(20)

A panel session—Computer structure—Past, present and future

Possibilities for Computer Structures 1971*

by C. GORDON BELL and ALLEN NEWELL

Carnegie-Mellon University

What computer structures come into existence in a given epoch depends on the confluence of several factors:

The underlying technology—its speed, cost, reliability, etc.
The structures that have actually been conceived.
The demand for computer systems (in terms of both economics and user influence).

One ignores any of these factors at one's peril. In particular, with technology moving rapidly, a real limitation exists on our ability as designers to discover appropriate structures that exploit the new trade-offs between the various aspects of a computer system.

The design of computer structures is not a systematic art. So new is it, in fact, that in a recent book (Bell and Newell, 1971) we found ourselves dealing with basic issues of notation. We are still a long way from concern with the sort of synthesis procedures that characterize, say, linear circuit design. However, the immaturity is dictated, not so much by youth (after all we have been designing computers for almost 30 years), as by the shifts in technology that continually throw us into previously uninhabited parts of the space of all computer structures. Whatever systematic techniques start to emerge are left behind.

This note comments on several possibilities for computer structures in the next half-decade. Given the unfamiliarity that we all have with the region of computer space into which we are now moving, there can be no systematic coverage. Neither is it appropriate simply to reiterate what would be nice to have. Such an exercise is not responsive to the new constraints that will limit the new designs. Such constraints will certainly continue to exist, no matter how rapidly logic speed rises and logic costs fall. In fact, it is useful to view any prognostication of new computer structures (such as this paper) as an attempt to reveal the nature of the design constraints that will characterize a new epoch of technology.

We will discuss five aspects of computer structures. Mostly, these represent design features that we think have a good possibility of becoming important in the next few years, though we have reservations on one. We have been actively engaged (with others) in working on particular structures of the type we present. Our selection of these is not a denial that other quite different structures might also be strong contenders for dominance during the next several years. Indeed, according to the point made earlier, with strong shifts in technology no one can know much about the real potentialities for new structures. Thus, that we have been working on these particular structures provides, mainly, a guarantee that we have thought hard enough about their particulars to have some feeling for the design limitations in their local vicinity.

Minicomputer multiprocessor structures

Consider the multiprocessor structure of Figure 1. There are p central processors (Pc) and m primary memories (Mp). We ignore, in this discussion, the remaining structure that connects the secondary memories and i/o. The switch (Smp) is effectively a crossbar, which permits any of the processors access to any of the memories.

* The ideas expressed in this presentation have emerged from a number of overlapping design efforts, mostly around CMU and DEC, but occasionally elsewhere (e.g., at Newcastle-on-Tyne, the ARPA list processing machine effort, and the effort at the Stanford AI project). Consistent with this being a short note, we have attempted to indicate the individuals involved in these efforts at appropriate places in the text. But we wish here to acknowledge more generally the contribution of all these individuals. The preparation of this paper was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense (F44620-70-C0107) and is monitored by the Air Force Office of Scientific Research. The paper is to be published in the *Proceedings of the FJCC, 1971* and may not be copied without permission.

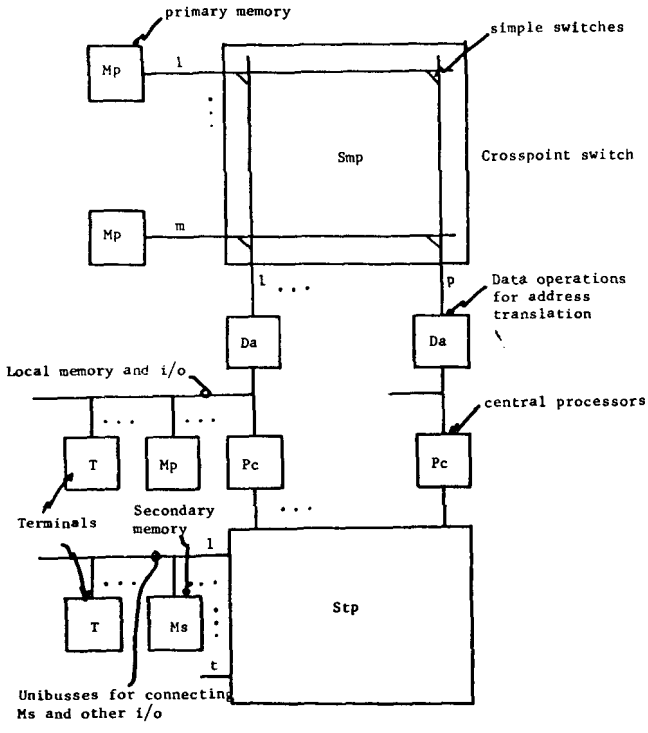

Figure 1—Smp (crosspoint) for connecting p central processors (Pc) from primary memories (Mp)

