

$R1$  and the address is in  $AR$ . The write operation can be stated symbolically as follows:

$$\text{Write: } M[AR] \leftarrow R1$$

This causes a transfer of information from  $R1$  into the memory word  $M$  selected by the address in  $AR$ .

## 4-4 Arithmetic Microoperations

---

A microoperation is an elementary operation performed with the data stored in registers. The microoperations most often encountered in digital computers are classified into four categories:

1. Register transfer microoperations transfer binary information from one register to another.
2. Arithmetic microoperations perform arithmetic operation on numeric data stored in registers.
3. Logic microoperations perform bit manipulation operations on nonnumeric data stored in registers.
4. Shift microoperations perform shift operations on data stored in registers.

The register transfer microoperation was introduced in Sec. 4-2. This type of microoperation does not change the information content when the binary information moves from the source register to the destination register. The other three types of microoperations change the information content during the transfer. In this section we introduce a set of arithmetic microoperations. In the next two sections we present the logic and shift microoperations.

The basic arithmetic microoperations are addition, subtraction, increment, decrement, and shift. Arithmetic shifts are explained later in conjunction with the shift microoperations. The arithmetic microoperation defined by the statement

$$R3 \leftarrow R1 + R2$$

*add*  
microoperation

specifies an *add* microoperation. It states that the contents of register  $R1$  are added to the contents of register  $R2$  and the sum transferred to register  $R3$ . To implement this statement with hardware we need three registers and the digital component that performs the addition operation. The other basic arithmetic microoperations are listed in Table 4-3. Subtraction is most often

*subtract  
microoperation*

implemented through complementation and addition. Instead of using the minus operator, we can specify the subtraction by the following statement:

$$R3 \leftarrow R1 + \overline{R2} + 1$$

$\overline{R2}$  is the symbol for the 1's complement of  $R2$ . Adding 1 to the 1's complement produces the 2's complement. Adding the contents of  $R1$  to the 2's complement of  $R2$  is equivalent to  $R1 - R2$ .

TABLE 4-3 Arithmetic Microoperations

Symbolic designation	Description
$R3 \leftarrow R1 + R2$	Contents of $R1$ plus $R2$ transferred to $R3$
$R3 \leftarrow R1 - R2$	Contents of $R1$ minus $R2$ transferred to $R3$
$R2 \leftarrow \overline{R2}$	Complement the contents of $R2$ (1's complement)
$R2 \leftarrow \overline{R2} + 1$	2's complement the contents of $R2$ (negate)
$R3 \leftarrow R1 + \overline{R2} + 1$	$R1$ plus the 2's complement of $R2$ (subtraction)
$R1 \leftarrow R1 + 1$	Increment the contents of $R1$ by one
$R1 \leftarrow R1 - 1$	Decrement the contents of $R1$ by one

The increment and decrement microoperations are symbolized by plus-one and minus-one operations, respectively. These microoperations are implemented with a combinational circuit or with a binary up-down counter.

The arithmetic operations of multiply and divide are not listed in Table 4-3. These two operations are valid arithmetic operations but are not included in the basic set of microoperations. The only place where these operations can be considered as microoperations is in a digital system, where they are implemented by means of a combinational circuit. In such a case, the signals that perform these operations propagate through gates, and the result of the operation can be transferred into a destination register by a clock pulse as soon as the output signal propagates through the combinational circuit. In most computers, the multiplication operation is implemented with a sequence of add and shift microoperations. Division is implemented with a sequence of subtract and shift microoperations. To specify the hardware in such a case requires a list of statements that use the basic microoperations of add, subtract, and shift (see Chapter 10).

## Binary Adder

To implement the add microoperation with hardware, we need the registers that hold the data and the digital component that performs the arithmetic addition. The digital circuit that forms the arithmetic sum of two bits and a previous carry is called a full-adder (see Fig. 1-17). The digital circuit that

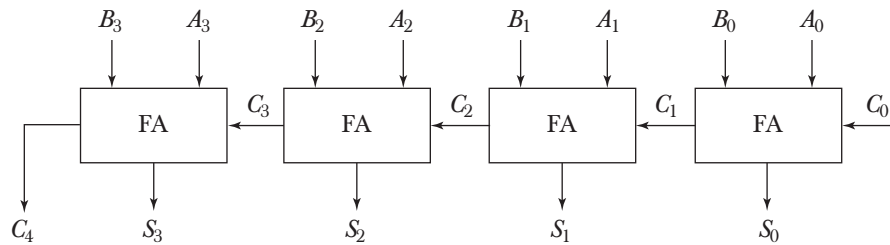


Figure 4-6 4-bit binary adder.

