Efficient partitioning for the batch-fabricated fourth generation computer

by N. CSERHALMI, O. LOWENSCHUSS and B. SCHEFF

Raytheon Company
Bedford, Massachusetts

INTRODUCTION

The computer industry is on the verge of an upheaval, due to drastically new hardware and modern computational concepts. Read-only memories which operate near 0.1 microseconds are available, while large-scale integration (LSI) offers the promise of inexpensive, batch fabricated processing of logic and storage elements. The problem which confronts the computer designer is how to use these elements in an efficient manner to take full advantage of their speed and flexibility. Many approaches have been proposed, but none have shown a clear solution to the problem. The concept presented in this paper is the result of extensive development activities.

The partitioning problem

A study of both general and special purpose digital hardware indicates that certain basic arithmetic and Boolean functions are repeated in various combinations throughout a given computer system. Presently available complex logic arrays such as adders, shift registers, and counters perform many of these tasks, and reduce logic cost, compared to the use of simple gates. At the same time they create a serious logistics problem to the hardware product engineer, because stock and maintenance parts increase. In a rapidly growing technology, where labor is so costly, the design, procurement, fabrication, and test cycles are all lengthened by the needs for many different types of parts. By combining several related operations into a single array, significant improvements in logistics, interconnect ratios, power-speed merit figures, and component logic efficiency are realized.

An array can be fabricated with discretionary wiring methods (slice technology) or with the "cell approach" (chip technology), or with hybrid technology. From the component point of view, bipolar or MOS devices may be used.

The size and the complexity of the array is determined by the logic partitioning. The proper selection of the components and interconnection method is dictated by packaging, power consumption, performance, and economic considerations.

How can one tell whether the partitioning is right? Two numeric criteria provide a measure for any given partitioning design;

a) *Maximum Gate/Pin Ratio*—This ratio is generally between 0.1 to 0.6 in present integrated circuit (IC) systems. A ratio near 1 is considered outstanding. With the LSI building blocks described below, a ratio of 3 to 10 can be achieved. The requirement for increased gate/pin ratio is a basic reason for the existence of LSI arrays, because interconnections on the microelectronics level are less expensive and more reliable than at the package level.

b) *Minimum Number of Array Types (Part-numbers)*—This is the best criterion of the logic partitioning, and demonstrates the level of coordination between semiconductor, logic and system designers. Most IC systems employ from 50 to 100 types of printed-circuit modules. Minimum "part-numbers" is one of the primary requirements in all military systems, and is an economics requirement for commercial and low-volume production sys-

tems. Several digital IC systems (including general purpose computers) have been built that contain 3 to 15 types of printed-circuit plug-in boards at 40 or more gates per board.

The penalty paid for maximum gate-pin ratio and minimum part numbers is an inefficiency in the use of silicon (gates) and hence of power. Thus, in "tight" applications such as space vehicles, the designer might prefer to forego the economic advantages of this type of partitioning in favor of minimum power.

Directions in LSI computer architecture

Current developments

With LSI technology progressing toward maturity, more useful documents are being published in the field of LSI system architecture. Most assume the unlimited success of the semiconductor process, but only a few cite documented hardware development. LSI development projects, either completed or near completion, differ in their partitioning concepts; all are designed to replace third generation computers in whole or in part. Some of these development projects are briefly described to supply a background to this paper.

1) The sole producer of discretionary wiring (or slice technology), Texas Instruments, has designed and built an LSI airborne computer, the Model 2502. This general-purpose 16-bit word computer consists of a Central Processing Unit (CPU) and a 3-channel Input/Output Unit. The single address CPU is controlled by a 2 MHz clock, and the 36-instruction set is executed through four general purpose registers. (Reference 1)

   From the partitioning point of view, the most significant feature is that the CPU is implemented with 16 identical arrays, which include all arithmetic and register functions except for the control logic. The elimination of "special-purpose" registers (index, accumulator, etc.) must also be emphasized. While the CPU is partitioned on a bit basis, the three I/O channels are implemented by six arrays of another type, partitioned on a functional basis. A gate-to-pin ratio of two was achieved on both arrays.

   Although automated design was used to assist in the implementation of the control logic, the use of ten different arrays with low gate-to-pin ratios was required. Another disadvantage is that the industry-wide claims of LSI speed improvements do not appear; the computer is three to five times slower than is attainable using currently mass-produced TTL circuits.