There is nothing new per se about a multiprocessor structure. Many dual processors exist, as do genuine multiprocessors whose additional processors (beyond one Pc) are functionally specialized to i/o and display. General multiprocessors have been proposed and a very few have come into existence (e.g., the Burroughs D825). But they have not attained any substantial status. The main technological reasons appear to be (1) the cost and reliability of the Smp and (2) the relative cost of many processors. Software (i.e., operating systems) is also a critical difficulty, no doubt, but not one that appears yet to prohibit systems from coming into existence.

Both of these technical factors appear to be changing sufficiently to finally usher in multiprocessor systems of substantial scope. The cost of the processor is changing most rapidly at the minicomputer end of the scale. Thus, we expect to see minicomputer multiprocessors systems before those with large work-length Pc's. An additional impediment for large Pc's is the bandwidth required through the switch, which is substantially less for 16b/w machines than for 32–64b/w machines both in terms of cost and reliability.

As a basis for discussing detailed technical issues, let us describe a multiprocessor system involving the DEC PDP-11. Variant designs of this system have been proposed both at CMU and at Newcastle-on-Tyne.* A set of p PDP-11's have access to a set of m Mp's aggregating 2^{21} 8b bytes.** Each Pc maintains its address space of 2^{16} bytes, but an address mapping component (Da) associated with each Pc permits this address space to be distributed as 2^3 independent pages of 2^{13} bytes each. The details of this addressing, though important, need not be discussed here. Similarly, the details of the Smp need not be discussed. Each link through the Smp is essentially a unibus (the bus of the PDP-11, see Bell et al., 1969). Connections are made on a memory access basis, so that the a PC broadcasts d's address to all Mp's and the connection is made to the recognizing Mp for the data transfer.

The three critical questions about the Smp are its performance, measured in terms of Pc effectiveness, its reliability and its cost. Figure 2 gives the calculated expected performance (Strecker, 1970) in terms of total effective memory cycle access rate of the Pc's (whose number is shown along the abscissa). Each instruction

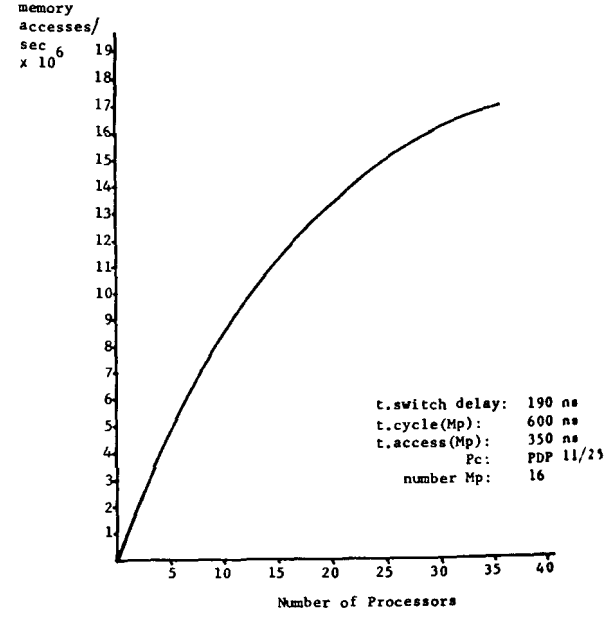

Figure 2—Performance of a multiprocessor computer with 16 independent Mp's.

* The original design was proposed by W. Wulf and W. Broadley, based on a switch design by Bell and Broadley; a second more general design was proposed by C. G. Bell, H. Lauer and B Randall at Newcastle-on-Tyne; the version described here is by C. G. Bell, W. Broadley, S. Rege and W. Wulf. No published descriptions are yet available on any of the designs, though some are in preparation.