*binary adder**full-adder*

generates the arithmetic sum of two binary numbers of any length is called a binary adder. The binary adder is constructed with full-adder circuits connected in cascade, with the output carry from one full-adder connected to the input carry of the next full-adder. Figure 4-6 shows the interconnections of four full-adders (FA) to provide a 4-bit binary adder. The augend bits of  $A$  and the addend bits of  $B$  are designated by subscript numbers from right to left, with subscript 0 denoting the low-order bit. The carries are connected in a chain through the full-adders. The input carry to the binary adder is  $C_0$  and the output carry is  $C_4$ . The  $S$  outputs of the full-adders generate the required sum bits.

An  $n$ -bit binary adder requires  $n$  full-adders. The output carry from each full-adder is connected to the input carry of the next-high-order full-adder. The  $n$  data bits for the  $A$  inputs come from one register (such as  $R1$ ), and the  $n$  data bits for the  $B$  inputs come from another register (such as  $R2$ ). The sum can be transferred to a third register or to one of the source registers ( $R1$  or  $R2$ ), replacing its previous content.

### Binary Adder-Subtractor

The subtraction of binary numbers can be done most conveniently by means of complements as discussed in Sec. 3-2. Remember that the subtraction  $A - B$  can be done by taking the 2's complement of  $B$  and adding it to  $A$ . The 2's complement can be obtained by taking the 1's complement and adding one to the least significant pair of bits. The 1's complement can be implemented with inverters and a one can be added to the sum through the input carry.

*adder-subtractor*

The addition and subtraction operations can be combined into one common circuit by including an exclusive-OR gate with each full-adder. A 4-bit adder-subtractor circuit is shown in Fig. 4-7. The mode input  $M$  controls the operation. When  $M = 0$  the circuit is an adder and when  $M = 1$  the circuit becomes a subtractor. Each exclusive-OR gate receives input  $M$  and one of the inputs of  $B$ . When  $M = 0$ , we have  $B \oplus 0 = B$ . The full-adders receive the value of  $B$ , the input carry is 0, and the circuit performs  $A$  plus  $B$ . When  $M = 1$ , we have  $B \oplus 1 = B'$  and  $C_0 = 1$ . The  $B$  inputs are all complemented and a 1 is added through the input carry. The circuit performs the operation  $A$  plus the

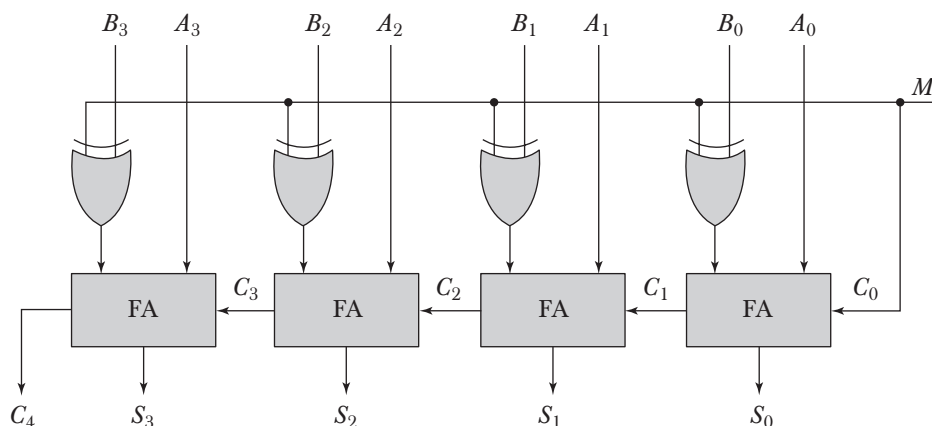


Figure 4-7 4-bit adder-subtractor.