2) A larger and more powerful general-purpose aerospace computer has been designed (hardware near completion) at Raytheon Company. This parallel 32-bit word computer offers 94 instructions, indefinite chaining of Index and Indirect Addressing, and Multiprocessor capability. The seven full-length registers of the CPU are partitioned into eight identical arrays. Each array contains four stages of all seven registers, including the transfer gate structure. These eight LSI Arrays of one type contain 56 percent of the entire CPU logic.

   The control logic is temporarily implemented with 14-lead TTL circuits. Some further work is being done to replace the majority of control with a Read-Only Memory (ROM), leaving only less than five percent non-LSI hardware.

3) In another Raytheon development program, a CPU is under construction which utilizes 25 "Raytheon AS-80" arrays. This multi-purpose 4-bit Counter/Register is used as a building block to form the eleven working registers which are 4, 8, or 16 bits long. The block diagram of this ar-

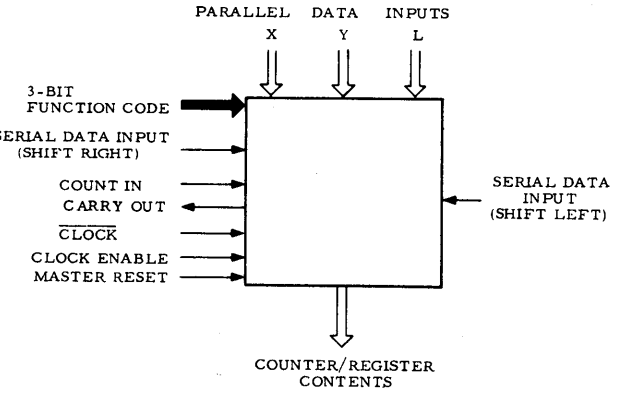


FIGURE 1—Multi-purpose counter/register

ray is shown in Figure 1; it will be described in detail in a later part of this paper.

The processor is a 16-bit, parallel, single-address, high-speed unit. Operating with a solid-state, read/write memory, it can execute an add instruction in less than ½ μsec, and a 16-bit multiply in less than 2 μsec average.

From a partitioning point of view, the CPU is organized as follows: 80 percent of the logic is implemented by two types of arrays (Raytheon AS-80 and Signetics 8260) and the remaining 20 percent is implemented by high-speed TTL circuits. The majority of the control logic will be eventually replaced with ROM.

4) A classical example of functional partitioning is the LIMAC (Large Integrated Monolithic Array Computer) concept being implemented by RCA. The hardware of this 16-bit machine is grouped into functional execution units, each of which contain the control and the operand registers. A simple centralized control directs the data flow, governed by micro-instructions. (Reference 2)

   This concept favors large arrays, but it is not restricted to any specific size. As semiconductor technology improves, more circuits can be included in a package which leads to increased gate-to-pin ratio and decreased unique parts in this particular system organization; e.g., with 100 gate blocks, a gate-to-pin ratio of 2.5 can be achieved; with 1000 gate blocks, a gate-to-pin ratio of better than seven can be realized.

Efficient partitioning with building blocks

In the following paragraphs two building block approaches will be presented. The first utilizes an 80 gate array with 3 to 1 gate-to-pin ratio, the second concept is based on a more complex array with gate-to-pin ratio of ten, and Read-Only Memories.

1) *AS–80 Processor*—A basic register array with built-in microprogrammable functions can be fabricated with 4, 8, 16 or even 32-bit elements. A 4-bit array was designed and fabricated to prove the feasibility of this approach. The block diagram of this Raytheon AS–80 (Patent pending) is shown in Figure 1. Simultaneously, a processing unit was designed around 25 of the AS-80 arrays to demonstrate its usefulness.

The Raytheon AS–80 is an example of a high-speed, programmable, 4-bit register contained in a single 28-lead flat-pack. As shown in Figure 2, eighty NAND gates are interconnected to provide a programmable counter/register, capable of operating in any one of the following eight mutually exclusive modes by entering a 3-bit function code:

- Clear—Also enter data into selected bit positions
- Shift-left—
- Shift-right—
- Load-Enter 4 bit parallel data
- Hold
- Complement
- Count-down—straight-binary counter, decrementing
- Count-up—straight-binary counter, incrementing

The Raytheon Register comprises four flip-flops, each consisting of six NAND gates. Additional logic circuits are included to enable storage elements to be programmed and used as either a storage register, binary counter (up or down), shift register (left or right) and to be complemented (change state).

