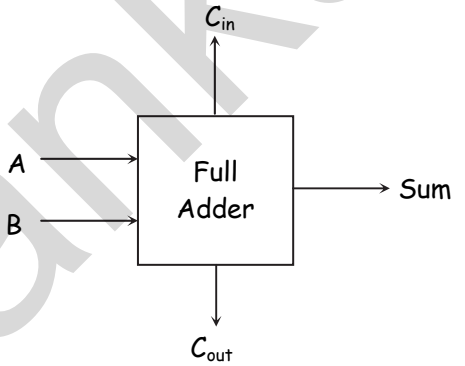
Q.1(a) Draw truth table and logic diagram of full adder. [4]

(A) Full Adder

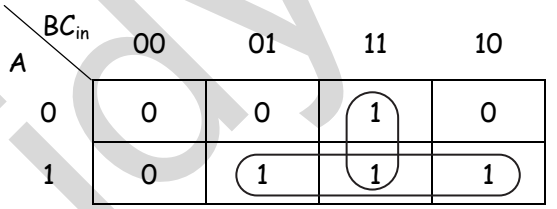
It is combinational circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs. Two of the input variables denoted by A and B represent the two significant bits to be added. The third input C_{in}, represents the carry from the previous lower significant position.

Inputs			Outputs	
A	B	C_{in}	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



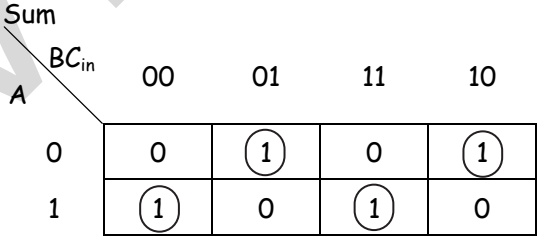
(Truth table for full–adder)

K–map simplification for carry and sum carry (C_{out})



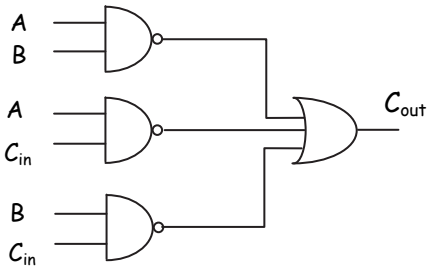
$C_{out} = AB + AC_{in} + BC_{in}$

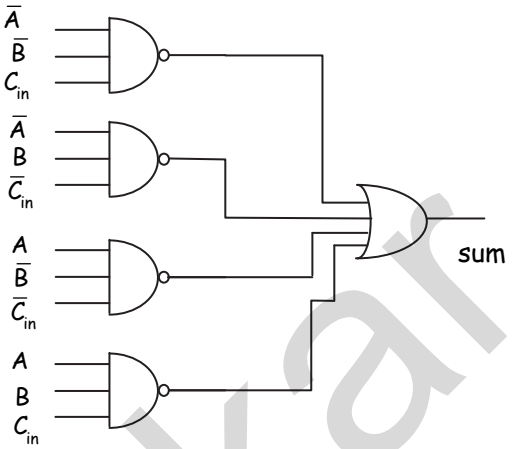
Sum



$Sum = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$

Logic Diagram





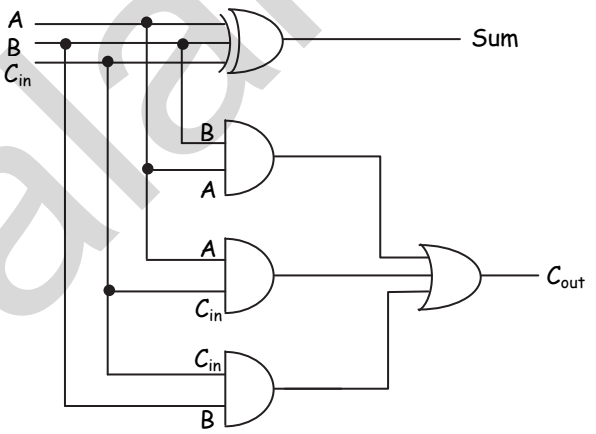
The Boolean function for sum can be simplified as follows:–

$$\text{Sum} = \overline{A}\overline{B}C_{in} + \overline{A}B\overline{C}_{in} + A\overline{B}\overline{C}_{in} + ABC_{in}$$