2's complement of  $B$ . For unsigned numbers, this gives  $A - B$  if  $A \geq B$  or the 2's complement of  $(B - A)$  if  $A < B$ . For signed numbers, the result is  $A - B$  provided that there is no overflow.

### Binary Incrementer

The increment microoperation adds one to a number in a register. For example, if a 4-bit register has a binary value 0110, it will go to 0111 after it is incremented. This microoperation is easily implemented with a binary counter (see Fig. 2-10). Every time the count enable is active, the clock pulse transition increments the content of the register by one. There may be occasions when the increment microoperation must be done with a combinational circuit independent of a particular register. This can be accomplished by means of half-adders (see Fig. 1-16) connected in cascade.

*incrementer*

The diagram of a 4-bit combinational circuit incrementer is shown in Fig. 4-8. One of the inputs to the least significant half-adder (HA) is connected to logic-1 and the other input is connected to the least significant bit of the number to be incremented. The output carry from one half-adder is connected to one of the inputs of the next-higher-order half-adder. The circuit receives the four bits from  $A_0$  through  $A_3$ , adds one to it, and generates the incremented output in  $S_0$  through  $S_3$ . The output carry  $C_4$  will be 1 only after incrementing binary 1111. This also causes outputs  $S_0$  through  $S_3$  to go to 0.

The circuit of Fig. 4-8 can be extended to an  $n$ -bit binary incrementer by extending the diagram to include  $n$  half-adders. The least significant bit must have one input connected to logic-1. The other inputs receive the number to be incremented or the carry from the previous stage.

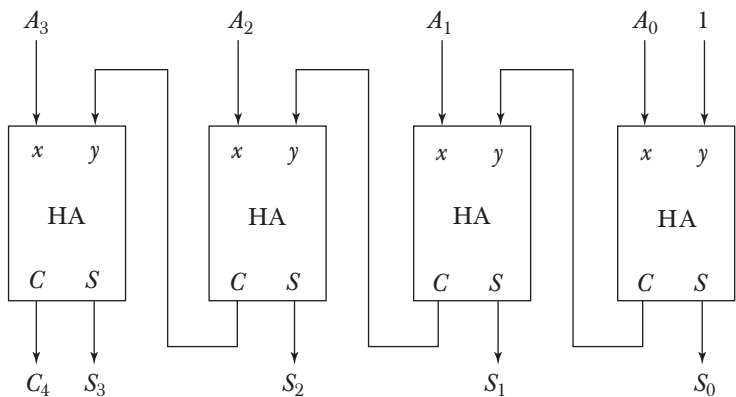


Figure 4-8 4-bit binary incrementer.

Arithmetic Circuit

arithmetic circuit

The arithmetic microoperations listed in Table 4-3 can be implemented in one composite arithmetic circuit. The basic component of an arithmetic circuit is the parallel adder. By controlling the data inputs to the adder, it is possible to obtain different types of arithmetic operations.

input carry

The diagram of a 4-bit arithmetic circuit is shown in Fig. 4-9. It has four full-adder circuits that constitute the 4-bit adder and four multiplexers for choosing different operations. There are two 4-bit inputs  $A$  and  $B$  and a 4-bit output  $D$ . The four inputs from  $A$  go directly to the  $X$  inputs of the binary adder. Each of the four inputs from  $B$  are connected to the data inputs of the multiplexers. The multiplexers data inputs also receive the complement of  $B$ . The other two data inputs are connected to logic-0 and logic-1. Logic-0 is a fixed voltage value (0 volts for TTL integrated circuits) and the logic-1 signal can be generated through an inverter whose input is 0. The four multiplexers are controlled by two selection inputs,  $S_1$  and  $S_0$ . The input carry  $C_{in}$  goes to the carry input of the FA in the least significant position. The other carries a connected from one stage to the next.

The output of the binary adder is calculated from the following arithmetic sum:

$$D = A + Y + C_{in}$$

where  $A$  is the 4-bit binary number at the  $X$  inputs and  $Y$  is the 4-bit binary number at the  $Y$  inputs of the binary adder.  $C_{in}$  is the input carry, which can be equal to 0 or 1. Note that the symbol  $+$  in the equation above denotes an arithmetic plus. By controlling the value of  $Y$  with the two selection inputs  $S_1$  and  $S_0$  and making  $C_{in}$  equal to 0 or 1, it is possible to generate the eight arithmetic microoperations listed in Table 4-4.

