CHAPTER - 9

ARITHMETIC LOGIC UNIT

9.1 Nibble Adder

An arithmetic logic unit (ALU) is a multi operation, combinational-logic digital function. It can perform a set of basic arithmetic operations and a set of logic operations. The ALU has a number of selection lines to select a particular operation in the unit. The selection lines are decoded within the ALU so that k selection variables can specify up to 2^k distinct operations.

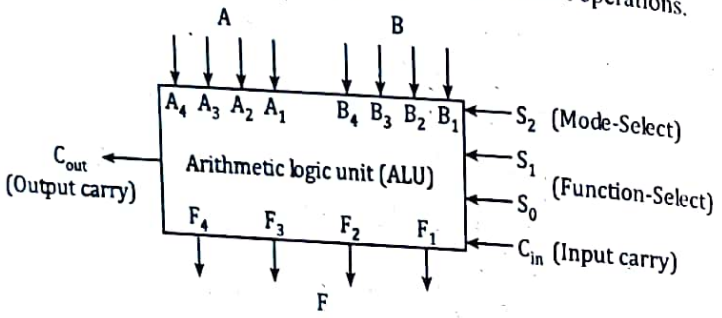


Fig.: Block diagram of a 4-bit ALU

The four data inputs from A are combined with the four inputs from B to generate an operation at the F outputs the mode select input S_2 distinguishes between arithmetic and logic operations. The two function select inputs S_1 and S_0 specify the particular arithmetic or logic operation to be generated. With three selection variables, it is possible to specify four arithmetic operations (with S_2 in one state) and four logic operations (with S_2 in the other state).

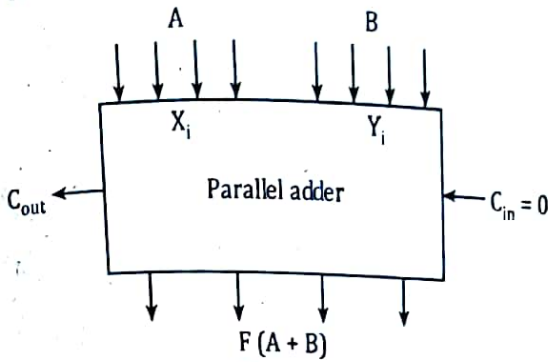
9.2 Arithmetic Unit

Example:

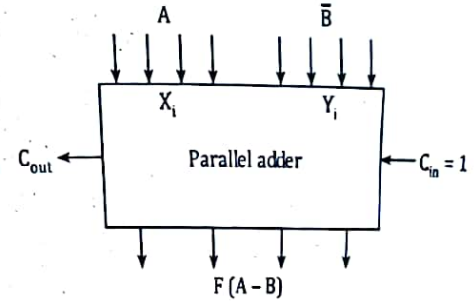
Design an adder subtractor circuit with one selection variable S and two inputs A and B. Where S = 0, circuit performs A + B and when S = 1, circuit performs A – B by taking 2's complement of B.
[Fall 2019,Spring 2018,Spring 2015].

Solution:

Step 1:



(a) Addition



(b) Subtraction

Step 2:

S	X_i	Y_i	C_{in}
0	A	B	0
1	A	$\bar{B}$	1

Step 3:

Truth table

S	A	B	X_i	Y_i
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	0
1	1	0	1	1
1	1	1	1	0

K-map for Y_i

S \ AB	00	01	11	10
0	0	1	1	0
1	1	0	0	1

$Y_i = S'B + SB' = S \oplus B$

$X_i = A$

$S = C_m$

Step 4: Logic diagram

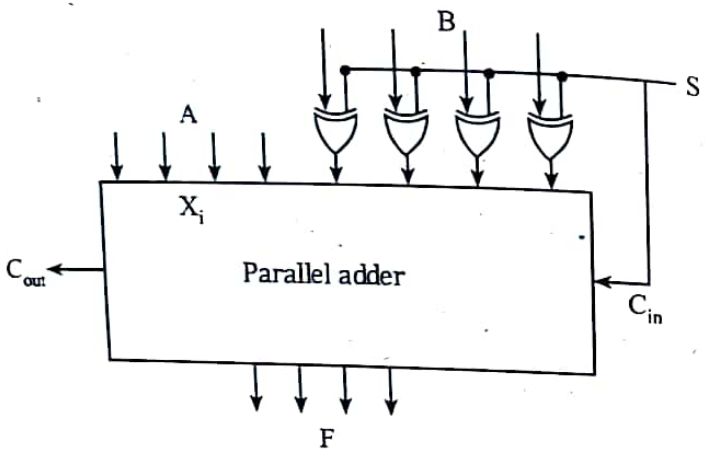


Fig.: Logic diagram of arithmetic circuit

Example:

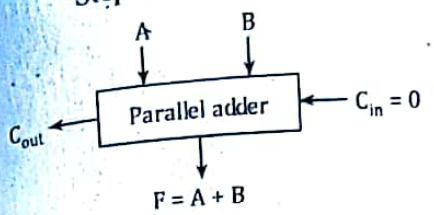
Design arithmetic circuit that performs following operations

1. **Addition (A + B)**
2. **Addition with carry (A + B + 1)**
3. **A plus 1's complement of B (A + $\overline{B}$)**
4. **Subtraction (A + $\overline{B}$ + 1)**
5. **Transfer A, using B = 0**
6. **Increment A (A + 1), B = 0**
7. **Decrement a (A – 1), B = 1**
8. **Transfer A , using B = 1**

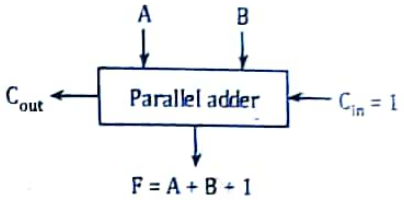
[Fall 2018,Spring 2017,Spring 2013]

Solution:

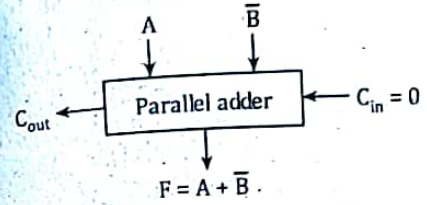
Step 1:



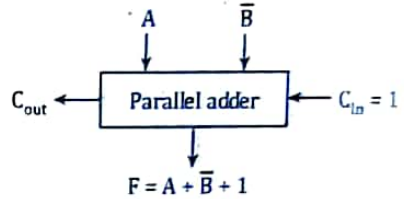
1. Addition



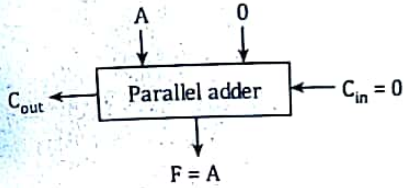
2. Addition with carry



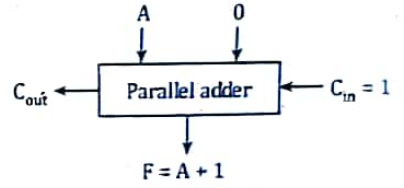
3. A plus 1's complement of B



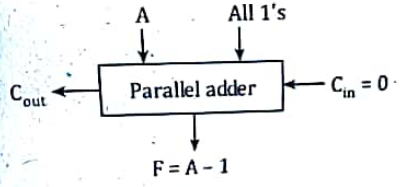
4. Subtraction



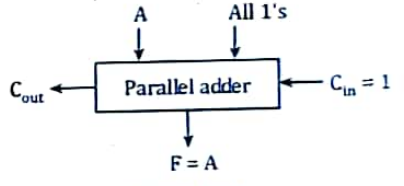
5. Transfer A



6. Increment A



7. Decrement A



8. Transfer A

Fig.: Operation obtained by controlling one set of inputs to a parallel adder

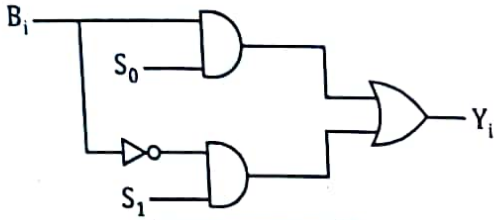
Step 2:

Function table

Function select			X equals	Y equals	Output equals	Function
S_1	S_0	C_{in}				
0	0	0	A	0	F = A	Transfer A
0	0	1	A	0	F = A + 1	Increment A
0	1	0	A	B	F = A + B	Add B to A

Function select			X equals	Y equals	Output equals	Function
S_1	S_0	C_{in}				
0	1	1	A	B	F = A + B + 1	Add B to A plus 1
1	0	0	A	$\overline{B}$	F = A + $\overline{B}$	Add 1's complement of B to A
1	0	1	A	$\overline{B}$	F = A + $\overline{B}$ + 1	Add 2's complement of B to n
1	1	0	A	All 1's	F = A – 1	Decrement A

The circuit that controls input B to provide the function illustrate in step 1 is called a true/complement, one/zero element.



S_1	S_0	Y_i
0	0	0
0	1	B_i
1	0	$\overline{B}_i$
1	1	1

Step 3:

Truth table

S_1	S_0	B_i	Y_i
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

K-map for Y_i

S_1 \ S_0B	00	01	11	10
0	0	0	1	0
1	1	0	1	1

$Y_i = S_0B_i + S_1B_i'$

$X_i = A_i$

Step 4:

Final logic diagram

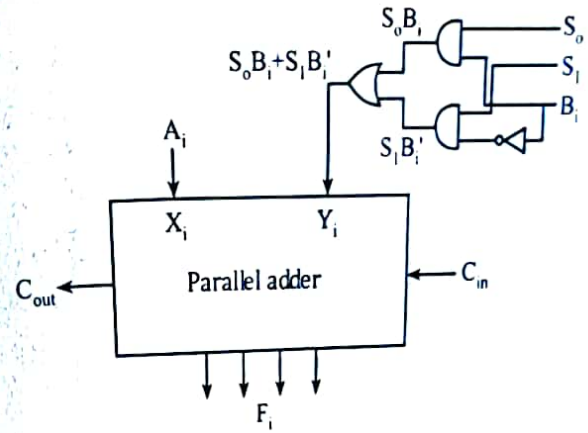


Fig.: Logic diagram of arithmetic circuit

9.3 Logic Unit

The logic microoperations manipulate the bits of the operands separately and treat each bit as a binary variable. With 'n' variables we can create 2^{2n} functions. If n = 2, we can create 16 function but all these 16 function can be generated using 'AND', 'OR' & 'NOT' operation.

Example:

Design a logic circuit, which can perform following logic operation.

a. OR
b. AND
c. AND
d. NAND
e. NOT A
f. NOT B
g. X-OR
h. X-NOR

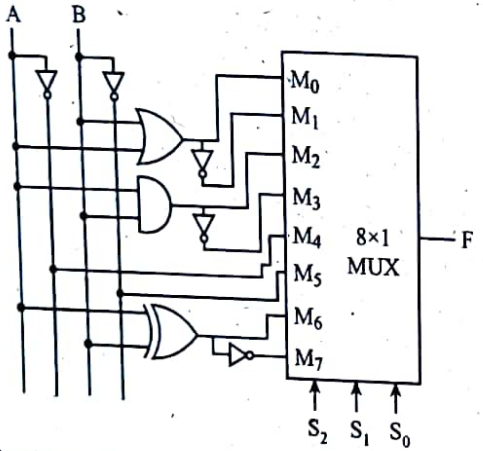
Solution:

For 8 functions, we use 3 selection lines i.e., S_2, S_1, S_0 and input A, B.

Function table

Function select			Output equals	Function
S_2	S_1	S_0		
0	0	0	A + B	OR
0	0	1	$\overline{A+B}$	NOR
0	1	0	A.B	AND
0	1	1	$\overline{A.B}$	NAND
1	0	0	$\overline{A}$	NOT A
1	0	1	$\overline{B}$	NOT B
1	1	0	$A \oplus B$	X–OR
1	1	1	$A \odot B$	X–NOR

Logic diagram using 8 × 1 mux



The circuit must be repeated 'n' times for n bit logic circuit. The figure above generates 8 logic operation with 3 selection variables.

