

1A

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.

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 one is added through the input carry. The circuit performs the operation A plus

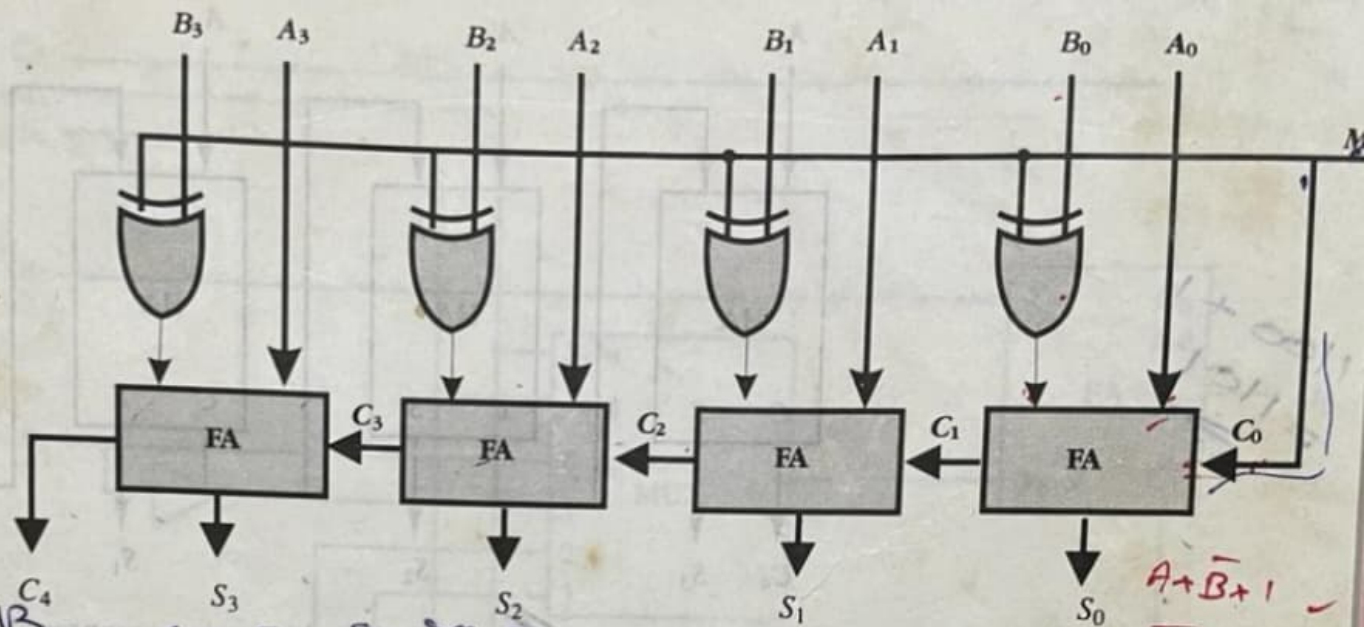


Figure 4-7 4-bit adder-subtractor.

$A = 0011$
 $B = 1011$
 $S_3 = 1$
 $S_2 = 1$
 $S_1 = 1$
 $S_0 = 0$

16

→ In bus transfer system, 16 registers of 32 bits each then.

1. How many multiplexers are needed?

16 registers

2. How many selection bits are needed?

32 bits each

3. What is the configuration of each multiplexer?

1Ans:- 32 (no. of bits)

2Ans:- 4 ($2^4 = 16$) ($2^2 = 4$)

