Solution to The Class Test-2017

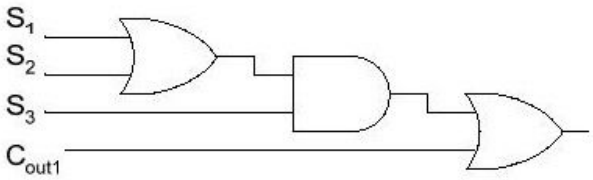
Digital System Design (CSE- 2111)

1.Q: Draw the logic diagram of a BCD adder and discuss its operation?

BCD binary numbers represent Decimal digits 0 to 9. A 4-bit BCD code is used to represent the ten numbers 0 to 9. Since the 4-bit Code allows 16 possibilities, therefore the first 10 4-bit combinations are considered to be valid BCD combinations. The latter six combinations are invalid and do not occur.BCD Code has applications in Decimal Number display Systems such as Counters andDigital Clocks. BCD Numbers can be added together using BCD Addition. BCD Addition is similar to normal Binary Addition except for the case when sum of two BCD digits exceeds 9 or a Carry is generated. When the Sum of two BCD numbers exceeds 9 or a Carry is generated a 6 is added to convert the invalid number into a valid number. The carry generated by adding a 6 to the invalid BDC digit is passed on to the next BCD digit.Addition of two BCD digits requires two 4-bit Parallel Adder Circuits. One 4-bit Parallel Adder adds the two BCD digits. A BCD Adder uses a circuit which checks the result at the output of the first adder circuit to determine if the result has exceeded 9 or a Carry has been generated. If the circuit determines any of the two error conditions the circuit adds a 6 to the original result using the second Adder circuit. The output of the second Adder gives the correct BCD output. If the circuit finds the result of the first Adder circuit to be a valid BCD number (between 0 and 9 and no Carry has been generated), the circuit adds a zero to the valid BCD result using the second Adder. The output of the second Adder gives the same result.

The circuit that checks if the output of the first Adder has exceeded 9 is a simple combinational circuit with the function table specified.
The Boolean expression for the Invalid BCD Number Detector obtained from the

Karnaugh Map which maps the function table is S 3S 2 + S 3S1 = S 3 (S 2 + S1)
The Invalid BCD Number is represented by two error conditions, either the BCD number is one
of the invalid numbers or a Carry out has been generated. Therefore the complete expression
for determining an incorrect BCD output is Cout1 + S 3 (S 2 + S1) .



Input				Output	Input				Output
S_3	S_2	S_1	S_0	F	S_3	S_2	S_1	S_0	F
0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	1	0	1
0	0	1	1	0	1	0	1	1	1
0	1	0	0	0	1	1	0	0	1
0	1	0	1	0	1	1	0	1	1
0	1	1	0	0	1	1	1	0	1
0	1	1	1	0	1	1	1	1	1

Table 15.1 Function Table of Invalid BCD Number detector

S_3S_2 \ S_1S_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

Figure 15.2 Mapping of Invalid BCD Number detector function

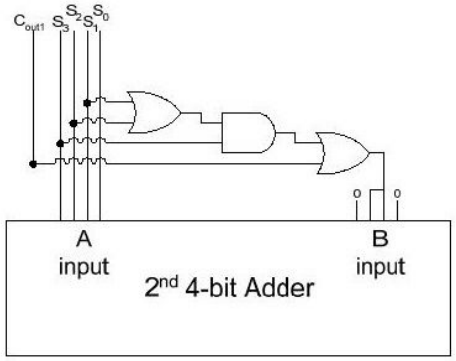
Connection of Invalid BCD Detector Circuit to second Adder

Adding of 6 when error conditions are detected and adding a zero when errorconditions are not detected is implemented by connecting the output of the Invalid BCDNumber Detector circuit to bits B_1 and B_2 of the Adder. Bits B_0 and B_3 are permanently connected to 0. Figure 15.4. When an error condition is detected the output of the circuit is set

to logic 1, setting bits B_1 and B_2 to 1 and the 2nd Adder input B to 0110. When the error condition is not detected the circuit output is 0 and the 2nd Adder input B is set to 0000