9.3 Design of Arithmetic and Logic Unit

Here, we design an ALU with eight arithmetic operations and fair logic operations. Three selection variables S_2, S_1 and S_0 select eight different operations, and the input carry C_{in} is used to select four additional arithmetic operations. With $S_2 = 0$, selection variables S_2 and S_0 together with C_{in} will select the eight arithmetic operation with $S_2 = 1$, variables S_1 and S_0 will select the four logic operations OR, XOR, AND and NOT.

The steps involved in the design of an ALU are as follows:

1. Design the arithmetic section independent of the logic section.
2. Determine the logic operations obtained from the arithmetic circuit in step 1, assuming that the input carries to all stages are 0.
3. Modify the arithmetic circuit to obtain the required logic operations.

The Function Table

Selection				Output	Function
S_2	S_1	S_0	C_{in}		
0	0	0	0	F = A	Transfer A
0	0	0	1	F = A + 1	Increment A
0	0	1	0	F = A + B	Addition
0	0	1	1	F = A + B + 1	Add with carry
0	1	0	0	F = A – B – 1	Subtract with borrow
0	1	0	1	F = A – B	Subtraction
0	1	1	0	F = A – 1	Decrement A
0	1	1	1	F = A	Transfer A
1	0	0	×	$F = A \cup B$	OR
1	0	1	×	$F = A \oplus B$	XOR
1	1	0	×	$F = A \cap B$	AND
1	1	1	×	$F = \overline{A}$	Complement A

The inputs to each full adder circuit are specified by the boolean functions.

$X_i = A_i + S_2S_1'S_0'B_i + S_2S_1S_0'B_i'$

$Y_i = S_0B_i + S_1B_i'$

$Z_i = S_2'C_{in}$

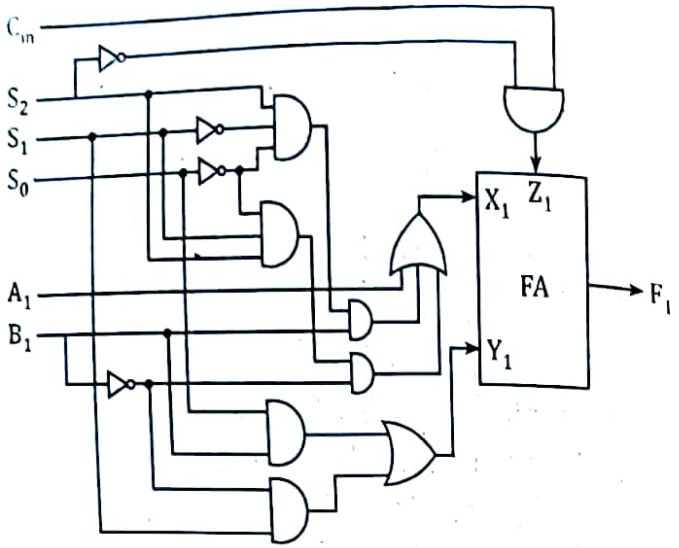


Fig.: One bit logic diagram of arithmetic logic unit (ALU)

9.4 Status Register

It is sometimes convenient to supplement the ALU with a status register where these status-bit conditions are stored for further analysis. Status-bit conditions are sometimes called condition-code bits or flag bits.

Figure below shows the block diagram of an 8-bit ALU with a 4-bit status register. The four status bits are symbolized by C, S, Z & V. The bits are set or cleared as a result of a operation performed in the ALU.

1. Bit C is set if the output carry of the ALU is 1. It is cleared if the output carry is 0.
2. Bit S is set if the highest- order bit of the result in the output of the ALU is 1. It is cleared if the highest-order bit is 0.
3. Bit Z is set if the output of the ALU contains all 0's and cleared otherwise.
4. Bit V is set if exclusive - OR of carries C_8 and C_9 is 1, and cleared otherwise. This is the condition for overflow when the numbers are in sign-2's - complement representation. For the 8-bit ALU, V is set if the result is greater than 127 or less than –128.