** Addressing in the PDP-11 is by bytes, though it is preferable to view it as a 16b machine.

requires one to five memory accesses. The curve is parameterized by the number of Mp's (m=16 here), the cycle of the Mp (350 ns here) and the delay through the switch (190 ns here). The criteria we have used for ideal performance is p stand-alone computers with no switching delays. Thus, the loss is due to both switching delay and multi-Pc interference. The parameters shown are attainable with today's technology. The number of memory references per processor decreases as the number of processors increase, since the calculation assumes a reference to any Mp is equally likely. The reliability cannot yet be estimated accurately, but appears to be adequate, based on a component count. The cost per Pc is of the order of one quarter to one times the Pc, measured in amounts of logic for a 16X16 switch. Thus, the Smp cost is appreciable, but not prohibitive.

What does one obtain with such a structure? Basically, Pc cycles have been traded for (1) access to a larger memory space and (2) Mp-level interprocessor communication. These benefits come in two styles. Statistically, the Smp permits configuration of the Pc's with various amounts of memory and isolation. An important design feature, not stressed above, is that the PDP-11 components remain essentially unmodified, so that they can be moved in and out of the system at will. This feature extends to permitting the addition and extraction of components to the system while in operation. Dynamically, the Smp permits the set of processors to cooperate on various tasks and to decrease the system overhead for input/output and operating systems programs. Coupled with this is common access to the secondary memory and peripheral parts of the systems, permitting substantially lower total system cost as opposed to p independent systems.*

Caches for multiprocessors

A key design parameter in multiprocessor organizations, such as the one above, is the delay through the switch, measured relative to the performance of the Mp's and Pc's. The total instruction (e.g., for a memory access instruction) of a Pc can be partitioned as:

$$\text{t.instruction} = \text{t.Pc} + \text{t.Smp} + \text{t.Mp}$$

In current memory technology overlap is possible between Pc and Mp since accessed information is available before the rewrite cycle is completed. How much this can be exploited in a multiprocessor depends on t.Smp. Thus, the relevant t.Mp is that which would obtain in a non-switched system.

Current technology makes all the above terms comparable, from 50~500 nanoseconds. Thus, variations of a factor of 2 in any of the component terms can have a determining effect on the design. Most important here is that t.Smp can easily become large enough to make t.instruction(with Smp) twice t.instruction(without Smp).

The cache appears to offer a solution to this problem within the currently emerging economic design parameters. The basic concept of a cache is well established.* To review: a cache operates by providing a small high access content addressed memory (M.cache) for recently accessed words. Any reference to Mp first interrogates M.cache to see if the information is there, and only if not is an access made to Mp. The basic statistical regularity of system performance underlying the cache is that words recently accessed will be accessed again. This probability of reaccess depends of course on the size of the past maintained. Available statistics show that if a few thousand words of cache can be kept, then well over 90 percent of the Mp accesses will be found in the cache, rather than having to go to Mp itself. If technology provides a steep trade-off between memory size, memory cycle time and cost per word, then a cache is a valuable structure.

If we associate the cache with the Pc, as in Figure 3, then the net effect of the cache is to decrease t.Pc (for fixed computational power delivered). In organizations

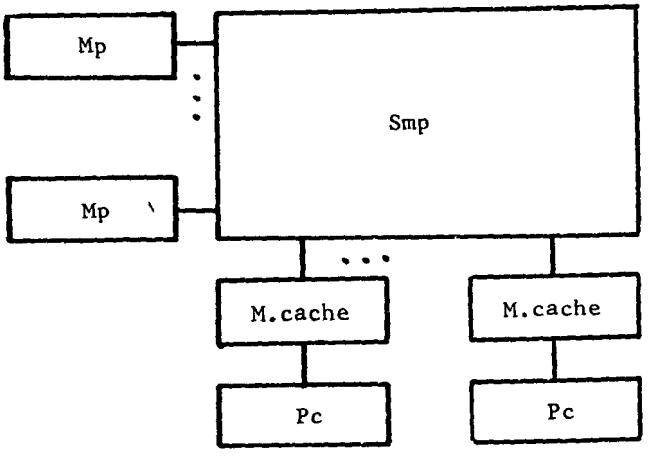

Figure 3—Multiprocessor computer with cache associated with each Pc.