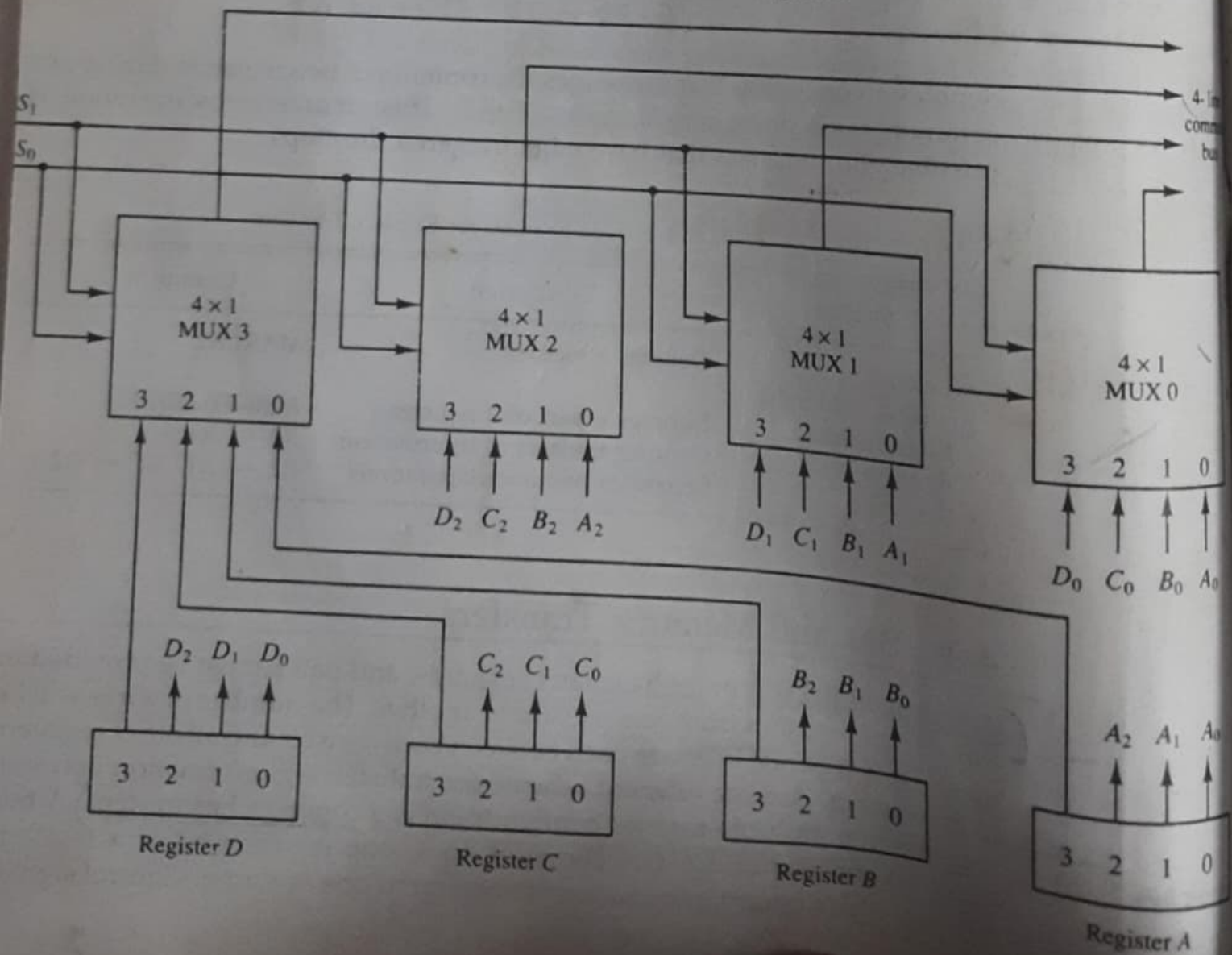
no. of control lines
Selection lines

3Ans:- 16x1 (no. of registers)

microoperations

Thus MUX 0 multiplexes the four 0 bits of the registers, MUX 1 multiplexes the four 1 bits of the registers, and similarly for the other two bits.

Figure 4-3 Bus system for four registers.



