J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*

BIT-SLICE COMPUTERS

Bit-Slice Computers

In early days of microprocessors, when semiconductor technology was not capable of integrating 16-bit or wider processors on a single chip, integrated circuit (*IC*) designers developed a set of 4-bit or 8-bit microprocessor parts that would allow system designers to develop their own wide data path microprocessors (1,2,3). These chips, called *bit-slice processors*, had a modular construction that facilitated expansion to units of longer word lengths. Designers and system developers built several interesting computers and controllers by cascading several of these basic building blocks (1,3). This bit-sliced design was a very popular method of designing processors during the late 1970s and early 1980s. Although the higher levels of system integration supported by modern semiconductor technology has diminished the popularity of the bit-slice design approach, principles of bit-slice computing can be observed in native signal processing instruction set extensions incorporated in microprocessors such as the Intel Pentium (4).

The Bit-Slice Design Approach. Bit-slice processor design is the technique of constructing an N bit microprocessor using m copies of a k bit cascadable basic building block, where $N = km$. Generally, k is fairly small (e.g., 4-bits). This cascaded arithmetic and logic unit (ALU), which is an array of several smaller ALUs, is called bit-sliced because it is comprised of several smaller chips each of which processes a slice of the km bit operand.

Each bit-slice is a building block that performs a variety of functions on a small number of bits under the control of a set of input control signals. Figure 1 illustrates a simplified view of a bit-slice ALU. Four 4-bit ALU bit slices are interconnected in an array fashion to obtain a 16-bit ALU. The data buses are concatenated to form a wider bus. The control signals are tied together so that all units receive the same control signals.

Two popular manufacturers of bit-slice chips were Advanced Micro Devices (AMD) (1,3) and Texas Instruments (*TI*) (2). The commercial bit-slice chips that came from these manufacturers were designed to support all basic operations in earlier microprocessors.

Architecture of a Bit-Slice Building Block. Manufacturers of bit-slices built their chips around the organization of a typical computer central processing unit (*CPU*). Despite differences in data width and internal circuitry, the general architecture of most processors is similar. Figure 2 illustrates an abstract view of the typical processor architecture. The ALU is the heart of the *CPU*. The control unit provides necessary timing and control signals to the data path enclosed within the dotted lines in Fig. 2. A collection of registers, usually called a *register file*, provides inputs to an ALU. The register file has two read ports used to read two operands at a time and one write port used to write the result. The manufacturers abstracted a processor slice from this architecture and provided ICs that capture 4- or 8-bits of this data path. By cascading these bit-slices in an array, it then becomes possible to construct a powerful computer.

The bit-slice ALU chips from Advanced Micro Devices and Texas Instruments had an architecture very similar to the abstract view of the data path in Fig. 2. This can be observed in Fig. 3, which illustrates the major elements in AMD's Am2901 bit-slice *CPU*. There are 16 4-bit registers organized as a 16×4-bit RAM.