* If this latter goal were all that were required, then one might consider less expensive alternatives. However, a price must be paid in system overhead for less general coupling and the trade-off is far from clear. In fact, we are not justifying the design here, but simply presenting a concrete example.

* The first machine really to use a cache was the 360/85 under the name of "buffer memory" (Conti, 1969). Wilkes (1965) termed it the "look-aside" memory. "Cache" seems by now an accepted designation.

such as the 360/85 this permits balance to be achieved between a fast Pc and a slower Mp. In the case of multiprocessor, this permits the delay of Smp to be of less consequence (for aggregated t.Smp and t.Mp play the same role as does t.Mp in a uniprocessor system).

There is a second strong postive effect of caches in a multiprocessor organization of the kind under discussion. As the graph of Figure 2 shows, performance is a function not only of the delay times, but of the frequency of accessing conflicts. These conflicts are a monotone function of the traffic on the switch, increasing sharply as the traffic increases. The cache on the Pc side of switch operates to decrease this traffic, as well as to avoid the delay times. There is one serious problem regarding the validity of the data in a system such as Figure 3, where multiple instances of data coexist. In a system with p caches and an Mp, it is conceivable that a single address could be assigned p+1 different contents. To avoid this problem by assuring a single valid copy would appear to require a large amount of hardware and time. Alternatively, the burden might be placed on the operating system to provide special instructions both to dump the cache back into Mp and to avoid the cache altogether for certain references.

In a recent attempt to design a large computer for use in artificial intelligence (C.ai), we considered a large multiprocessor system (Bell and Freeman, 1971; Barbacci, Goldberg and Knudsen, 1971; McCracken and Robertson, 1971).* The system is similar to the one in Figure 1; in fact, the essential design of the Smp for the minicomputer-multiprocessor came from the C.ai effort. C.ai differs primarily in having 10–20 large Pc's with performance in the 5×10^7 operation/sec class (e.g., the cache-based Pc being designed at Stanford, which is aimed at 10×Pc(PDP-10) power). An essential requirement for this large multiprocessor was the use of caches for each Pc in the manner indicated.

Why then are caches not needed on the minicomputer-multiprocessor? Interestingly enough, there are three answers. The first is that the performance of minicomputers is sufficiently low, relative to the switch and the Mp, so that reasonable throughput can be obtained without the cache. The second is that the first answer is not quite true for the PDP-11 Pc. To achieve a reasonable balance in our current design requires an upgrading of the bus driving circuits on the PDP-11.** The third answer is that the benefits that accrue from a cache in fact hold for minicomputers as well. Recently a study by Bell, Cassasent and Harris (1971) showed that a system composed of a cache and a fast minicomputer Pc was able to attain a fivefold increase in power over a PDP-8. The cost of the cache was comparable to the Pc, yielding a substantial net gain (i.e., for a minimal system the power increased by 5 while the cost doubled). Thus, caches would undoubtedly further improve the design of Figure 1 at a lower cost. Alternatively, one could simply add more Pc's, rather than increase the cost of the Pc by a cache

Multiple cache processors

One additional design feature of the C.ai is worth mentioning, in addition to its basic multiprocessor structure and cache structure vis-a-vis the Smp.

The general philosophy of the multiprocessor is that of functionally specialized Pc's working into a very large Mp. In the context of artificial intelligence, functional specialization of the entire Pc to a completely specific system (such as the language, Lisp) seems required to exploit algorithm specialization.* Thus, we engaged in the design of two moderate sized Pc's, one for Lisp and one for a system building system called L* (Newell, McCracken, Robertson and DeBenedetti, 1971).

Figure 4 shows the basic PMS organization of one of these processors (actually the one of L*, but it makes little difference to the discussion at hand). The important feature is the use of multiple caches, one for data and one for the microprogram. Two gains are to be obtained from this organization. On the performance side, the gain is essentially a factor of 2, arising from the inherent parallelism that comes from the lockstep between the data and instruction streams. The cache is indicated by the design decision to permit the microcode to be dynamic. Thus, the second gain is in replacing a deliberate system programming organization for changing the microcode with the statistical structure of the cache, thus simplifying considerably the total system organization (including the operating system)

* Many people at CMU participated in the C.ai effort; a list can be found in the reports referenced. Furthermore, the C.ai effort was itself imbedded in a more general design effort initiated by the Information Processing Technology Office of ARPA and was affected by a much wider group.