20

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 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 con-

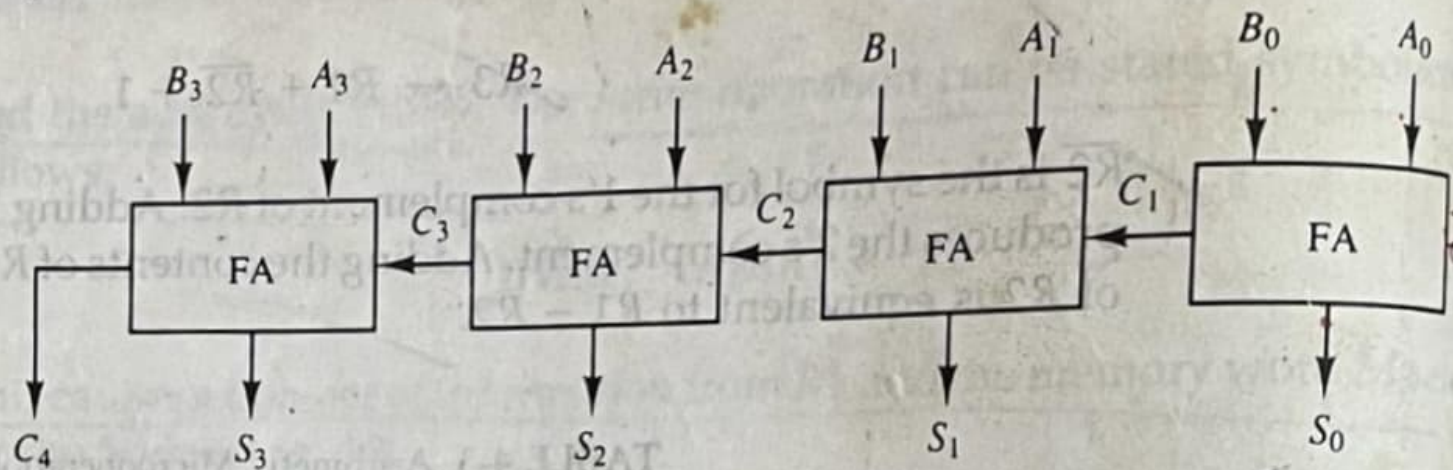
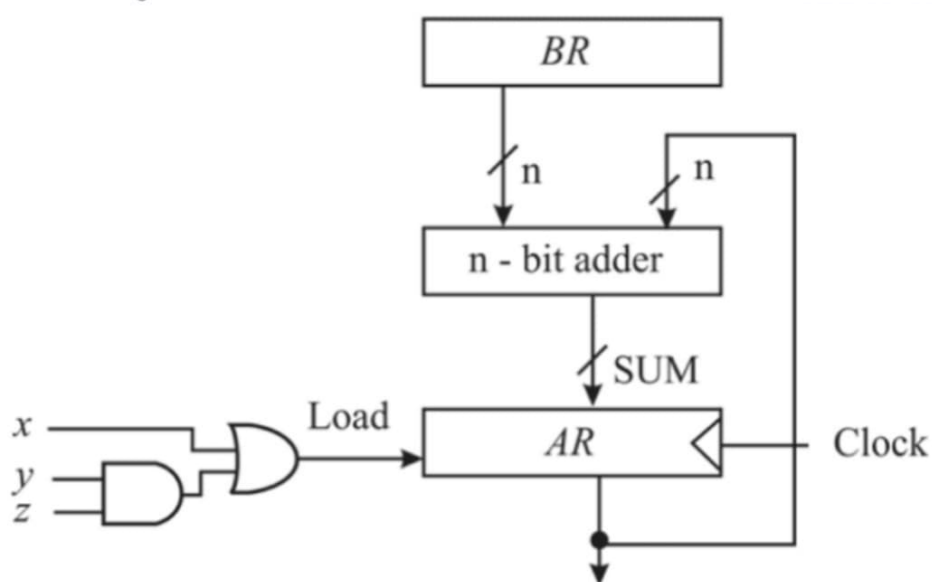


Figure 4-6 4-bit binary adder.

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 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.