TABLE 1

	AS–80 Array	Equivalent Logic Built of Standard TTL IC's
Number of Packs	1	28
PC Board Connections	28	292
Wire Bonds	56	584
Clock Interval and Power	60 ns 0.75W	125 ns 1.4W
Speed. Power Product	45	175

Several Boolean operations are performed without the addition of external logic, by combining coded commands with the "L" data input. For example, the execution of both hold and load (L)

yields the logic "OR" function (A+L).

Table 1 compares the AS–80 array with equivalent logic, constructed by means of present-day standard TTL integrated circuits.

Interface with the AS–80 is at standard Transistor-Transistor-Logic (TTL) levels. Input and output characteristics are identical to those published for high level TTL circuits. Gate outputs designed to be used internally on the chip can be connected together to form wired OR functions. The AS–80 array is currently fabricated by Sylvania on a single monolithic "chip" with three levels of metallization for the internal connections. Figure 3 is the microphotograph of an area of a silicon slice where an AS–80 is formed. On the periphery the bonding pads are also visible.

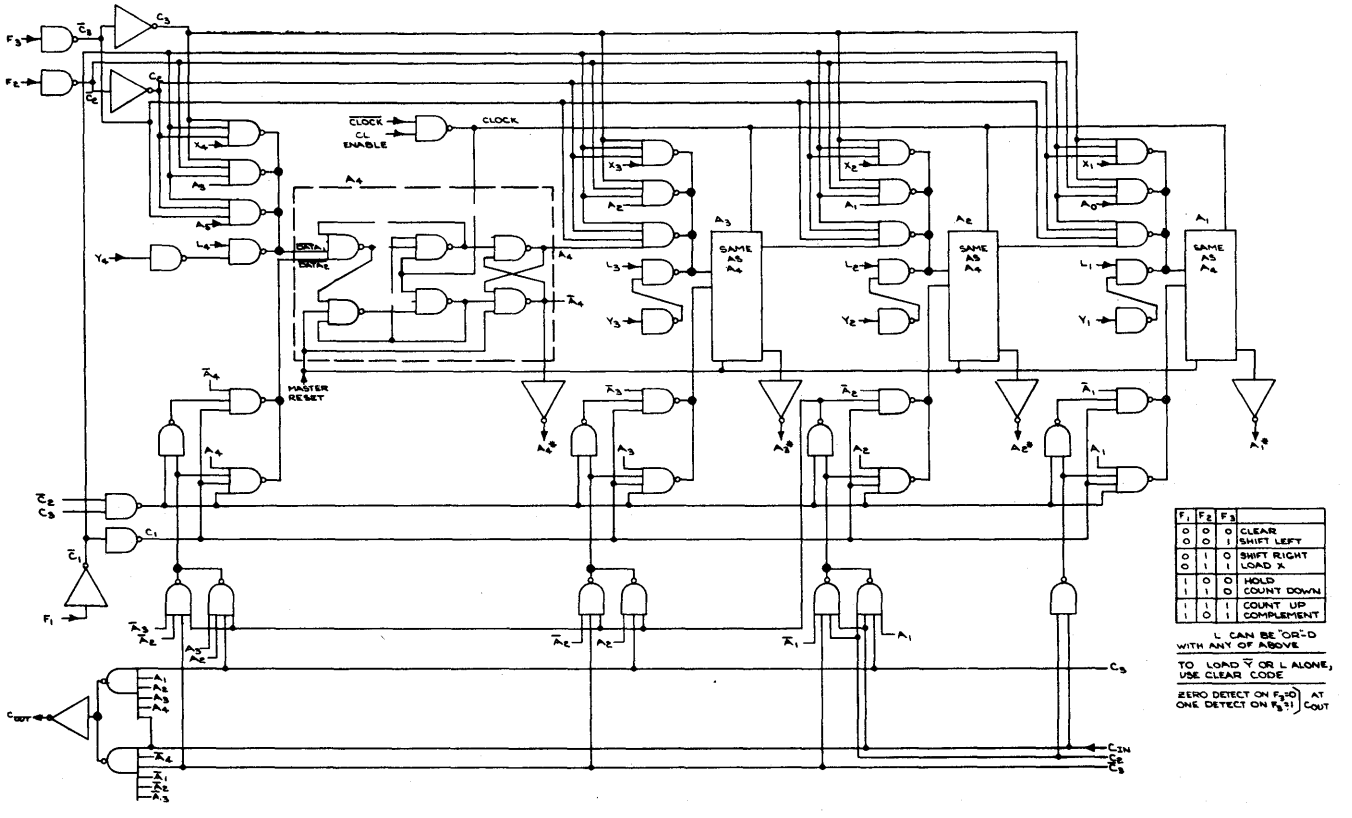
Arithmetic operations require the use of another logic element. A highly complex four-bit array was designed which performs most arithmetic and logic combinations of two operands, and can be implemented with 70 gates and 24 pins. However, it functionally overlaps the AS–80 register array, and its usage is limited; therefore, its building block usefulness is questionable.