** Modification of these circuits constitutes the primary modification of the PDP-11 Pc for participation in the system. The only other modification is the use of two bits in the program status word to indicate extended addressing.

* The argument is somewhat complex, involving the fact that specialization to artificial intelligence per se (and in particular to list processing) does not produce much real specialization of hardware. Not until one moves to a completely particular specification of internal data types and interpretation algorithms can effective specialization occur.

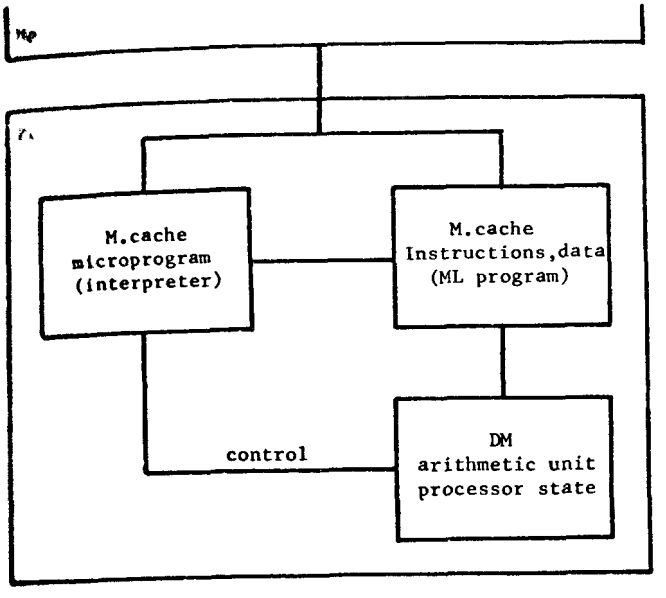

Figure 4—Multiple (two) cache system

The gains here are not overwhelming. But in the light of the many single cache organizations (Conti, 1969) and non-cache dynamic microprogramming organizations (Husson, 1970; Tucker and Flynn, 1971) being proposed, it seems worth pointing out. The concept could be extended to more than two caches in computers that are pipelined, where additional parallelism is available in the controls.

Register transfer modules

Some time ago Wes Clark (1967) proposed a system of organization that he called Macromodules. These traded speed and cost to obtain true Erector set constructability. For a given domain of application, namely sophisticated instrument-oriented laboratory experimentation, a good case could be made that the trade-off was worthwhile. The modules essentially incorporate functions at the register-transfer level of computer structure, thus providing a set of primitives substantially higher than the gates and delays of the logic circuit level.

More recently, another module system has been created, called Register-Transfer-Modules (RTM'S)* (Bell and Grason, 1971). RTM's differ from Macromodules at several design points, being cheaper (a factor of 5), slower (a factor of 2), harder to wire, and more permanent when constructed. On some dimensions (e.g., checkout time) not enough evidence is yet available. Thus, they occupy a different point in a design space of RT modules. For our purposes here these two systems can be taken together to define an approach to a class of computer systems design.

Register transfer modules appear to be highly effective for the realization of complex controls, e.g., instrument controls, tape and disk controls, printer controls, etc. They appear to offer the first real opportunity for a rationalization of the design of these aspects of computer systems. Their strong points are in the rationalization of the control itself and in the flexibility of data structures.

An extremely interesting competition is in the offing between minicomputers and register transfer modules.* As the price of the minicomputer continues to drop, it becomes increasingly possible simply to use an entire C.mini for any control job. The advantages are low cost through standarization and hence mass production. To combat this the modular system has its adaptation to a particular job, especially in the data flow part of the design, thus saving on the total amount of system required and on the time cost of the algorithm.

An important role in this competition is played by memory. If substantial memory is required, its cost becomes an important part of the cost of the total system. An Mp essentially requires a Pc and lo! a minicomputer has been created. Stated another way: a minicomputer is simply a very good way to package a memory. Consequently, RT modules cannot compete with minicomputers in a region of large Mp. This extends to task domains that require very large amounts of control, since currently a memory is the most cost effective way to hold a large amount of control information. Thus, the domain of the RT modules appears to be strongly bounded from above.