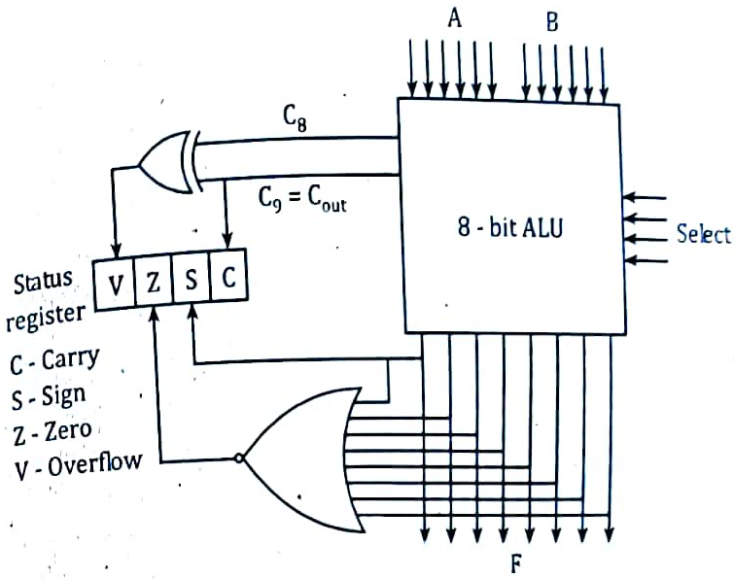


Fig.: Setting bits in a status register

9.5 Design of Shifter

Shifter

The shift unit attached to a processor transfers the output of the ALU into the output bus. The shifter may transfer the information directly without a shift, or it may shift the information to the right or left. Provision is sometimes made for no transfer from the ALU to the output BUS. The shifter provides the shift micro-operations commonly not available in an ALU.

Design of Shifter

A combinational logic shifter can be constructed with multiplexers as shown in figure below. The two selection variables S_1 and S_0, applied to all four multiplexers select the type of operation in the shifter. When $S_1S_0 = 00$, no shift is executed and the signals from A go directly to the F lines. The next two selection variable causes a shift right operation and a shift-left operation. When $S_1S_0 = 11$, the multiplexers select the inputs attached to 0 and as a consequence the F outputs are also equal to 0 blocking the transfer of information from the ALU to the output bus.

Function table for shifter:

S_1	S_0	Operation	Function
0	0	F ← A	Transfer A to S (No sift)
0	1	F ← Shr A	Shift right A into F
1	0	F ← Shl A	Shift left A into F
1	1	F ← 0	Transfer 0's into S

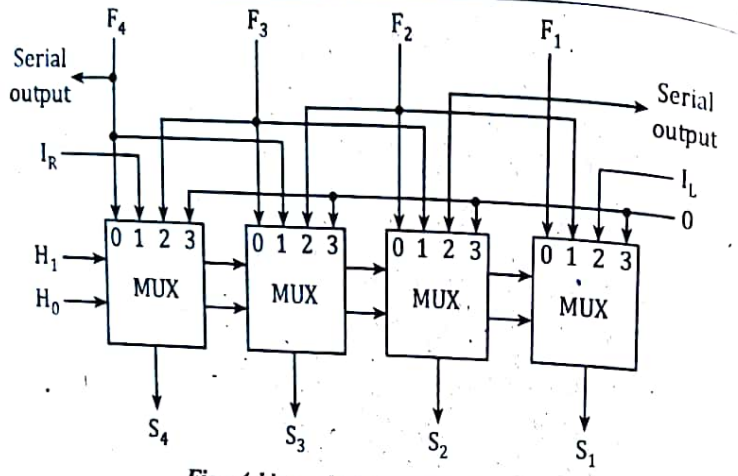


Fig.: 4-bit combination logic shifter

9.6 Processor Unit

The integral part of **Central Processing Unit (CPU)** of a computer is ALU. It comes in various forms with wide range of functionality. Other than normal addition, subtraction; it can perform increment, decrement operations. As logic unit, it performs usual AND, OR, NOT, EX-OR and many other complex logic functions. It also comes with PRESET and CLEAR option which makes the function outputs 1 and 0 respectively. Normally, a mode selector input (M) decides whether ALU performs a logic operation or an arithmetic operation. In each mode, different functions are chosen by appropriately activating a set of selection input.