Examination of the AS–80 array shows that many of the functions of a general arithmetic unit are included in the array (shift, complements, count, etc.). All that is necessary in the arithmetic unit are the pure add (and therefore subtract) operations associated with a central processor. Selection gating provided at one of the operand inputs of the register array enables convenient implementation of subtraction. In addition to the register and adder arrays, control logic is required to provide synchronization, gating signals, and a sequence of logic instructions in order to be operated as a homogeneous system.

2) *Register Array Concept for the Fourth Generation (RACON–4)*

Every system previously described had to carry the burden of third generation computer operation, such as specialized hardware, high application programming cost, and excessive processing time for internal scheduling (Reference 3).

Users of third generation systems have found that in the case of leased equipment, the cost of hardware is only about five



F_1	F_2	F_3	
0	0	0	CLEAR
0	0	1	SHIFT LEFT
0	1	0	SHIFT RIGHT
0	1	1	LOAD Y
1	0	0	HOLD
1	1	0	COUNT DOWN
1	1	1	COUNT UP
1	0	1	COMPLEMENT

FIGURE 2—Logic schematic of AS–80

percent of the total operating cost. The system electronic circuitry represents roughly one-third of that five percent. Of the remaining 95 percent of operating cost, approximately 35 percent is absorbed by software; 30 percent is required by special environmental facilities, power, and operating costs; and the balance (30 percent) is made up of manufacturer's costs, sales fees, maintenance, and profit. This means that even if LSI technology makes the entire electronics available at no cost it does not represent any significant economies for the user.

In less than one year, the industry will have mastered a new generation of hardware. However, if this new generation of hardware is used only as a replacement for the hardware of a third generation system, it will have a doubtful future. Only systems that can exhibit reduction of the other 95 percent of cost can be considered advanced systems and candidates for the fourth generation family.

Thinking further ahead, such advanced systems will lead to a new market situation where the cost of the hardware becomes significant and a major factor in a Buy/No-Buy decision.

If we assume that a small number of general-purpose building blocks can be developed to perform all the computer functions, the problems of