$$= C_{in}\left(\overline{A}\overline{B} + AB\right) + \overline{C}_{in}\left(\overline{A}B + A\overline{B}\right)$$

$$= C_{in}\left(\overline{A \oplus B}\right) + \overline{C}_{in}(A \oplus B)$$

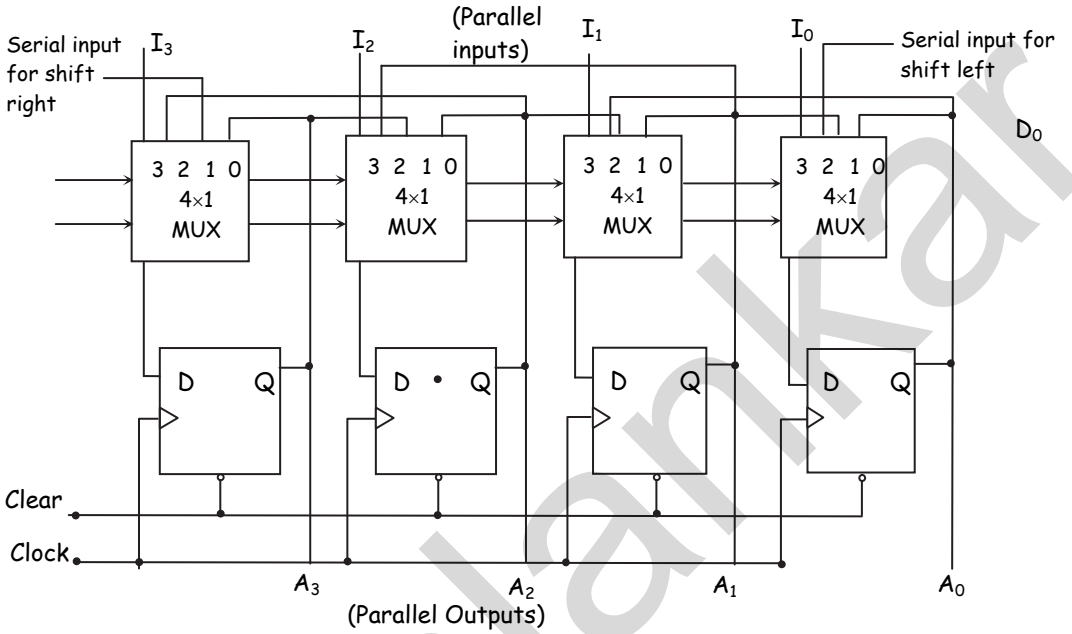
$$= C_{in} \oplus \left(A \oplus B\right)$$



Q.1(b) Explain universal shift register. **[4]**

(A) A register capable of shifting in one direction only is a unidirectional shift register. A register capable of shifting in both directions is bidirectional shift register. If the register has both shifts (right–shifts and left–shift) and parallel load capabilities, it is referred to as universal shift register. It consist of four flip–flops and four multiplexers. The four multiplexers have two common selection inputs S_1 and S_0 and they select appropriate input for D flip–flop. The register operation depends on selection inputs of multiplexers. When S_1S_0 = 00, input 0 is selected and present value of register is applied to D inputs of flip–flops. This results in no change in register value. When S_1 S_0 = 01, input 1 is selected and circuit connections

are such that it operates as right shift register. When S_1 S_0 = 10, input 2 is selected and circuit connections are such that it operates as left–shift register. When S_1S_0 = 11, binary information on parallel input lines is transferred to register simultaneously and it is a parallel load operation.



Mode control		Register operation
S_1	S_0	
0	0	No change
0	1	Shift-right
1	0	Shift-left
1	1	Parallel load

Q.1(c) Compare synchronous and asynchronous counters. [4]

(A)

	Synchronous	**Asynchronous counters**
1.	There is no connection between output of first flip–flop and clock input of next flip–flop	Counter flip–flops are connected in such a way that output of first flip–flop drives the clock for next flip–flop
2.	All flip flops are clocked simultaneously	All flip–flops are not clocked simultaneously.
3.	Design involves complex logic circuit as number of states increases	Logic circuit is very simple even for more number of states