2-digit BCD Adder

Two singe digit BCD Adders can be cascaded together to form a 2-digit BCD Adder. Four, 4-bit 74LS283 MSI chips are used. Two 74LS283s are required to directly add the two 2-digit BCD numbers and the remaining two 74LS283s are required to add a six to the result if any of the two digits add up to invalid BCD digits or generate a Carry. Two invalid BCD detector circuits are used



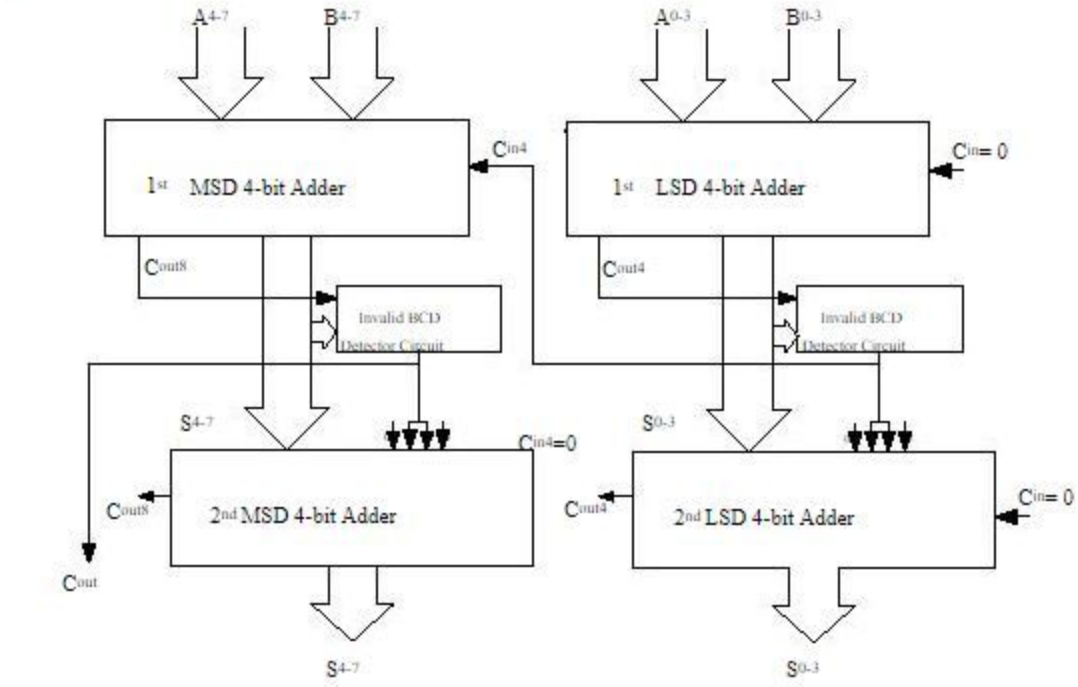
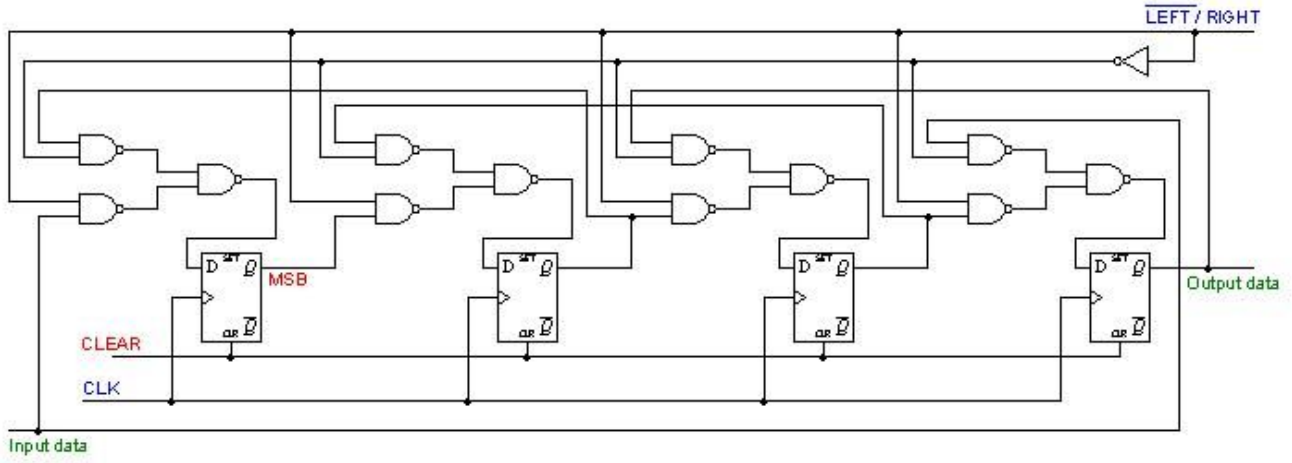


Figure 15.5 2-Digit BCD Adder

2.Q: Design a 4-bit bi -directional shift registor?