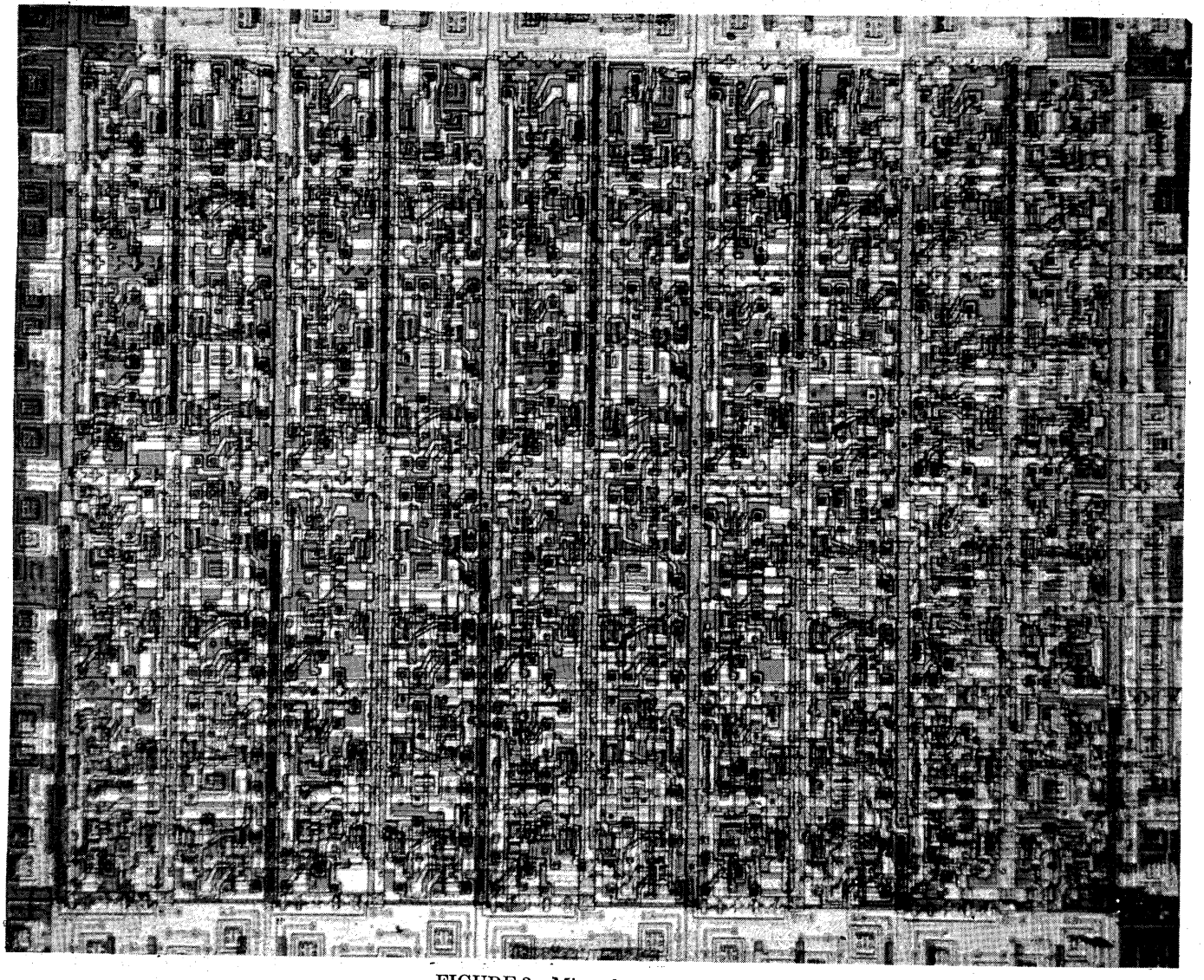


FIGURE 3—Microphotograph of AS-80

specialized hardware, of general-purpose capabilities where only specialized functions are required, of excessive programming costs, and of prohibitive processing times would be solved for both user and manufacturer. The producer of computing machines would build up an inventory of these basic blocks and assemble them to the size and configuration of the user's requirements with a minimum of system design effort. Computing systems will then be defined by the total number of bits of all working registers, rather than by the number and the size of registers. Fixed hardware implemented blocks will "personalize" the system. The user, on the other hand, will not be forced to buy capacity and features he does not need and the cost would become linearly proportional to the size and computing power required.

With this concept in mind, the current and future computing systems can be characterized as:

- Third generation: Specialized hardware, General-purpose systems
- Fourth generation: Specialized systems, General-purpose hardware

RACON–4 is a feasibility model of a fourth generation system utilizing fourth generation hardware. The system block diagram (Figure 4) can be best explained by starting from the working register area. The workhorse of the system is a number of RA–4 register arrays. The actual number of RA–4 arrays is determined by the need and the budget of the user. Each RA–4 or groups of RA–4's can be programmed by a read-only memory to perform any of sixteen functions. The output of this memory passes through the Register-Function Matrix (RFM) which couples a number of registers together to perform identical functions and thus determines the word length. The contents of the registers are routed through the Register Data Bus (RDB) which is a single, dual, or triple-bus system, depending on the performance reguirements. Both the bus system and the scratch-pad memory are controlled by the output of the micro-command memory. This memory consists of Read-only Command Memory (RCM–16) arrays, each containing 256 bits of information in a 16 x 16 arrangement, and the Read-only Memory Logic (RML) which is basically an addressing system.