4.	As clock is simultaneously given to all flip–flops, there is no problem of propagation delay. Hence, they are high speed counter and are preferred when number of flip–flops increase in given design	Main drawback of these counters is low speed as clock is propagated through number of flip-flops before it reaches last flip–flop

Q.1(d) Explain Stuck at 0 and 1 faults. [4]

(A) Stuck at '0' and '1' faults

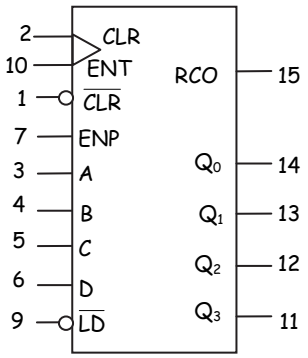
They occur when a line is permanently stuck to V_{DD} or ground giving a faculty output. This line maybe an input or output to any gate. Also this fault can be single or multiple stuck at faults. When a signal or gate output is stuck at a 0 or 1 value, independent of the inputs to the circuit, the signal is said to be 'stuck at' and fault model used to describe this error is called 'stuck at fault model'. A fault model is an engineering model of something that could go wrong in the construction or operation of a piece of equipment. From the model, the designer or user can predict the consequences of this particular fault. To use the stuck–at fault model, each input pin on each gate in turn, is assumed to be grounded and a test vector is developed to indicate that circuit is faulty. If a circuit has 'n' signal lines, there are potentially '2n' stuck–at faults defined on the circuit. the test vector is a collection of bits to apply to circuit's inputs and a collection of bits expected at circuits output. If the gate pin under consideration is grounded, and this test vector is applied to the circuit, at least one of the output bits will not agree with corresponding output bit in test vector. After obtaining the test vectors for grounded pins, each pin is connected in turn to a logic one and another set of test vectors is used to find faults occurring under these conditions. Each of these faults is called a single struck–at '0' or single stuck–at 1 fault, respectively. It is a logical fault model because no delay information associated with fault definition. It is structural because it is defined based on structural gate level circuit model.

Q.1(e) What are the advantages and disadvantages of Quine McCluskey method [4]

(A) Quine McCluskey method of simplification can be applied to any number of variables and it is algorithmic nature. The simplification process of Qunie McCluskey method becomes lengthy and time consuming as number of variables increases. However, the algorithm can be easily programmed to run on a computer to identify prime implicants and implicates of any function.

Q.2(a) Explain IC 74163 with diagram. **[8]**

(A) IC 74163 is a synchronous 4-bit binary counter with active-low load and clear inputs. The internal logic diagram of IC 74163 is as shown in the following :



Inputs				Current State				Next State			
$\overline{CLR}$	$\overline{LD}$	ENT	ENP	Q_3	Q_2	Q_1	Q_0	Q_3^+	Q_2^+	Q_1^+	Q_0^+
0	x	x	x	x	x	x	x	0	0	0	0
1	0	x	x	x	x	x	x	D	C	B	A
1	1	0	x	x	x	x	x	Q_3	Q_2	Q_1	Q_0
1	1	x	0	x	x	x	x	Q_3	Q_2	Q_1	Q_0
1	1	1	1	0	0	0	0	0	0	0	1
1	1	1	1	0	0	0	1	0	0	1	0
1	1	1	1	0	0	1	0	0	0	1	1
1	1	1	1	0	0	1	1	0	1	0	0
1	1	1	1	0	1	0	0	0	1	0	1
1	1	1	1	0	1	0	1	0	1	1	0
1	1	1	1	0	1	1	0	0	1	1	1
1	1	1	1	0	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0	1	0	0	1
1	1	1	1	1	0	0	1	1	0	1	0
1	1	1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	1	1	1	1	0	0
1	1	1	1	1	1	0	0	1	1	0	1
1	1	1	1	1	1	0	1	1	1	1	0
1	1	1	1	1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0