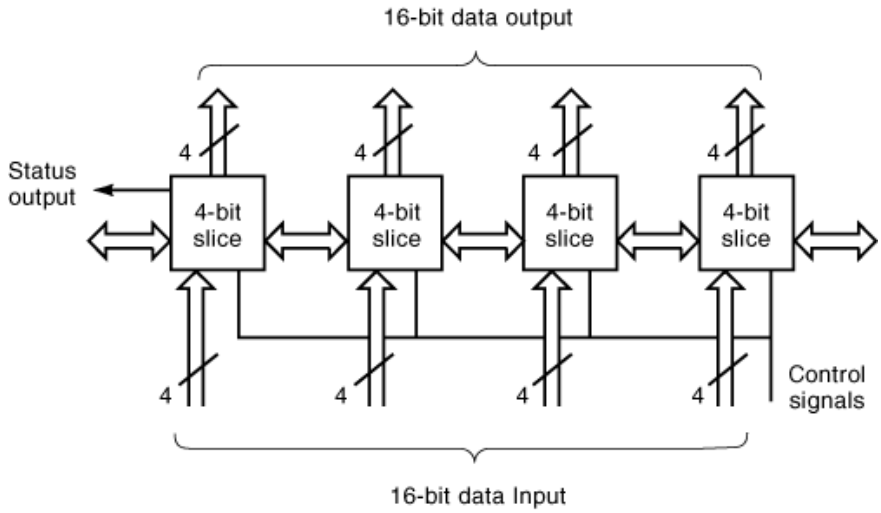

Fig. 1. A simplified view of a bit-sliced ALU. The data buses are concatenated to form a wider bus. The control signals are tied together so that all units receive the same control signals.

An additional register, called the *Q*-register or quotient register is provided to hold the multiplier and quotient while implementing multiplication and division using shift and add/subtract algorithms. The ALU inputs are obtained from the register file, the external input data bus, or the quotient register. It is also possible to choose an input that is all 0's. The register file has two read ports and one write port and multiplexers are provided to guide the appropriate input to the ALU. The ALU can perform arithmetic, logic, and arithmetic right or left shifts, and there are also shifters shown outside the ALU. These shifters allow the ALU results to be shifted to support shift and add/subtract operations as required in algorithms for multiplication and division. The results generated by the ALU can be stored in the registers or sent out on the external data bus. The control word for controlling the bit-slice includes bits to select the input data for the ALU, bits to select the ALU operation such as ADD, SUB AND, OR, XOR, and XNOR and bits to control the destination and the direction and magnitude of shifts if any. A 9-bit control word is split into three 3-bit fields that control the source operand selection for the ALU, the ALU operation, and destination selection. Multiplication or division by two is accomplished by shifting the ALU output left or right. In addition to the result, the ALU produces status flags such as carry out, sign, zero, and overflow.

Construction of a Bit-Slice Arithmetic and Logic Unit. Figure 4 shows how a 16-bit ALU can be constructed from four 4-bit ALU slices. The ALU circuits of the individual slices are cascaded to form 16-bit ALU circuits. The control lines are connected to each other and to the external control signals in such a way that each slice performs the same ALU operation on a different 4-bit slice of the input operand, and produces the corresponding 4-bit part of the result. Some of the operations require transfer of data between the slices. The major interslice connection is feeding the carry out from one slice to the carry-in of the next higher slice for proper arithmetic operations. Logical operations such as AND, OR, XOR, and NOT can be implemented with independent slices. For shift operations, there will be connections from each slice to the adjacent slices. The bit-slices that are conventionally available from vendors such as AMD and TI also provide status bits and the status bits have to be appropriately connected to produce the set of status signals that corresponds to the result of the ALU operation. The 4-bit registers on each bit-slice are concatenated to form 16-bit registers. 5 provides a simplified description of the construction of a bit-slice ALU.