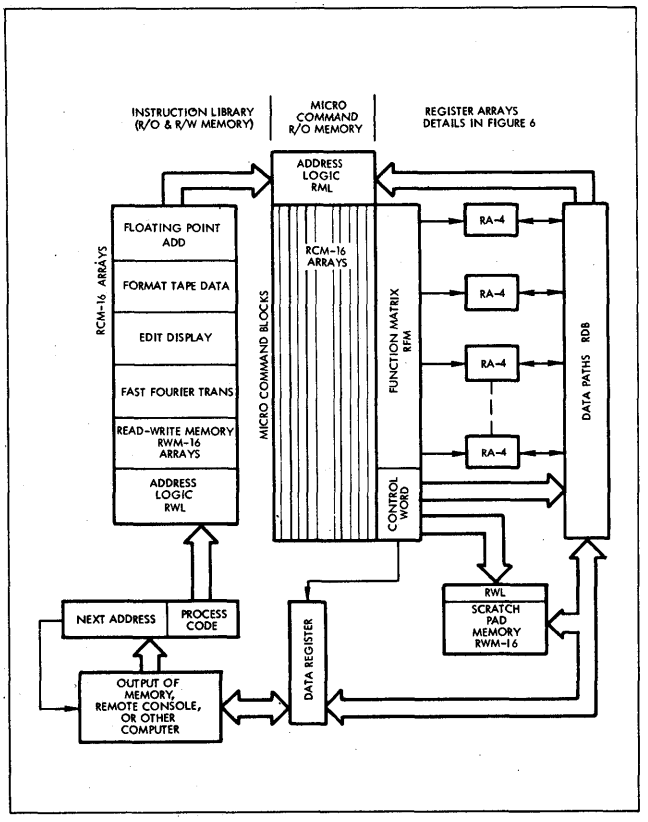


FIGURE 4—System block diagram, RACON–4

A sequence (or block) of micro-commands is selected by the Instruction Library memory. This memory is composed of the same hardware as the micro-command memory. An additionad read-write section is available as an option for the user, and utilizes the same RWM–16 hardware as the scratch-pad memory. The input/output Data Register is an optional feature since the scratch-pad memory could serve the same purpose, with performance restrictions.

For better understanding of the features of RACON–4, consider a processing unit consisting of sixteen four-bit variable arrays and a Read-Only Memory Module as shown in Figure 5. The memory contains the word address logic, a number of parallel words, and an interconnection matrix with 64 plug-in terminals. The four control inputs of each of the sixteen register arrays are permanently connected to the other half of the 64 terminals. With this concept, each word of the memory can program every 4-bit register to perform any of the sixteen functions. By inserting a point-to-point interconnection pattern between terminals and bit lines of the memory, the processor can be operated as a 4-bit machine with sixteen registers. It requires words 64 bits long from the memory for every step of operation. The insertion of the pattern shown in Figure 5(a) will