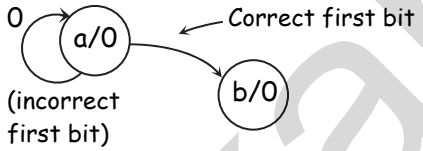
The IC74163 uses D flip-flops to perform load and clear functions. Each D input is driven by a 2-input multiplexer form by the combination of an OR gate and two AND gates. The multiplexer output is 0 if the $\overline{CLR}$ input is asserted. Otherwise, the top AND gate passes the data input (A, B, C or D) to the output if $\overline{LD}$ is asserted. If neither $\overline{CLR}$ nor $\overline{LD}$ is asserted, the bottom AND gate passes the output of XNOR gate to the multiplexer output. One input of XNOR gate corresponds of one count bit either Q_A, Q_B, Q_C or Q_D. The XNOR gate gives complement output if and only if both enable ENP and ENT are asserted and all of the lower – order count bits are 1. The ripple carry out (RCO) bit is 1 if all of the count bits are 1 and ENT is asserted. The RCO signal indicates a carry from the most significant bit position.

Q.2(b) Design Moore type sequence detector to detect a serial input sequence of 101. **[12]**

(A) In Moore type design, the output depends only on flip-flop states. Hence, in the state diagram of Moore machine, the output is written with state instead of with transition between the states. Let us start with state 'a' as initial state.

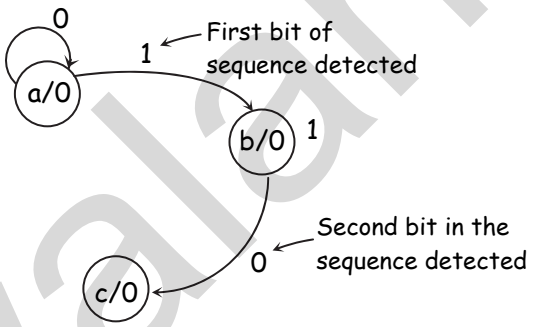
(i) State 'a'

When input is 1, we have detected the first bit in the sequence, hence we have to go to the next state to detect next bit in the sequence.



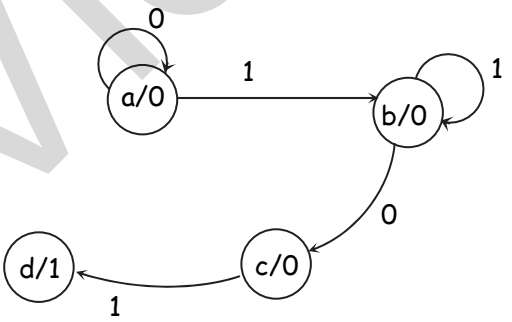
When input is '0', we have to remain in state 'a' because bit '0' is not the first bit in the sequence. In both cases, the output is '0'. Since we have not yet detected all the bits in the sequence.

(ii) State 'b'

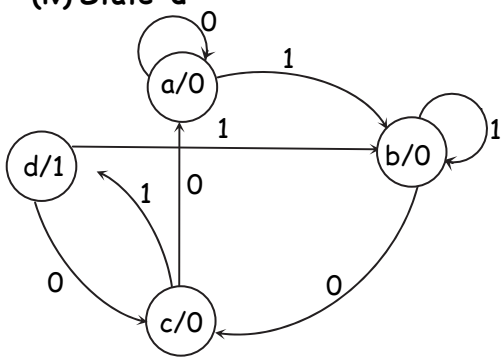


When input is '0', we have detected the second bit in the sequence, hence we have to go to the next state to detect next bit in the sequence. When input is 1, we have to remain in state 'b' because '1' which we have detected may start the sequence output is still zero.

(iii) State 'c'



(iv) State 'd'



The state table can be determined as follows:–

Present Table	Next State		Output z
	x = 0	x = 1	
a	a	b	0
b	c	b	0
c	a	d	0
d	c	b	1

Since there are four states we need two flip–flops. Assigning state a = 00, b= 01, c = 10, d = 11, we can determine excitation table as follows

Present Table		Next State $A^+ B^+$				Output z
A	B	x = 0		x = 1		
0	0	0	0	0	1	0
0	1	1	0	0	1	0
1	0	0	0	1	1	0
1	1	1	0	0	1	1

K-map simplification

AB \ x	0	1
00	0	0
01	(1)	0
11	0	0
10	(1)	0

$$A^+ = \bar{A}B\bar{X} + A\bar{B}\bar{X}$$
$$= \bar{X}\left(\bar{A}B + A\bar{B}\right)$$
$$= \bar{X}\left(A \oplus B\right)$$