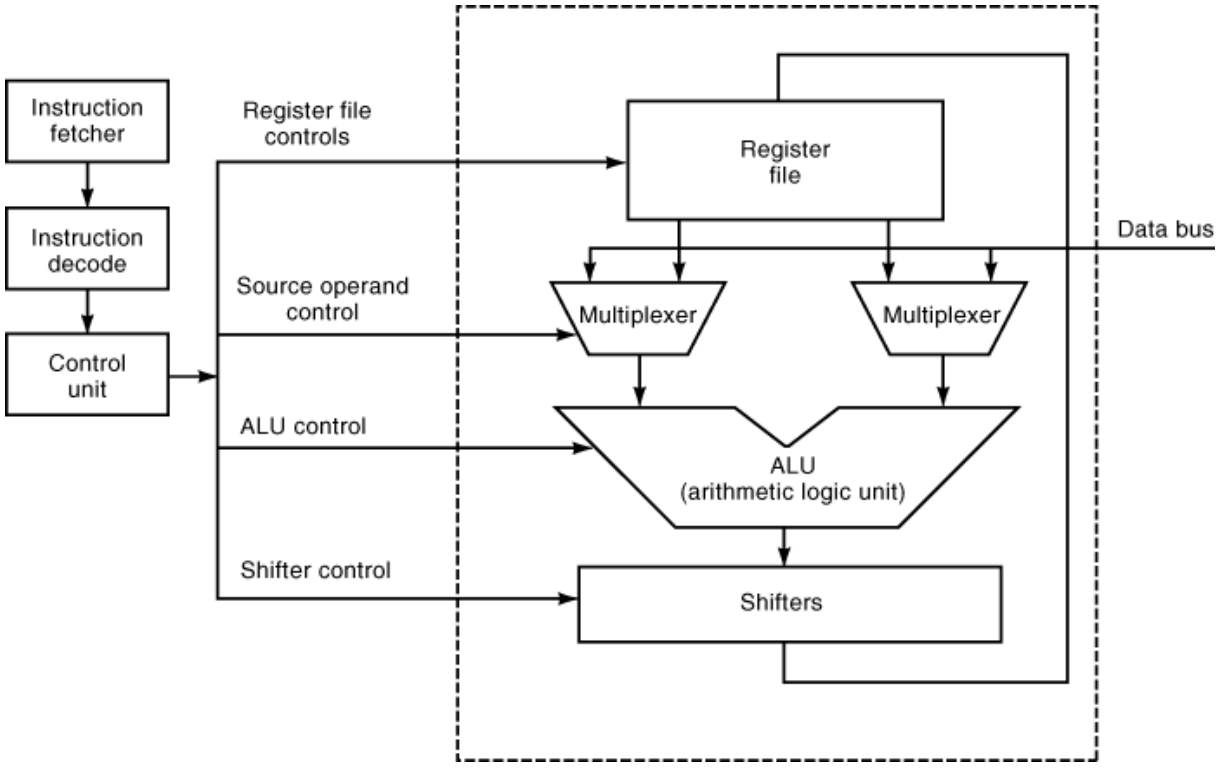

Fig. 2. Abstract view of a *CPU*. The datapath is enclosed in the dotted lines. The control unit provides necessary timing and control signals to the various elements of the datapath.

An obvious problem with cascading adders is the increase in carry propagation delay. If the bit-slice processor designer simply connects the carry out from each slice to the carry in of the next higher slice, the carry has to ripple through all the slices and the adder will be very slow because of the serial propagation of the carry. Instead the designer can choose to make use of advanced adder techniques such as carry look ahead schemes. Carry look ahead adders are fast adders that generate signals called *carry generate and carry propagate and* use them to realize the adder operation without waiting for the serial (ripple) carry from the previous stage. To support such fast addition most commercial ALU bit-slices also generate carry generate and carry propagate signals and the vendors make special carry look ahead block circuits that take these signals from several slices and simultaneously generate the appropriate carry signals for the succeeding bit-slices. The Am2901 produces carry look ahead signals such as and propagate in complemented form. Each slice feeds its generate and propagate signals into the carry look ahead generator chip which employs a fast carry generation circuit.

Figure 5 shows the construction of a fast 16-bit adder using four 4-bit slices and a carry look ahead generator chip. The AMD Am2900 family of ICs includes ALU chips such as Am2901 and Am2903, and look ahead carry generators such as Am2902. The zero flag from each of the slices is fed to an gate to provide the zero flag of the 16-bit ALU. The circuit in Fig. 5 can also be realized using 74LS181 4-bit ALU and the 74LS182 carry look ahead generator manufactured by Texas Instruments and other companies.