The registers discussed so far involved only right shift operations. Each right shift operation has the effect of successively dividing the binary number by two. If the operation is reversed (left shift), this has the effect of multiplying the number by two. With suitable gating arrangement a serial shift register can perform both operations. A *bidirectional*, or *reversible*, shift register is one in which the data can be shift either left or right. A four-bit bidirectional shift register using D flip-flops is shown below.



Here a set of NAND gates are configured as OR gates to select data inputs from the right or left adjacent bistables, as selected by the LEFT/RIGHT control line.
The animation below performs right shift four times, then left shift four times. Notice the order of the four output bits are not the same as the order of the original four input bits. They are actually reversed!

3.Q: Design a synchronus counter usinh j-k FF to count the following sequnce 1,3,15,5,8,2,0,12,6,,9 and repeat.

State Transistion Table:

Q3	Q2	Q1	Q0		Q3	Q2	Q1	Q0	J3	K3	J2	K2	J1	K1	J0	K0
0	0	0	0		0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1		0	0	1	1	0	X	0	X	1	X	X	0
0	0	1	0		0	0	0	0	0	X	0	X	X	1	0	X
0	0	1	1		1	1	1	1	1	X	1	X	X	0	X	0
0	1	0	1		1	0	0	0	1	X	X	1	0	X	X	1
0	1	1	0		1	0	0	1	1	X	X	1	X	1	1	X
1	0	0	0		0	0	1	0	X	1	0	X	1	X	0	X
1	0	0	1		0	0	0	1	X	1	0	X	0	X	X	0
1	1	0	0		0	1	1	0	X	1	X	0	1	X	0	X
1	1	1	1		0	1	0	1	X	1	X	0	X	1	X	0

J3(Q3,Q2,Q1,Q0)=∑3,5,6
K3(Q3,Q2,Q1,Q0)=∑8,9,12,15
J2(Q3,Q2,Q1,Q0)=∑3
K2(Q3,Q2,Q1,Q0)=∑5,6
J1(Q3,Q2,Q1,Q0)=∑1,8,12
K1(Q3,Q2,Q1,Q0)=∑2,6,15
J0(Q3,Q2,Q1,Q0)=∑0,8
K0(Q3,Q2,Q1,Q0)=∑5

4.Q: Design 8-3 priority Encoder?

An 8-bit priority encoder. This circuit basically converts a one-hot encoding into a binary representation. If input n is active, all lower inputs (n-1 .. 0) are ignored. Please read the description of the 4:2 encoder for an explanation.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	X	0	0	1
0	0	0	0	0	1	X	X	0	1	0
0	0	0	0	1	X	X	X	0	1	1
0	0	0	1	X	X	X	X	1	0	0
0	0	1	X	X	X	X	X	1	0	1
0	1	X	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	X	1	1	1

Where X equals “dont care”, that is logic “0” or a logic “1”.

From this truth table, the Boolean expression for the encoder above with data inputs $D_0$to D_7 and outputs Q_0, Q_1, Q_2 is given as:

Output Q_0

$$Q_0 = \sum(1, 3, 5, 7)$$

$$Q_0 = \sum\left(\bar{D}_7\bar{D}_6\bar{D}_5\bar{D}_4\bar{D}_3\bar{D}_2D_1 + \bar{D}_7\bar{D}_6\bar{D}_5\bar{D}_4D_3 + \bar{D}_7\bar{D}_6D_5 + D_7\right)$$

$$Q_0 = \sum\left(\bar{D}_6\bar{D}_4\bar{D}_2D_1 + \bar{D}_6\bar{D}_4D_3 + \bar{D}_6D_5 + D_7\right)$$

$$Q_0 = \sum\left(\bar{D}_6\left(\bar{D}_4\bar{D}_2D_1 + \bar{D}_4D_3 + D_5\right) + D_7\right)$$