An interesting application of the above proposition can be witnessed in the domain of display consoles. First, substantial memory is required to hold the information to be displayed. Thus, in essence, small computers (P.display-Mp) have been associated with displays. A few years back costs were such as to force time-sharing; each P.display serviced several scopes. But the ratio is finally coming down to 1-1, leading to simplification in system organization, due to the elimination of a level of hierarchical structure.

Our argument above, however, has a stronger point. Namely, a minicomputer (namely, a Pc-Mp organization) will dominate as long as there is already the requirement for the memory. Thus, the specialized display processors are giving way to general organizations. In fact, it is as effective to use an off-the-shelf mini-

* Also called the PDP-16 by DEC.

* Actually there may be a third contender, microprogrammed controllers.

computer for the display processor as one specially designed for the purpose. Our own attempt to show this involves a PDP-11 (Bell, Reddy, Pierson and Rosen, 1971).

There is an additional reason for discussing RT modules, beyond their potentiality for becoming a significant computer structure. They appear to offer the impetus for recasting the logic level of computer structure. The register transfer level has slowly been gathering reality as a distinct system level. There appears to be no significant mathematical techniques associated with it. But in fact the same is true of the logic level. All of the synthesis and analysis techniques for sequential and combinatorial circuits are essentially beside the point as far as real design is concerned. Only the ability to evaluate—to compute the output given the input, to compute loadings, etc.—has been important. Besides this, what holds the logic level intact is (1) a comprehensible symbolism, (2) a clear relation of structure to function so a designer can create useful structures with ease, and (3) a direct correspondence between symbolic elements and physical elements.

RT modules appear to have the potential to provide all three of these facilities at the register transfer level (rather than the sequential and combinatorial logic level). The ability to evaluate is already present and has been provided in several simulators (e.g., Darringer, 1969; Chu, 1970). The module systems provide the direct correspondence to physical components, which is the essential new ingredient. But there is also emerging a symbolism with clear function-structure connections, so that design can proceed directly in terms of these components. For the Macomodules of Clark one can actually design directly in terms of the modules. With our RTMs we have been able to adapt the PMS notation (Bell and Newell, 1971) into a highly satisfactory symbolism* It is too early to see clearly whether this conceptual event will take place. If it does, we should see the combinatorial and sequential logic levels shrink to a smaller, perhaps even miniscule, role in computer engineering and science. Actually, even if these modules do not cause such an emphatic shift in digital design, it is almost safe to predict this change solely on the basis of minicomputers and microprogrammed controllers being used for this purpose. This will lead to a decrease in the need for, and interest in, conventional sequential and combinatorial logic design.

A cautionary note on microprogramming

With the right shaped trade-off function on memory speeds, sizes and costs relative to logic, microprogramming becomes a preferred organization, because of the regularity in design, testability and design flexibility that it offers. Memories of 10^5 bits must be available at speeds comparable to logic and at substantially lower cost per effective gate. With only 10^4 bits there is not enough space for the microcode of a large Pc. If memory is too slow or too costly, the resulting Pc's simply cannot compete with conventional hardwired Pc's in terms of computational-power/dollar.

The conditions for microprogramming* first became satisfied with read-only memories (circa 1965). In the first major experiment, the IBM System/360, a variety of hardware was used at different performance levels of the series, all of it M.ro. Some of the memories permitted augmentation, and in fact this feature attained some significant use, e.g., the RUSH system of Alan Babcock (a Joss-like commercial timesharing system based on PL/I) which is able to be both cost effective and interpretive by putting parts of the interpreter into the M.microprogram of the 360/50.

More recently read-write memories have become available at speeds and costs that satisfy the conditions for microprogramming. This leads, almost automatically, to dynamic microprogramming, in which the user is able to modify the microcode under program control. This allows his program to be executed at higher speeds. The effect is not quite to make the microcode the new machine language, for the trade-offs still do not permit $10^6 \sim 10^7$ bits of M.μp, which is required for full sized programs. Thus, the original functional concept of microprogramming remains operative: a programmed interpreter and instruction set for another machine language, which occupies a much larger Mp.

All this story is a rather straightforward illustration of the principle that computer structures are a strong function of the cost-performance trade-offs within a given set of technologies. Different regions in the space of trade-offs lead, not to parametric adjustments in a given invariant computer structure, but to qualitatively different structures.