Not every ALU operation can be completely partitioned in this manner to smaller bit-slices. Multiplication and division on wide words cannot be achieved by simply cascading narrow multipliers and dividers. But they can still be achieved using bit-slices with shift and add/subtract algorithms. As indicated before, some of the

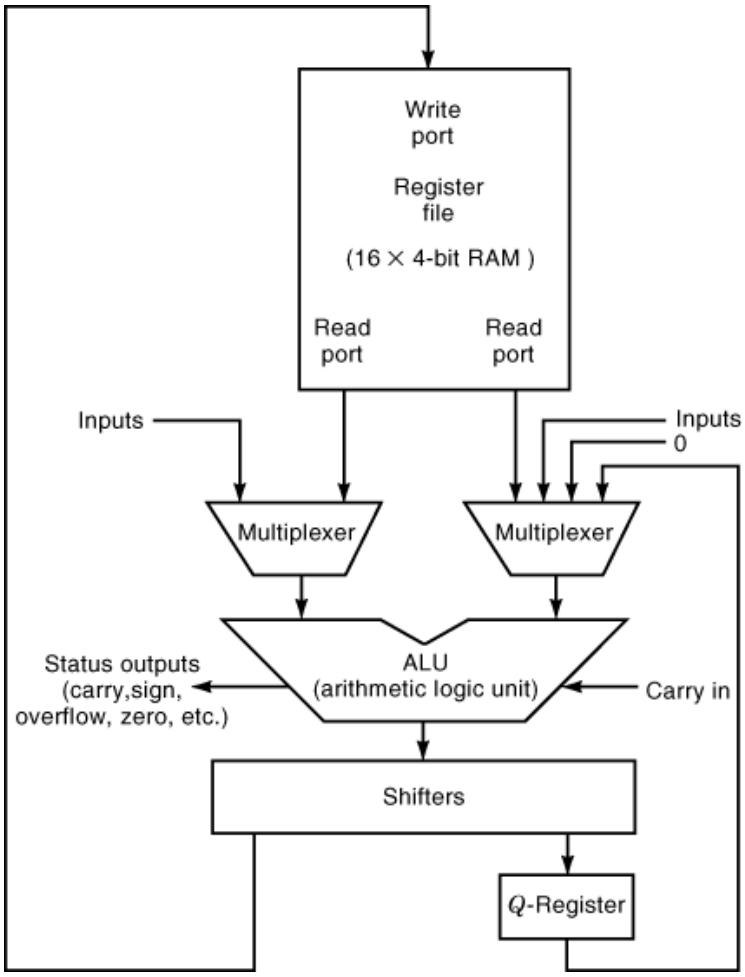

Fig. 3. Simplified architecture of AMD's Am2901 bit-slice ALU. All registers and the ALU are 4-bits wide. The Q-register provides intermediate storage during shift and add/subtract algorithms during multiplication and division. The ALU inputs can be obtained from the register file, the external input data bus, or the Q (quotient) register. It is also possible to choose an input which is all zeros.

bit-slice registers are used to hold intermediate results during shift-and-add operations in a multiplication algorithm.

Construction of a Bit-Slice Computer. Two important parts of a computer *CPU* are the data path and the control path. The data path can be implemented by cascading several bit-slice building blocks. There are two general methods for implementing a control unit: hardwiring and microprogramming. In hardwiring, the control circuitry is implemented using random or unstructured logic, whereas in microprogramming, control words are stored in a memory area and accessed in appropriate sequence. This memory is generally called the *control store* or the *microprogram memory* or simply the *micro-memory* (6). The bit-slice design style became very popular in one period of computer design, and the manufacturers of bit-slice chips provided several auxiliary chips for microprogram sequencing, input–output support, and so on. The sequencer chips enabled the easy development of microprogrammed bit-slice processors. Figure 6 demonstrates how various chips from