2b



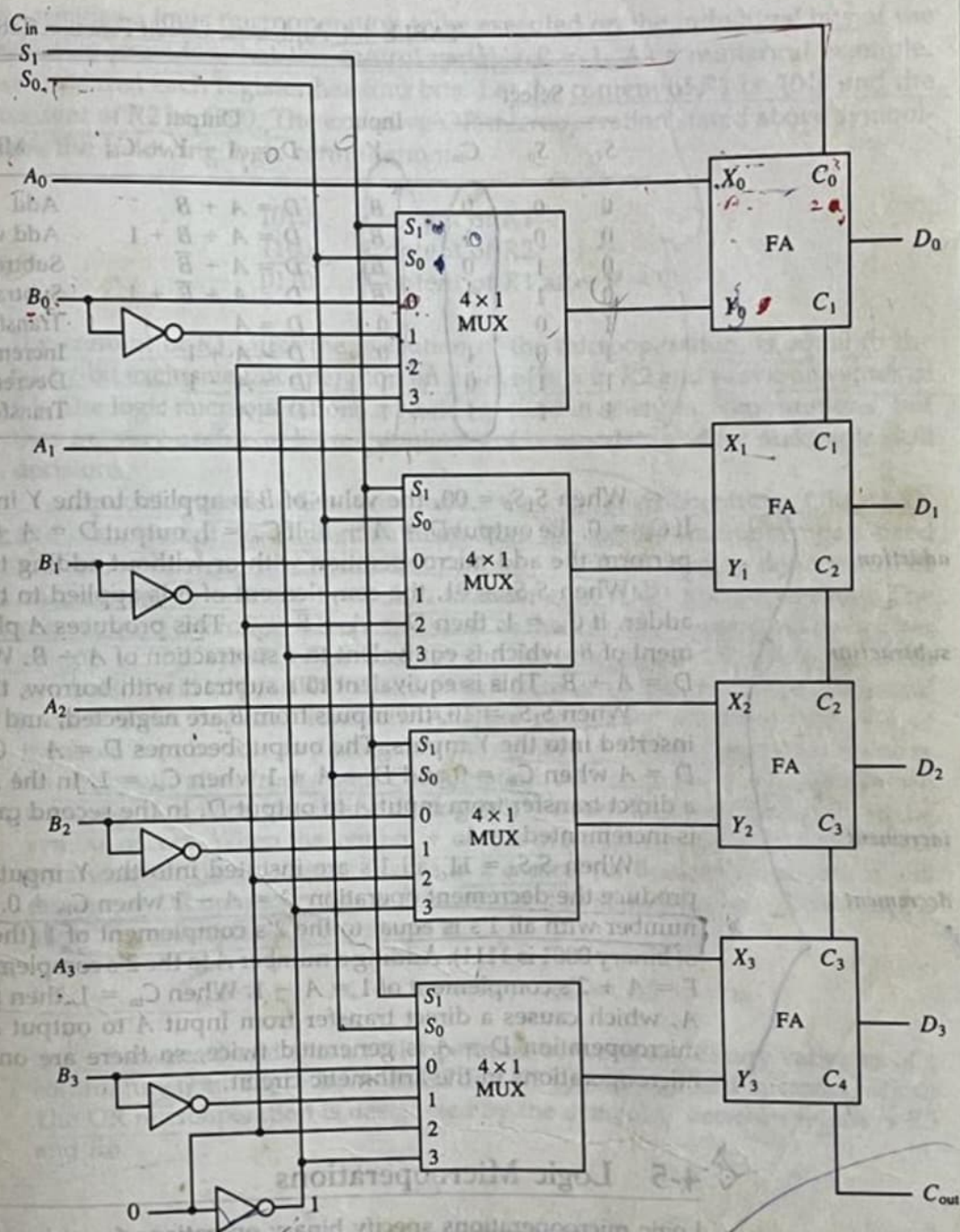


Figure 4-9 4-bit arithmetic circuit.