The cautionary note is the following. In our headlong plunge to discover the new organizations that seem to be effective in a newly emerging trade-off region we must still attempt to separate out the gains to be made from the various aspects of the new system—from the new components, from the newly proposed organizations, etc. The flurry of work in dynamic microprogramming seems to use to be suffering somewhat in this regard. The proposed designs (e.g., see Tucker and Flynn, 1971) appear to be conventional minicomputer

* Called Chartran in DEC marketing terminology. See Bell and Grason (1971) for examples.

* The microprocessor must operate at a speed of 4 to 10 times the processor being interpreted.

...registers with wide unencoded words.* They compare very favorably against existing systems (e.g., members of the 360 series), but when the performance gains are analyzed they appear to be due almost entirely to the ...componentry, rather than to any organizational gains (e.g., Tucker and Flynn, 1971).* The cost of these systems is usually missing in such analyses.

	High performance technology microprocessor	Model 50	Minicomputer
...er of instructions	8	11 (6)	5
...er of bits	512	224 (128)	80
...	10	9 (6)	7
...memory accesses)			
...y bandwidth (megabits/sec)	640~3840	16	16
...for 10 iterations (μsec)	4.3	191	70
...using high performance technology (μsec)	4.3	6.5 (4)	3.5

( ) indicates improvement in coding over Tucker and Flynn.

*There appears also to be some confusion in the application of the term "microprogram" to some of the proposed systems. The definition given by Wilkes (1969) is functional: a microprogrammed PC is one whose internal control is attained by another processor, P.microprogram. Thus, it is the cascade of two processors, one being the interpreter for the other. Certain structural features characterize current P.microprograms: wide words; the nature of the operations (control of RT paths); parallel execution of operations, and explicit next-instruction addressing (to avoid machinery in the P.microprogram). Many of the proposed dynamic programming systems maintain some of the structural features, e.g., wide words, but drop the functional aspect. This is, of course, essentially a terminological matter. However, we do think it would be a pity for the term microprogramming to attach to certain structural features, independent of function, rather than to the functional scheme of cascaded processors, one the interpreter for the other.

*A microprogrammed processor design using 1971 logic and memory technology was compared with IBM's 1964 Solid Logic Technology and core memory used in the 360 Model 50. Since the newer technology (50 nanoseconds/64 bits) was a factor of ..., at 50 faster than the Model 50 (2000 nanoseconds/32 bits) the microprogrammed processor was somewhat (only a factor of ... faster. Even using the faster technology the microprogrammed processor's times for multiplication given by Tucker and Flynn were almost the same as Model 50.

The following table of a Fibonacci number benchmark given by Tucker and Flynn shows that the main advantage of microprogramming is with high performance technology. A microprogrammed processor has about the same number of instructions and number of memory accesses. Due to the poor encoding of instructions a microprogram takes more bits (hence possibly ...). By having a comparatively high memory bandwidth ... execute the loop rapidly, but given a model 50 or a minicomputer constructed with a 50 ns memory the execution times are about the same.

Actually, there are signs of the watchmaker's delusion (Simon, 1969). A watchmaker, Tempus, attempted to construct watches out of very small components, but every time the phone rang with an order he was forced to start over. He got very few watches completed. His friend, Hora, decided first to build springs, releases, escapements, gears, etc., and then larger assemblies of these. Though he, too, was often called to the phone, he quite often had time to complete one of these small assemblies, and then to put these together to obtain an entire watch.

To apply the moral: Large systems can only be built out of components modestly smaller than the final system itself, not directly out of much smaller components. The dynamic microprogramming proposals take as given the same micro-components as have existed priorly (gates and registers). They do not propose any of the intermediate levels of organization that are required to produce a large system. Thus, e.g., when they propose to put operating systems directly in microcode they are close to the watchmaker's delusion. Insofar as the response is "But of course we expect these intermediate levels of organization to exist," then their proposals are radically incomplete, since the operative concepts of the design are missing.

The situation is even a little worse, for unlike conventional machine language organizations, microprogrammed processors are usually oriented to highly special technology, have multiple automatic units that have to be operated in parallel, can even perform in a non-deterministic manner, are location sensitive, and provide a combinatorially larger instruction set. Effective compilers and performance-monitoring software will be mandatory before users can effectively gain any order-of-magnitude increase in performance latent in the basic organization. Furthermore, since these processors are so technology oriented, it is difficult to guarantee that they will have successors or be members of compatible families.