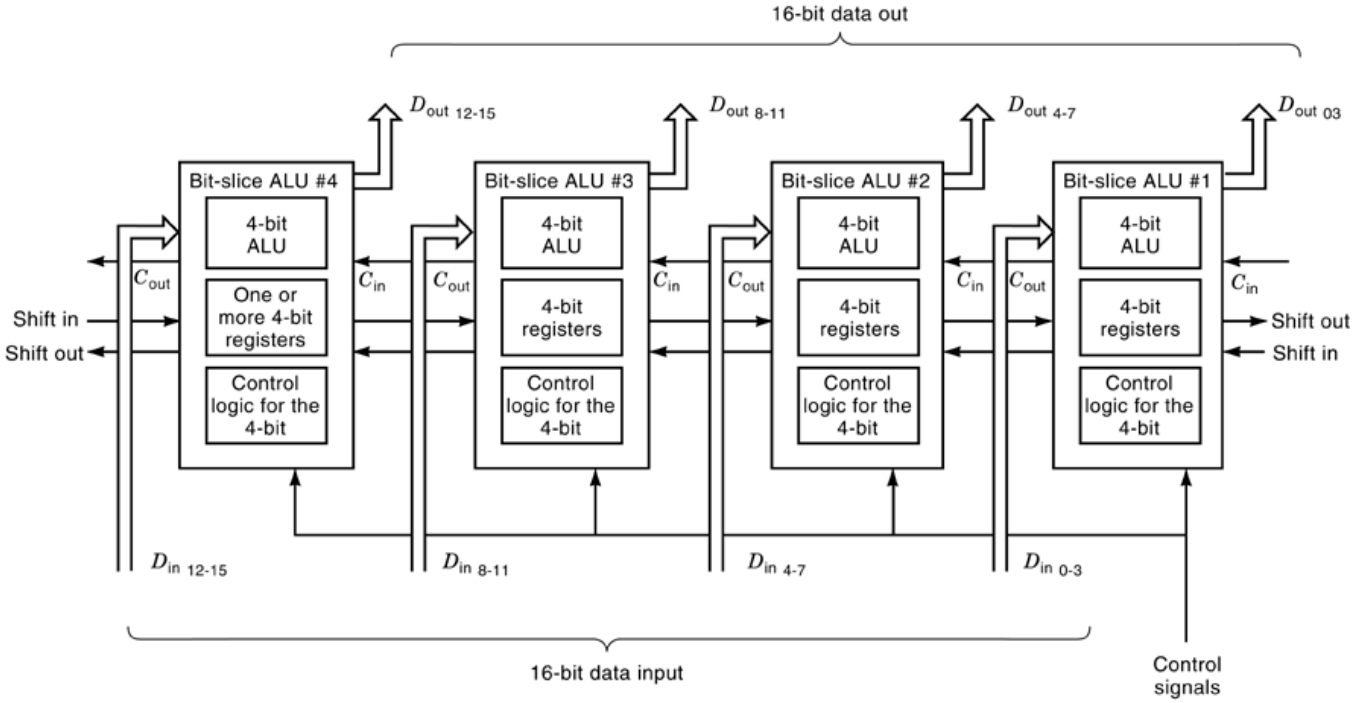

Fig. 4. Construction of a 16-bit ALU from four 4-bit slices.

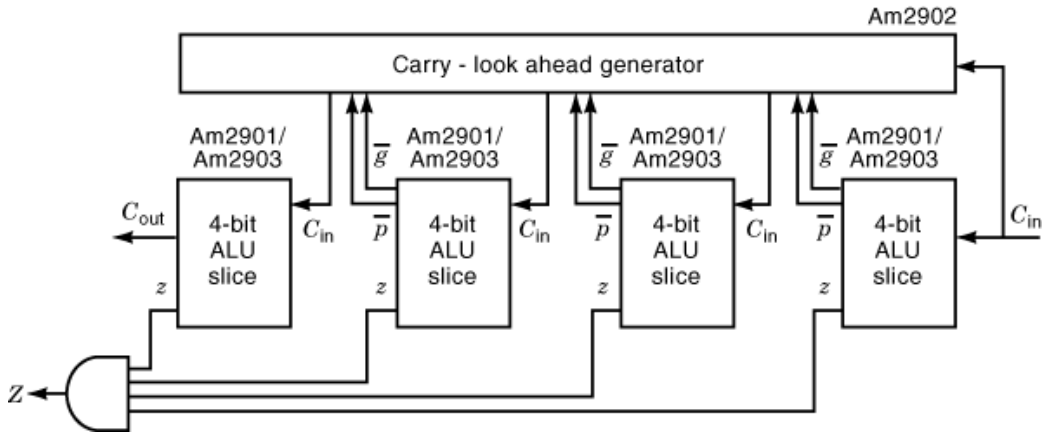

Fig. 5. A fast 16-bit adder constructed using four 4-bit ALUs and a look-ahead carry generator chip. The carry generate (g) and propagate (p) signals from each slice are fed into the carry look-ahead generator chip, which generates simultaneously the appropriate carry signals for all succeeding bit-slices.