Output Q_1

$$Q_1 = \sum(2, 3, 6, 7)$$

$$Q_1 = \sum\left(\bar{D}_7\bar{D}_6\bar{D}_5\bar{D}_4\bar{D}_3D_2 + \bar{D}_7\bar{D}_6\bar{D}_5\bar{D}_4D_3 + \bar{D}_7D_6 + D_7\right)$$

$$Q_1 = \sum\left(\bar{D}_5\bar{D}_4D_2 + \bar{D}_5\bar{D}_4D_3 + D_6 + D_7\right)$$

$$Q_1 = \sum\left(\bar{D}_5\bar{D}_4(D_2 + D_3) + D_6 + D_7\right)$$

Output Q_2

$$Q_2 = \sum(4, 5, 6, 7)$$

$$Q_2 = \sum\left(\bar{D}_7\bar{D}_6\bar{D}_5D_4 + \bar{D}_7\bar{D}_6D_5 + \bar{D}_7D_6 + D_7\right)$$

$$Q_2 = \sum(D_4 + D_5 + D_6 + D_7)$$

Then the final Boolean expression for the priority encoder including the zero inputs is defined as:

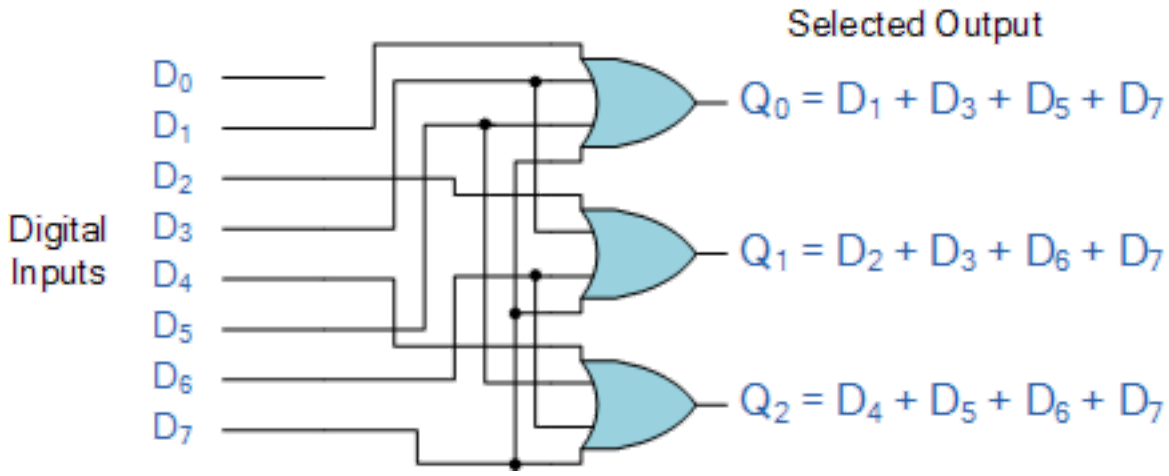
$$Q_0 = \sum\left(\bar{D}_6\left(\bar{D}_4\bar{D}_2D_1 + \bar{D}_4D_3 + D_5\right) + D_7\right)$$

$$Q_1 = \sum\left(\bar{D}_5\bar{D}_4(D_2 + D_3) + D_6 + D_7\right)$$

$$Q_2 = \sum(D_4 + D_5 + D_6 + D_7)$$

In practice these zero inputs would be ignored allowing the implementation of the final Boolean expression for the outputs of the 8-to-3 priority encoder. We can constructed a simple encoder from the expression above using individual OR gates as follows.

Digital Encoder using Logic Gates



5.Q: **Implement using 8-1 MUX F(A,B,C,D)=ABD+ACD+BCD**

Ans: Here we first need to create a truth table and find the boolean functions

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	7=BCD
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	11=ACD
1	1	0	0	0
1	1	0	1	13=ABD
1	1	1	0	0
1	1	1	1	0

So the boolan function is (7,11,13)

	I0	I1	I2	I3
A'B'	0	1	2	3
A'B	4	5	6	7
AB'	8	9	10	11
AB	12	13	14	15

AB A'B+AB'=A XOR B