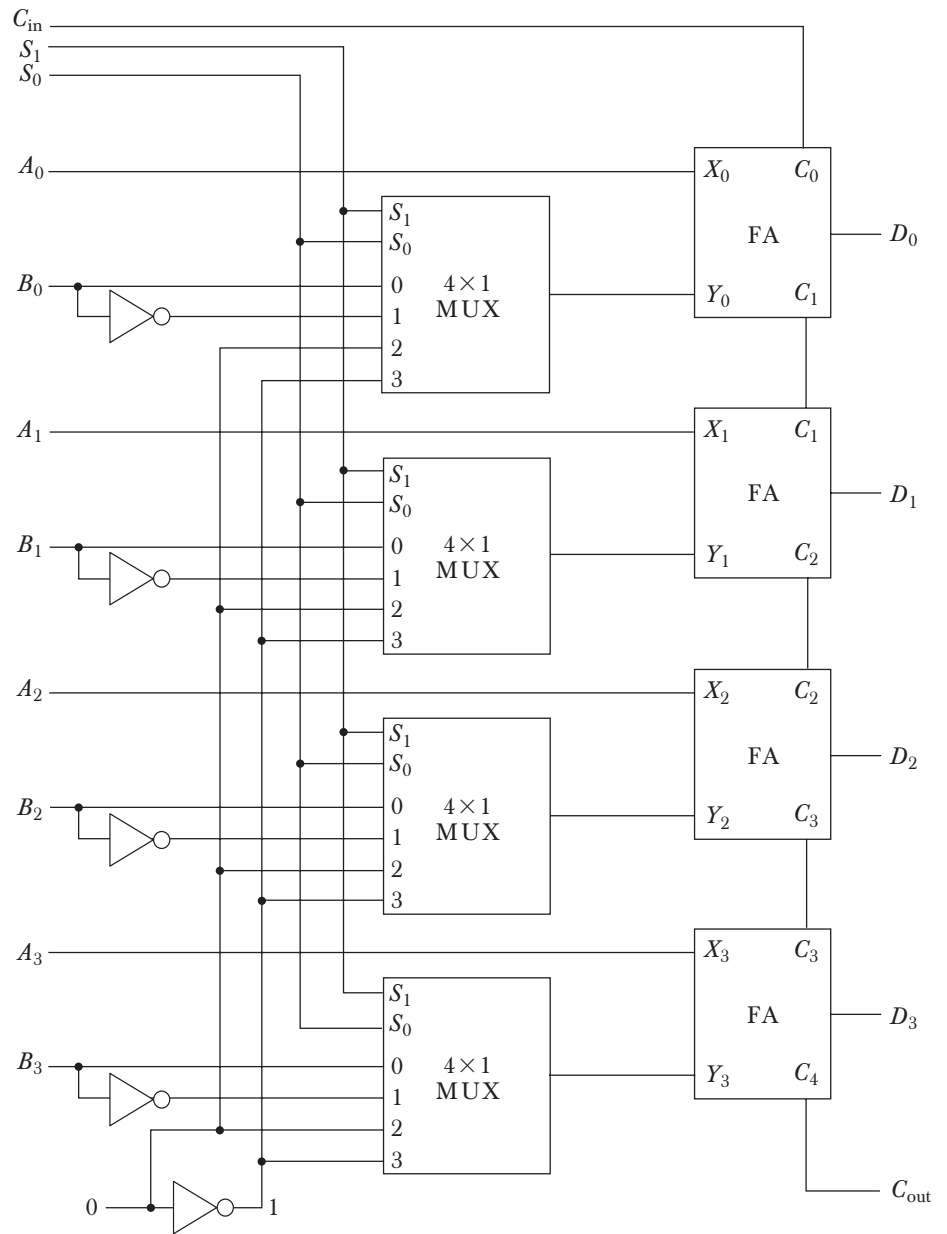


Figure 4-9 4-bit arithmetic circuit.