AMD's bit-slice family can be used to construct a general processor. The flow of signals in a microprogrammed bit-slice *CPU* is illustrated in Fig. 7. The sequencer generates the next address to the microprogram memory. The microprogram memory contains the appropriate microinstructions, which provide control information to the ALU and the sequencer. The microinstruction bits control the ALU, registers, bus transceivers, sequencer,

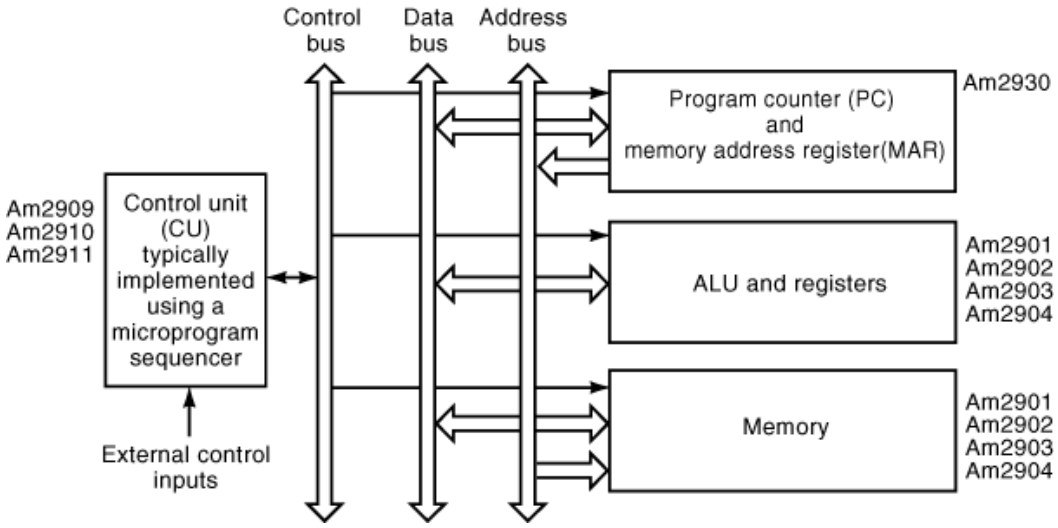

Fig. 6. Construction of a computer from traditional bit-slice components. Chips from the AMD Am2900 series that can be used to build each block are indicated on both sides of the figure.

and other system components. It may be noted that the processor has a different program counter (PC), memory address register (MAR), and instruction register (IR) for the user code. The micro-PC, micro-MAR, and micro-IR are different registers and are part of the control unit. The system memory stores the user code, each instruction of which triggers the appropriate microcode from the control store.

Advanced Micro Devices introduced a large family of bit-slice processor parts and also released a wealth of information on how to construct bit-slice computers using them. Table 1 illustrates a list of the most important bit-slice chips from Advanced Micro Devices and Texas Instruments. The detailed design of a 16-bit processor using four 4-bit slices is illustrated in Chapter 9 of Ref. 1 Texas Instruments also provides detailed information on how to construct 16-bit or wider processors using their SN74AS888 8-bit ALU slice and SN74AS890 sequencer (2).

Application Specific Instruction Set Computers

Although not necessarily a feature of bit-slice design as such, historically the bit-slice building blocks developed by vendors such as AMD and Texas Instruments supported microprogramming and hence building bit-slice processors also allowed the opportunity to create custom instruction sets specifically tailored towards specific applications (7,8). Several system designers made use of the opportunity, using microprogram sequencers specifically designed to work with the bit-slices and constructing application-specific processors of their choice.