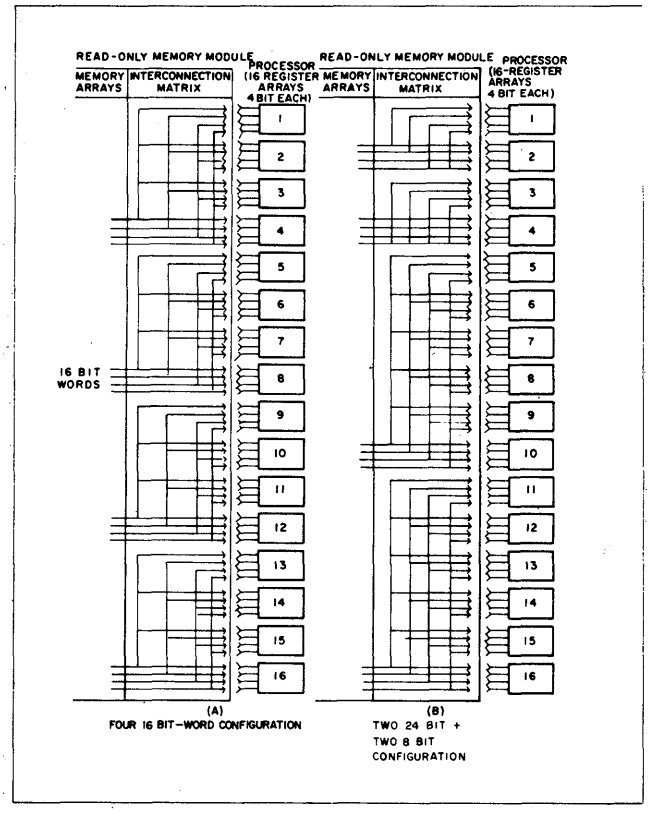


FIGURE 5—Register arrays for a 64-bit processor

result in a 16-bit machine with four registers requiring only a 16-bit word from the memory for each cycle. Another interconnection matrix could result in a 32-bit machine with two registers programmed by 6-bit words from the memory. Figure 5(b) shows a further possible configuration.

The RA–4 is a 4-bit register array similar to AS–80, but instead of the increment/decrement logic, a 4-bit full adder is incorporated. Programmable functions are increased to sixteen. Because

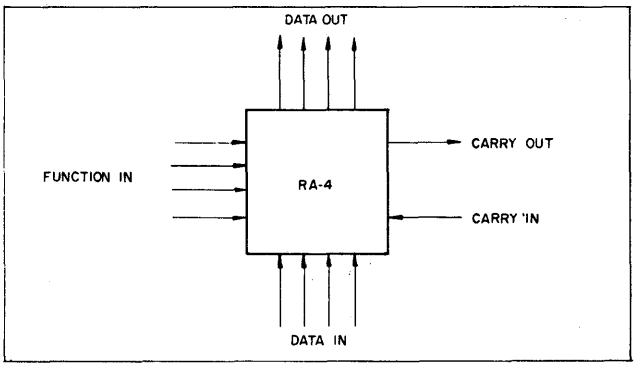


FIGURE 6—Block diagram of RA–4

of the bus oriented organization of RACON–4, the multiple entry of AS–80 was eliminated, as shown in Figure 6.

The array is capable of performing any one of the following functions in eighty nanoseconds:

1) Load
2) Load complement
3) Add
4) Subtract
5) Reset
6) Exclusive OR
7) Logic AND
8) Hold
9) Increment and all one detect
10) Decrement and all zero detect
11) Shift left by 1 place
12) Shift right by 1 place
13) Shift right by 2 places
14) Add and shift right by 1 place
15) Add and shift right by 2 places
16) Subtract and shift right by 1 place

Over 160 gates are needed for the implementation of RA–4, resulting in a unit with a gate-to-pin ratio of ten.

The flow of data is entirely under the control of the micro-command memory, with the aid of RDB and the scratch-pad Memory. The process portion of the instruction to be executed is used as a starting address in the Instruction Library. The contents of this memory are one or more words which form the starting addresses of the micro-command memory. The micro-command memory contains the group of 4-bit numbers that program the register arrays to perform the appropriate micro-commands.

Linking of 4-bit groups together can similarly be accomplished by "programming" to produce any multiple of 4-bit word sizes. Furthermore, different computer words sizes may be associated with different processes programmed in the second memory. This would facilitate, for example, the effective utilization of a computer with an external device.

Because there is no restriction on the special-purpose instructions or the internal formats and word size which could be created, the basic hardware building block approach leads to a system organization which provides virtually unlimited flexibility to the user. In addition, the potential software advantages are considerable; for example, existing software can be utilized without radically modifying existing pragrams. New instructions can be easily added to the existing instruction set and higher meta-level instructions can be created. Consequently, both programming and machine design becomes easier and more

adaptable to any application without any loss in generality.

The entire control system, consisting of all control algorithms and computer instructions, can be represented by programmed blocks of fundamental machine operations placed in the micro-command memory. This approach differs from the micro-programmed computers in which specific computer functions are programmed, or from a standard computer in which specific functions are hardwired. Instead of designating special registers (accumulators, counters, index registers, etc.) and parameters (transfers, invert, etc.) in each of these blocks of machine operation, a variable representation is used. Consequently, the blocks are general, so that each block can, with different register and parameter designations, perform the functions of a large number of specific instructions.

Creating the entire set of computer instructions with these general micro-command blocks may be contrasted with present-day digital computer control memory techniques. In today's computers, the instruction complement consists of a collection of special-purpose instructions which differ slightly in terms of processes; there are many different instructions, but only a few unique operations. Using the general micro-command blocks as a basis, any desired set of instructions can be defined.

