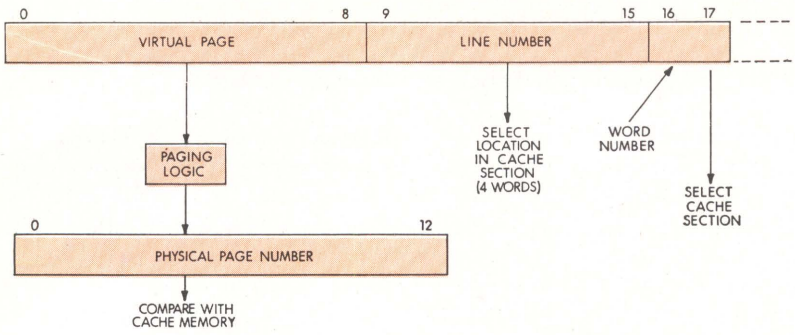


Fig. 4 Address transformation. The KL10 programmer always works with an 18-bit virtual address; its nine high-order bits most be transformed into a physical page number of 13 bits to identify the block of words that actually contains the desired data. Other bits go directly to cache

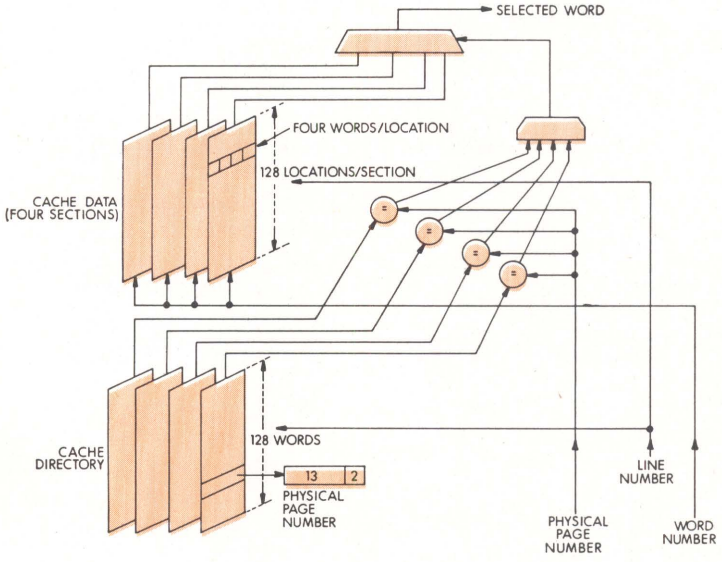


Fig. 5 Cache addressing. Cache memory is divided successively into four sections, 128 locations/section, four words/location. Status bits in each word keep track of independent updating in cache or in main memory, either of which renders the other at least nominally valid. Cache directory identifies the main-memory page, and therefore cache section, in which the desired address is found

physical page number. This is compared simultaneously with the four page numbers coming out of the cache directory.

Only one or none of the four addresses will match the physical page number from the paging logic. If there is a match, the cache directory section from which it came indicates a word from the corresponding section of the cache. If the valid bit of this word is 1, the word is the desired one, and it is transferred to the processor, while the use bits in the cache directory are updated to show that this word is the most recently addressed. If the valid bit of the selected word is 0, a core cycle is taken, updating all words in the quadword which have their valid bits off. Likewise, one or perhaps two core cycles follow if no physical page number from the cache directory matches the one from the paging logic. First, the use bits are checked to find the least recently used quadword. If any write bits in the quadword are 1, these words are written back into main memory, bringing it up to date with the cache. The same quadword is then replaced by a new quadword from the page specified by the paging logic. It is placed in the cache, the use bits are updated, and the desired word is passed to the processor.

Console Processor

Most previous large computer systems have included a conventional operator's console, from which the system is controlled and data entered by means of switches. By this method, an operator could, for example, enter an instruction and data by the appropriate switches, set the clock frequency at a preselected value, and observe and control a sequence of operations directly. He could view the status of all processor registers, including input/output registers and control flip-flops—all displayed by lights.