Any desired instruction set can be realized by appropriate microprogramming using the basic instruction set of the bit-slice. A complex instruction of the desired instruction set would consist of a sequence of several bit-slice instructions. This sequence, called the microprogram, is written into the *control store* and every time the complex instruction is desired, the sequencer controls the bit-slices to advance through the equivalent microprogram.

Advantages of Bit-Slice Approach. Bit-slice components made possible the design of arbitrarily wide processors that suited different applications. The bit-sliced approach has the advantage that any desired word

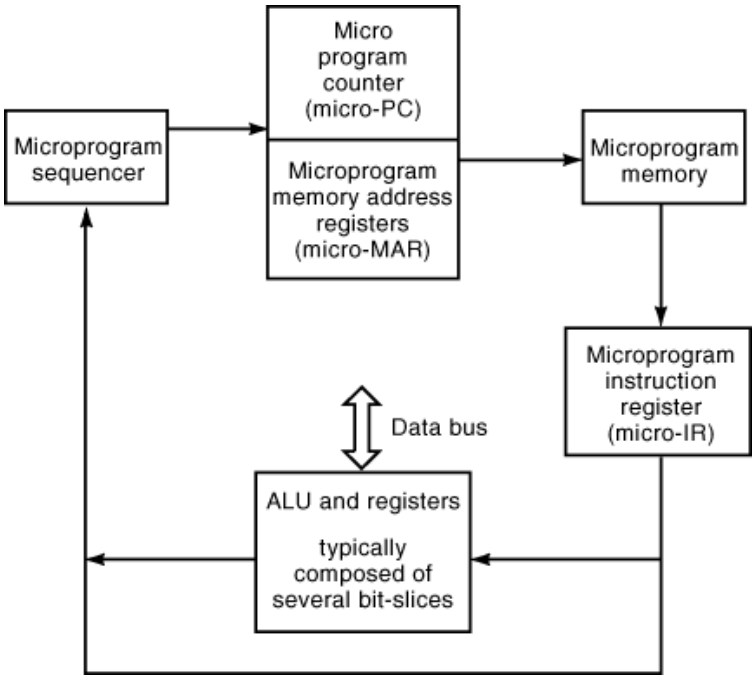

Fig. 7. Signal flow in a microprogrammed bit-slice ALU. The microprogram sequencer generates the address of the next microinstruction. The micro-PC sends this to the micro-MAR, which forwards the address to the microprogram memory. The corresponding microinstruction gets fetched to the micro-IR from the microprogram memory and provides appropriate control information to the ALU and the sequencer.

size can be handled by selecting the appropriate number of slices to use. This yielded significant scalability in design.

In the days when constructing computers using bit-slices was popular, the other alternatives were to use SSI (small scale integration)/MSI (medium-scale integration) building blocks or use fully functional microprocessors such as the Intel 8085/8086 or Motorola 6800/68000. Controllers for applications or custom processors using SSI/MSI option result in several hundreds of chips while the bit-slice option may only result in a few tens of chips. In comparison, an equivalent microprocessor-based system may only need three or four chips. The design time for bit-slice option was significantly lower than that for SSI/MSI chips, however comparable to that for microprocessor-based systems. Bit-slice chips provided more flexibility to the system architecture although SSI/MSI chips also provided enough flexibility. The cost would be the highest for the MSI option and lowest for the microprocessor option, with bit-slice cost lying in the middle range.

Bit-slice designs are modular. Because the design involves several building blocks interconnected in a structured manner, testing and debugging could be accomplished easily and systematically.

One significant advantage of traditional bit-slicing was that it made custom computing possible. Application-specific processors with custom instruction sets could be constructed relatively easily using bit-sliced ALU chips and microprogram sequencers. Although this might not be a feature of bit-slicing as such, it is only fair to count this as an advantage of bit-slices, considering the dedicated sequencer chips made available by bit-slice vendors and the popularity of microprogramming in conjunction with bit-slices.