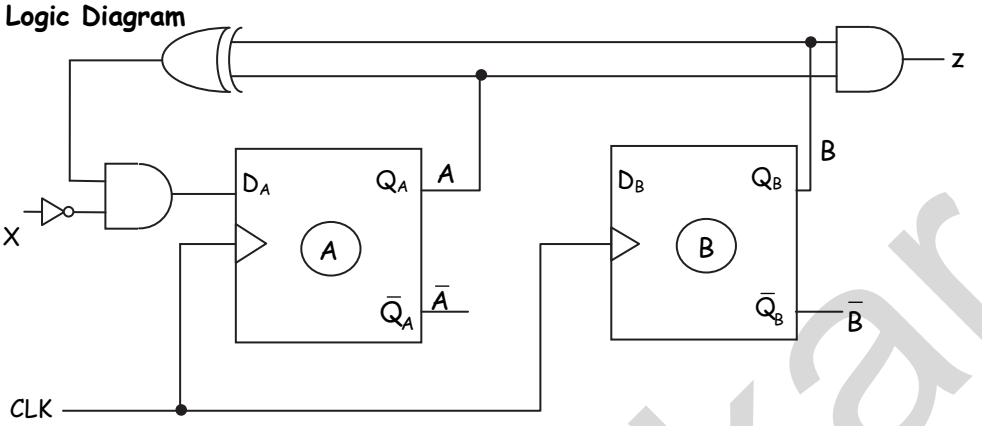
AB \ x	0	1
00	0	1
01	0	1
11	0	1
10	0	1

$B^+ = x$

A \ B	0	1
0	0	0
1	0	(1)

$z = AB$

Logic Diagram



Q.3(a) Design MOD-5 synchronous counter using T flip-flop. [10]

(A) To design MOD–5 synchronous counter using T flip-flop. For MOD–5 counter, we require 3 flip-flops

Excitation Table:

Present State			Next State			Flip-Flop inputs		
Q_A	Q_B	Q_C	Q_{A+1}	Q_{B+1}	Q_{C+1}	T_A	T_B	T_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	0	0	0	1	0	0

K–map simplification

Q_A \ Q_BQ_C	00	01	11	10
0	0	0	1	0
1	1	×	×	×

$T_A = Q_A + Q_B Q_C$

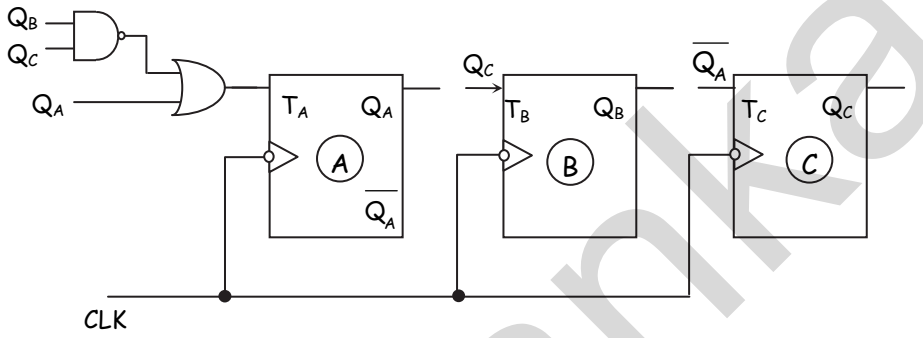
Q_A \ Q_BQ_C	00	01	11	10
0	0	1	1	0
1	0	×	×	×