9.7 Design of Accumulator

Some processor unit distinguish one register form all other and called accumulator register. Accumulator register is essentially bi-directional shift register with parallel load which is connected to an ALU. Because of

feedback connection from the register to the input of ALU, the accumulator register and its associated logic when taken as one unit constitute a sequential circuit. The register A in the figure is referred to as accumulator and is sometimes denoted by the symbol AC. The external input to the accumulator are the data inputs and control variables determine the micro-operation for the register. An accumulator is a multi function register that by itself can be made to perform all the micro-operation of processor unit.

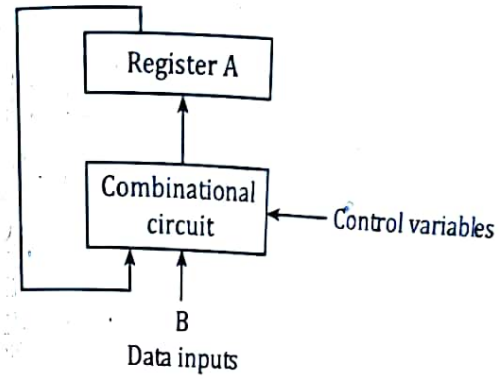


Fig.: Block diagram of accumulator

Design of Accumulator

The set of micro-operation for the accumulator is given in table below. Control variable S_1 to S_4 are generated by control logic circuits and should be considered as control functions that initiate the corresponding register transfer operation. Register A is source register and it represents the present state of the sequential circuit. The B register is used as a second source register for micro-operations. The four control variables sent to sequential circuit is also considered as input to circuit.

List of micro-operations:

Control variable	Micro-operation	Name
S_1	$A \leftarrow A + B$	Add
S_2	$A \leftarrow 0$	Clear
S_3	$A \leftarrow \overline{A}$	Complement
S_4	$A \leftarrow A.B$	AND

1. Add (A + B)

The excitation table for input for the J-K flipflop are listed below:

Present state	Inputs		Next state	Flip-flop input		Output
A_i	B_i	C_i	A_{i+1}	J_{Ai}	K_{Ai}	C_{i+1}
0	0	0	0	0	×	0
0	0	1	1	1	×	0
0	1	0	1	1	×	0
0	1	1	0	0	×	1
1	0	0	1	×	0	0
1	0	1	0	×	1	1
1	1	0	0	×	1	1
1	1	1	1	×	0	1

K-map for J_{Ai}

A_i \ B_iC_i	00	01	11	10
0	0	1	0	1
1	×	×	×	×

$J_{Ai} = B_i'C_i + B_iC_i'$

$= B_i \oplus C_i$

K-map for K_{Ai}

A_i \ B_iC_i	00	01	11	10
0	×	×	×	×
1	0	1	0	1

$K_{Ai} = B_i'C_i + B_iC_i'$

$= B_i \oplus C_i$

K-map for C_{i+1}

A_i \ B_iC_i	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$C_{i+1} = B_iC_i + A_iC_i + A_iB_i$

$= A_iB_i + A_iC_i + B_iC_i$

The J input of flip-flop Ai, designated J_{Ai}, and K input of flip-flip A designated by K_{Ai}, do not include the control variable S_i. These two equation should affect the flip-flop only when S_i is enable: therefore they should be ANDed with control variable S_1, S_0.

$J_{Ai} = (B_i \oplus C_i)\, S_1$ $\qquad C_i = A_iB_i + A_iC_i + B_iC_i$

$K_{Ai} = (B_i \oplus C_i)\, S_1$

2. Clear (S_2)

$J_{Ai} = 0$

$K_{Ai} = S_2$

It clears all the flip-flops in register A to cause this transition in a J-K flip-flop. We need to apply control variable S_2 only to K_{Ai}, input of flip-flop. The J input will be assumed to be zero if nothing is applied to it.