CONCLUSION

We have touched on a number of aspects of current research in computer structures that appear to have possibilities for being important structures in the next half decade. Our examples—and our style of discussing them—suggest several basic points about the design of computer structures. Some of these have been stated already in earlier sections, but it seems useful to list them all together:

(1) Computer design is still driven by the changes in technology, especially in the varying trade-offs.

(2) Distinct regions in the space of trade-offs lead to qualitatively different designs.
(3) These designs have to be discovered by us (the computer designers), and this can happen only after the trade-off characteristics of a new region become reasonably well understood.
(4) Thus, our designs always lag the technology seriously. Those that are reaching for the new technology are extremely crude. Those that are iterations on existing designs, hence more polished, fail to be responsive to the newly emerging trade-offs.
(5) Since the development cycle on new total systems is still of the order of years, the only structures that can be predicted with even minimal confidence are those already available in nascent form. The multiprocessor, cache and RT module organizations discussed earlier are all examples of this.
(6) The design tools that we have for discussing (and discovering) appropriate designs are weak, especially in the domain over which the structures under consideration here have ranged—essentially the PMS level.
(7) In particular, there is no really useful language for expressing the trade-offs in a rough and qualitative way, yet precisely enough so that the design consequences can be analyzed.
(8) In particular (as well), design depends ultimately on having conceptual components of the right size relative to the system to be constructed: small enough to permit variety, large enough to permit discovery. The transient character of the underlying space (the available space of computer structures) reinforces the latter requirement. The notion of M.cache is an example of a new design component with associated functions, not available until a few years ago. Even this small note shows it to be a useful component in terms of which designs can be sought. The potential conceptual revolution hiding in the RT modules provides another example.

REFERENCES

M BARBACCI H GOLDBERG M KNUDSEN
A LISP processor for C.ai
Department of Computer Science Carnegie Mellon University 1971

C G BELL R CADY H MCFARLAND B DELAGI J O'LAUGHLIN R NOONAN W WULF
A new architecture for mini-computers—The DEC PDP-11
AFIPS Conference Proceedings Vol 36 Spring Joint Computer Conference 1970

C G BELL D CASSASENT R HAMEL
The use of the cache memory in the PDP-8/F minicomputer
AFIPS Proceedings of the Spring Joint Computer Conference 1971

C G BELL P FREEMAN et al
A computing environment for AI research
Department of Computer Science Carnegie-Mellon University 1971

C G BELL J GRACON
The register transfer module design concept
Computer Design pp 87–94 May 1971

C G BELL A NEWELL
Computer structures
McGraw-Hill 1971

C G BELL D R REDDY C PIERSON B ROSEN
A high performance programmed remote display terminal
Computer Science Department Carnegie-Mellon University 1971 (For IEEE Computer Conference 1971)

Y CHU
Introduction to computer organization
Prentice-Hall 1970

W A CLARK
Macromodular computer systems
AFIPS Proceedings Spring Joint Computer Conference pp 335-336 1967 (This paper introduced a set of six papers by Clark and his colleagues pp 337–401)

C J CONTI
Concepts for buffer storage
IEEE Computer Group News March 1969

J A DARRINGER
The description, simulation and automatic implemenatation of digital computer processors
PhD dissertation Carnegie-Mellon University 1969

S S HUSSON
Microprogramming: Principles and practice
Prentice-Hall 1970

D MCCRACKEN G ROBERTSON
An L processor for C.ai*
Department of Computer Science Carnegie-Mellon University 1971

A NEWELL D MCCRACKEN G ROBERTSON L DEBENDETTI
L(F) manual*
Department of Computer Science Carnegie-Mellon University 1971

H A SIMON
The sciences of the artificial
MIT PRESS 1969

W Strecker
Analysis of instruction execution rates in multiprocessor computer system
PhD dissertation Carnegie-Mellon University 1970

A TUCKER M J FLYNN
Dynamic microprogramming: Processor organization and programming
Communications of the ACM 14 pp 240–250 April 1971

M V WILKES
Slave memones and dynamic storage allocation
IEEE Transactions on Computers Vol EC-14 No 2 pp 270-271 1965

M V WILKES
The growth of interest in microprogramming: A literature survey
Computing Reviews Vol 1 No 3 pp 139–145