Bit-Slice Computing. Although the term bit-slicing is traditionally used in the aforementioned meaning where one could construct a wider data path processor using smaller bit-slices, another perspective of bit-sliced computers is the use of a wide processor to operate on narrow slices of data. If you have a processor

Table 1. Partial List of Popular Bit-Slice Parts

Advanced Micro Devices[a]
Am2901—4-bit ALU
Am2902—4-bit Carry lookahead logic
Am2903—4-bit ALU similar to Am2901, but with improved func ality and cascadability
Am2904—Status and control device
Am2909—Microprogram sequencer
Am2910—Microprogram sequencer
Am2911—Microprogram sequencer
Am2913/14—Interrupt control unit
Am2930—Program control unit (Fetcher, Program Counter, etc.
Am2940—DMA address generator
Texas Instruments
SN74AS888—8-bit bit-sliced ALU
SN74AS890—Microprogram sequencer for the SN74AS888
74LS181—4-bit ALU
74LS182—Carry look ahead generator

[a] AMD introduced the 2900 series of bit-slice components in 1976.

that operates on N-bit integers, it is possible to devise algorithms where each N-bit data consists of k parts, each N/k-bits wide. For instance, if a data item is only 1 bit, 64 different data items can be packed as one 64-bit item, a single instruction can operate on all of them at the same time. Not every instruction in a general purpose processor would yield any meaningful computation for such bit-slices, however bit-wise operations such as AND, OR, XOR, and other logical operations often can be employed in a useful way. A little more specialized design can also handle bit-slice arithmetic operations. For instance, if a 64-bit processor can be designed with 8-bit and 16-bit arithmetic operations in mind, the designer would provide adequate intermediate carry outputs and other intermediate signals from each 8-bit/16-bit slice. One single instruction can now perform eight 8-bit or four 16-bit operations. This is exactly what happens in the MMX multimedia instruction set extensions (4) introduced by major microprocessor manufacturers such as Intel and AMD in 1996. Essentially one is able to utilize a uniprocessor as a parallel machine with k slices of m bit data. This parallel machine follows the SIMD (single instruction multiple data) model of computing, meaning that a single instruction is simultaneously acting on multiple data elements. Single instruction multiple data processors such as STARAN built by Goodyear around 1972 performed bit-slice operations in addition to word operations and were often referred to as bit-slice processors (9). Many parallel processors and associative processors operate on bit-slices or the same bit position in a group of words and there have been specialized bit-slice computing algorithms for searching, vector addition, image blurring, matrix multiplication, and so on (9).

BIBLIOGRAPHY

1. J. Mick J. Brick *Bit-Slice Microprocessor Design*, New York: McGraw-Hill, 1980.
2. Texas Instruments, *SN74AS888 SN74AS890 Bit-Slice Processor User's Guide*, Texas Instruments, Dallas, Texas, 1985.
3. D. E. White *Bit-Slice Design: Controllers and ALUs*, New York: Garlan STPM Press, 1981.
4. A. Peleg U. Weiser The MMX technology extension to the intel architecture, *IEEE Micro.*, **16** (4): 42–50, 1996.

5. J. P. Hayes *Computer Architecture and Organization*, New York: McGraw-Hill, 1988.
6. M. Morris Mano *Computer System Architecture*, 2nd ed., Englewood Cliffs, NJ: Prentice-Hall, 1982.
7. V. P. Nelson *74AS-EVM-16 Lab Manual No. 1, A Microprogramming Approach to Application-Specific Instruction Set Processor Design*, Texas Instruments, Dallas, Texas, 1987.
8. V. P. Nelson D. W. Jacobson *74AS-EVM-16 Lab Manual No. 2, Processor Design Using Microprogramming and Bit-Slice Techniques*, Texas Instruments, Dallas, Texas, 1987.
9. R. J. Baron L. Higbie *Computer Architecture*, Reading, MA: Addison-Wesley, 1992.

LIZY K. JOHN
The University of Texas at Austin
EUGENE B. JOHN
The University of Texas—Pan American