3. Complement (S_3)

$J_{Ai} = S_3$

$K_{Ai} = S_3$

For this we need to apply S_3 to both J and K input such that $J_{Ai} = S_3$ and $K_{Ai} = S_3$

4. AND (S_4)

Present state		Next state	Flip-Flop	
A_i	B_i	A_{i+1}	J_{Ai}	K_{Ai}
0	0	0	0	×
0	1	0	0	×
1	0	0	×	1
1	1	1	×	0

Using K-map,

For J_{Ai}

A_i \ B_i	0	1
0	0	0
1	×	×

$J_{Ai} = 0$

For K_{Ai}

A_i \ B_i	0	1
0	×	×
1	1	0

Since, AND operation is enabled when $S_4 = 1$. Thus, the input must be AND with S_4

$J_{Ai} = 0$

$K_{Ai} = B_i'S_4$

Thus, the final logic statement for input J_{Ai} and K_{Ai} is,

$J_{Ai} = B_iC_i'S_1 + B_i'C_iS_1 + S_3$

$C_{i+1} = A_iB_i + A_iC_i + B_iC_i$

$K_{Ai} = B_iC_i'S_1 + B_i'C_iS_1 + S_2 + S_3 + B_iS_4$

Final logic circuit

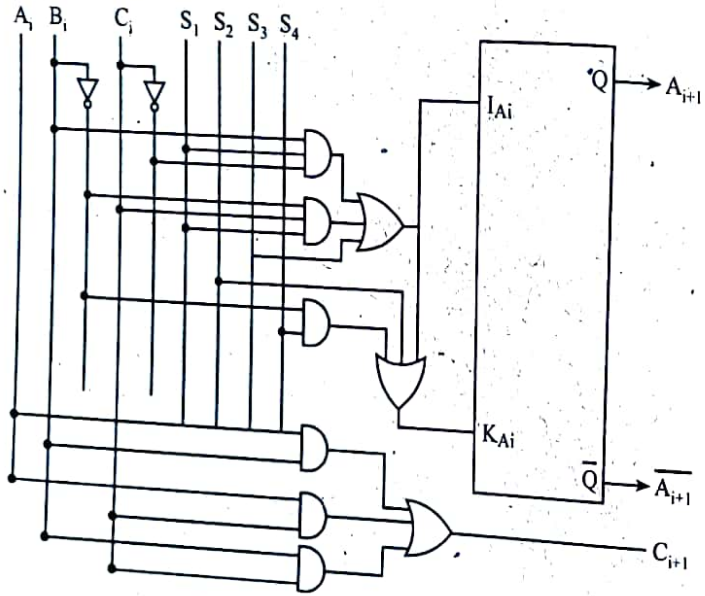


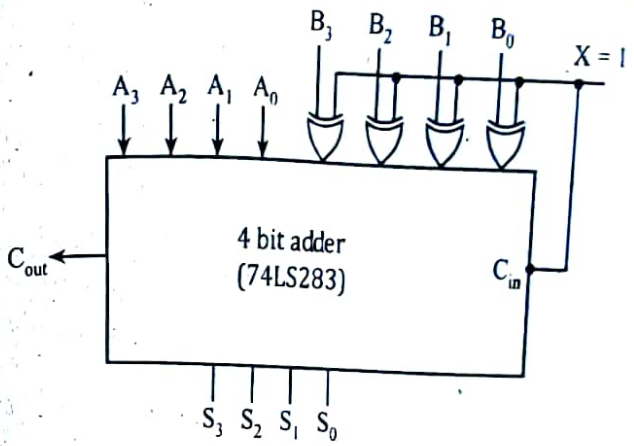
Fig.: Accumulator circuit (1 stage)

SOLUTION TO IMPORTANT AND EXAM QUESTIONS

1. Design a circuit for 4-bit full subtractor. [Fall 2016]

Solution:

Let 4 bit inputs are $A_3\ A_2\ A_1\ A_0$ and $B_3\ B_2\ B_1\ B_0$ and carry = C_{in}.



When X = 1, we get complement of B and $C_{in} = 1$.

∴ Difference = $(A_3\ A_2\ A_1\ A_0)$ + (1's complement of $(B_3\ B_2\ B_1\ B_0)$ + C_{in}

= $(A_3\ A_2\ A_1\ A_0)$ + (1's complement of $B_3\ B_2\ B_1\ B_0$) + 1

= $(A_3\ A_2\ A_1\ A_0)$ + (2's complement of $B_3\ B_2\ B_1\ B_0$)