In fact, different sets of blocks can be specified to effect different system characteristics, depending upon the application. A representative set of blocks forming a complete (and powerful) instruction complement is listed below, together with the specific instructions accomplished within the block:

1) Transfer: This block implements the operations of transfer, enter, execute, single-bit tests, etc.
2) Count: This block implements the count (increment or decrement).
3) Shift: This block implements all forms of single precision/double precision left/right, open/closed, algebraic/logical shifts.
4) Exchange: This block implements swaps between any two registers.
5) Logical: This block implements the logical operations such as "and", "or".
6) Sign and Magnitude Add-Subtracts: This block implements all of the variations of sign and magnitude addition and subtraction.
7) Mask test: This block implements the on/off tests of fields, tests of any collection of bits, and indicator resets.
8) Three-way comparison: This block implements all three-way comparisons as well as two-way logical comparison tests.
9) Multiply Divide Iteration step: This block implements the basic process for fixed/floating multiplication/division of fixed or variable length.

Because any register can be used by the micro-command block, arbitrary names for different registers are unnecessary. Because a register, such as an accumulator, has a definition in terms of its use, it may be convenient to give a name to a register in which a specific function or process is generally performed.

As a direct consequence of the fact that the general forms define basic processes, they are independent of normal computer word characteristics, such as word size and internal field definitions. Registers, in application, may have different word sizes. Consider, for example, the general form for a shift operation. A general shift operation is typically a set of the following operations:

1) Shift A n places and store in A.
2) Shift B n places and store in B.
3) Repeat steps 1 and 2 m times.
4) Shift sign of A to sign of B.

In this routine, A and B represent registers, the parameter n represents the hardwired shift algorithm, and the parameter m represents the number of times the shift algorithm is used. Obviously Steps 1 and 2, used together, represent a double precision shift; Step 1 alone is a single precision shift.

It should be observed that A and B could be different length registers.

As a result of being able to program the basic control sequences in terms of primitive forms, only a small percentage of control logic remains to be implemented. Part of this logic, primarily the creation of more complex instructions, and the control logic required to establish internal formats, can also be handled by programming techniques.

Specialized processes consist of collections of the basic forms. The program to implement these

processes utilizes the "language" of the basic general forms. Consider the floating point operations; these are programmed by combining the basic forms. A "floating add" is a collection of "transfer," "logical," "exchange," "shift," and "sign and magnitude" subroutines. "Floating subtract" is the same as "floating add" after the first subroutine, a sign change. This is analogous to the manner in which the control of complex computer instructions is presently achieved in computer design (built-in subroutines with specialized suborders). For example, the standard multiply instruction consists of 1) determining the sign of the answer, 2) the multiplication process of add, shift, and iterate, and 3) the final adjustment of the answer.

The programming of these specialized processes is performed in a second read-only memory called the instruction library. The ability to program this memory, using the general building block forms in the micro-command memory as a source language, allows functions normally ascribed to the software to be built into the hardware by being programmed in the instruction library. Thus specialized instructions can be created and specialized routines for which high processing speed is required, such as input/output mechanizations (reading and writing data, editing procedures, input/output buffer control) or a Fast Fourier Transform algorithm, can be implemented in additional to normal computer instructions.

This approach permits the computer to be tailored for many specific applications within the mass-produced standard computer design without the need for specialized software. Much of the complexity of a higher order language is bypassed.

CONCLUSION

Recent developments in batch fabrication of logic and memories have made it necessary to reconsider the interrelations between design effort, hardware cost, and partitioning. It is possible to achieve a low "part-number" count by making use of multipurpose register array packages that include control logic inputs. These control inputs are supplied from read-only memories. The control inputs cause the data in the package to undergo microprogrammed operations to form the instructions of the computer.

REFERENCES

1 R C JENNINGS
Design and fabrication of a general-purpose airborne computer using LSI arrays
Digest of IEEE Computer Group Conference June 1968

2 H R BEELITZ et al
System architecture for large-scale integration
Proceedings of Fall Joint Computer Conference 1967

3 C J WALTER
Impact of fourth generation software on hardware design
IEEE Computer Group News July 1968