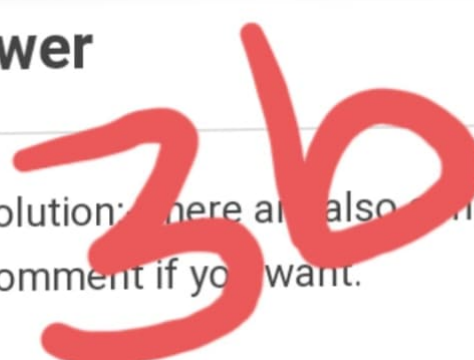


1 Answer



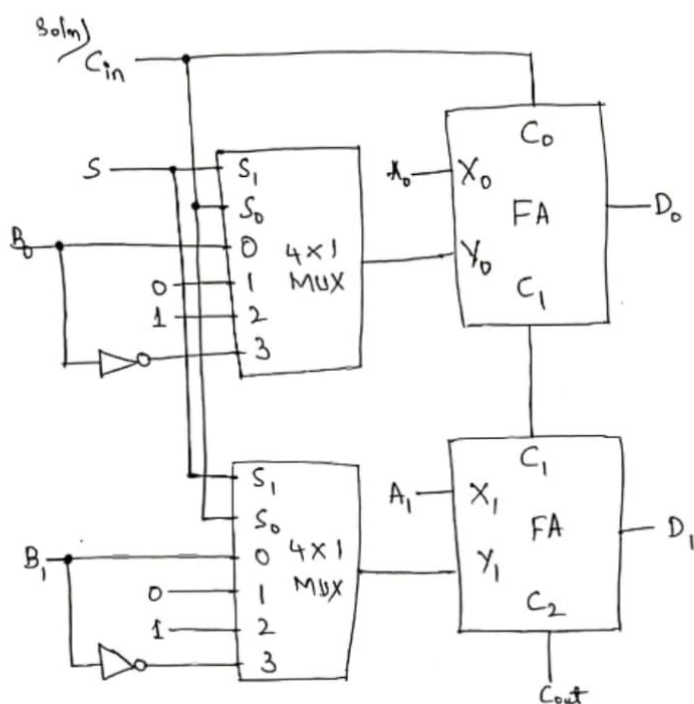
0



Solution: Here are also sample diagram. 

Comment if you want.

Thank you



S	C_{in}	X	Y	
0	0	A	B	$(A+B)$
0	1	A	0	$(A+1)$
1	0	A	1	$(A-1)$
1	1	A	\overline{B}	$(A-B)$

answered Sep 6



Mohammed Hussain

GA

Applications of logic microoperations

1. selective-set

- The selective-set operation sets to 1 the bits in register A where there are corresponding 1's in register B. It does not affect bit positions that have 0's in B. The following numerical example clarifies this operation:

1010	A before
<u>1100</u>	B (logic operand)
1110	A after

2. selective-complement

- The selective-complement operation complements bits in A where there are corresponding 1's in B. It does not affect bit positions that have 0's in B.

For example:

1010	A before
<u>1100</u>	B (logic operand)
0110	A after

3. Selective-clear

- The selective-clear operation clears to 0 the bits in A only where there are corresponding 1's in B.

For example:

1010	A before
<u>1100</u>	B (logic operand)
0010	A after

4. mask

- The mask operation is similar to the selective-clear operation except that the bits of A are cleared only where there are corresponding 0's in B. The mask operation is an AND micro operation as seen from the following numerical example:

1010	A before
<u>1100</u>	B (logic operand)
1000	A after masking

5. Insert

- The insert operation inserts a new value into a group of bits.
- This is done by first masking the bits and then ORing them with the required value.
- For example, an A register contains eight bits, 0110 1010. To replace the four leftmost bits by the value 1001.

First mask the four unwanted bits.

0110 1010	A before masking
<u>0000 1111</u>	B (mask)
0000 1010	A after masking

Now insert the new value.

0000 1010	A before insert
<u>1001 0000</u>	B (insert)
1001 1010	A after insertion

**The mask operation is an AND microoperation and the insert operation is an OR microoperation.

6. clear

- The clear operation compares the words in A and B and produces an all 0's result, if the two numbers are equal.
- This operation is achieved by an exclusive-OR microoperation as shown by the following example.

1010	A
<u>1010</u>	B
00	$A \leftarrow A \oplus B$
01	