Now a significant change in operating technique has been made, bypassing operational problems that arise because lights or switches can fail, particularly in ways that may not be obvious. A console processor (Fig. 6) can be designed into the system to replace the lights and switches, and to perform several other functions as well,

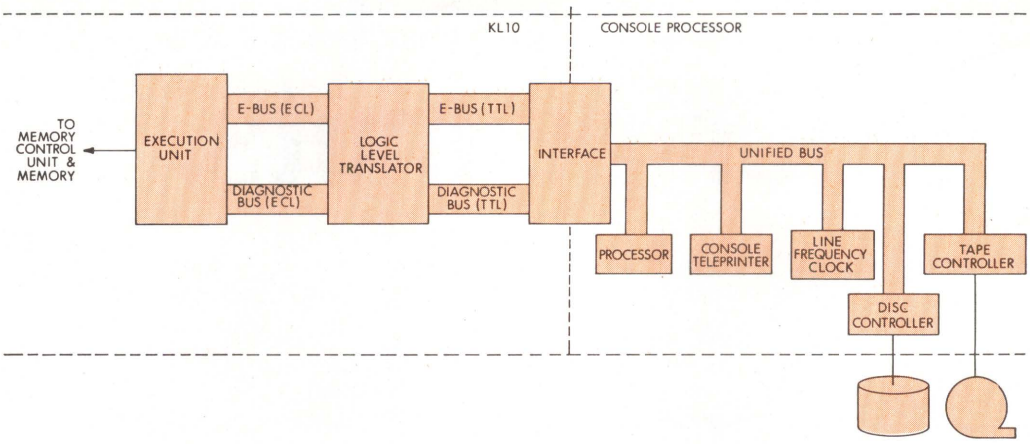


Fig. 6 Console processor. The KL10 has no operator's console in the usual sense; its functions are taken over by a separate minicomputer with its own set of peripherals, communicating with the main processor to control its sequencing and to bootstrap it, change its microprogram, and diagnose its malfunctions

such as loading all diagnostic programs, loading the microcode into the writable control store, bootstrapping the system (starting it from a completely cold or dead start), and providing a long-term power line frequency clock. (In the KL10 the console processor is a DEC PDP-11/40.)

An interactive console command language provides full diagnostic abilities, previously supplied by the lights and switches. With this language an operator can communicate with the system through the console teletypewriter, obtain a readout of the status of any register, and trace the progress of any instruction or data. Furthermore, the processor can control the number of clock pulses before providing a status readout, or it can have the clock make single steps. In effect, the console processor is a system troubleshooter, as well as controller, and can diagnose the central processor and other system components.

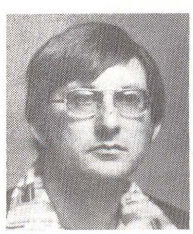
The processor's diagnostic functions include the ability, if a cache failure has occurred, to turn off one or more of the four cache sections. Although processor throughput is thus reduced, the system is gracefully degraded rather than shut down completely. Likewise, for a failure in a section of main memory, that section may be ignored and 1- or 2-word fetches made from the remaining memory sections. This degrades memory bandwidth, as does turning off part of the cache, but allows the system to continue operating.

The processor's access to main memory is limited to two areas which are defined by software. One is written by the main processor and read by the console processor; the other is written by the console processor and read by the main processor. Both areas have their own relocation and protection checks, which are controlled by the main processor. Relocation permits the main processor to move these areas from place to place in the main memory, without consulting the console processor but not affecting the latter's access to them. Protection prevents a user from inadvertently writing in these areas and thus destroying control and status information.

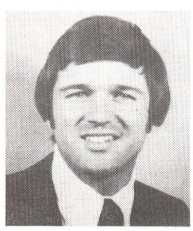
Either processor can interrupt the other, permitting both to carry out their appointed tasks asynchronously.

Conclusion

The system demonstrates that sufficient attention paid to design details allows advanced techniques to be combined with relatively straightforward packaging, fabrication, and manufacturing methods. None of the system's new features compromises the goal of compatibility with earlier systems of its family. It provides a new dimension of growth for large-computer users.



Gerald Atterbury holds a degree in electrical engineering from Brighton College of Technology, England. As Systems Group engineering manager-Large Computer Engineering at Digital Equipment Corp, he is responsible for development of central processors and memories, in addition to circuit design, system packaging, and system interaction problems.



George F. Adams is a field applications engineer with Motorola Semiconductor, working with computer-oriented companies in development of emitter-coupled logic, microprocessors, and memory systems. He is a graduate of the University of Miami.