$T_B = Q_C$

Q_A \ Q_BQ_C	00	01	11	10
0	1	1	1	1
1	0	×	×	×

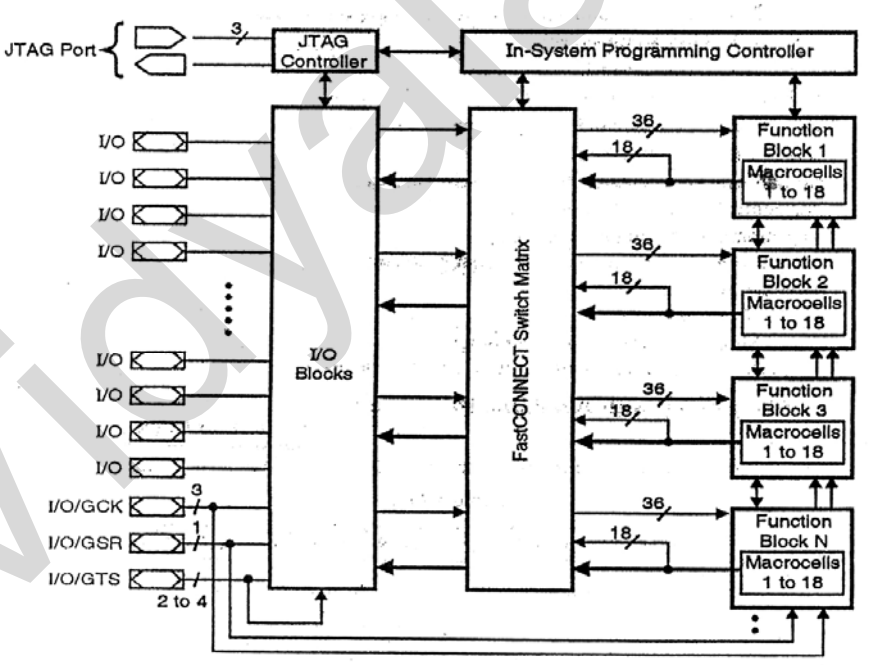
$T_C = \bar{Q}_A$

Logic Diagram



Q.3(b) Explain Xilinx XC 9500 CPLD architecture. [10]

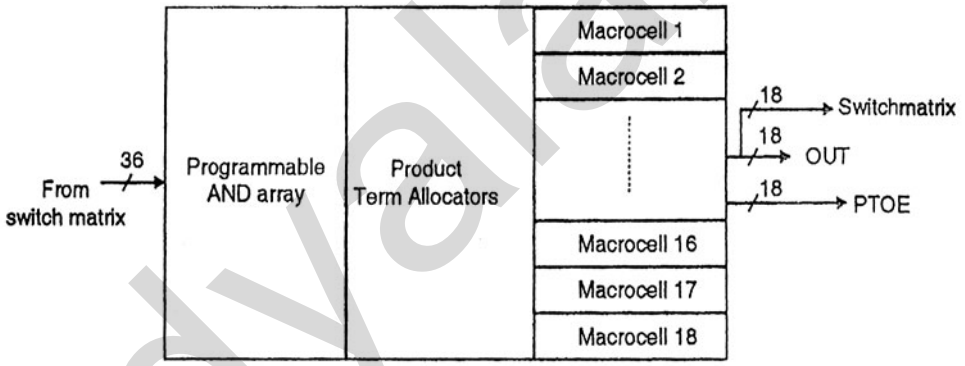
(A)



XC9500 architecture

An External I/O pin of Xilinx XC9500 CPLD can be used as an input, an output or a bidirectional pin according to device programming. The I/O pins

at the bottom are also used for special purposes. Any of the 3 pins can be used as 'Global Clocks' (GCK). Each Macrocell can be programmed to use a selected clock input. One pin can be used as 'Global set/Reset' GSR. Each Macrocell can use this signal as an asynchronous Preset or Clear. Two or four pins depending on the devices can be used a 'Global Three State Controls' GTS. One of the signals can be selected in each macrocell to output enable the corresponding output driver when macrocell output is hooked to an external I/O pin. Only four functional blocks are shown but XC9500 scales to accommodate 16FB's in XC95288. Regardless of the specific family member, each FB programmable receives 36 signals from switch matrix. The inputs to the switch matrix are the 18 macrocell outputs from each of the functional blocks and external inputs from I/O pins. Each functional blocks also has 18 outputs that run under the switch matrix and connect to the I/O block output drives, they are used when FB macrocell's output is hooked upto an external I/O pin. Each functional block has programmable logic capability with 36 inputs and 18 outputs. Fast connect switch matrix connects all functional block outputs to I/O blocks and inputs signals from I/O blocks to the functional block.



The Functional block consist of 18 independent macrocells capable of implementing combinational or registered function. Functional block also receives global clock, output enable, set/reset signals. It generates 18 outputs that drive FastCONNECT switch matrix. Products term allocator allocates any number of these 90 product terms to each macrocell. Each function block supports local feedback paths that allows any number of functional output to drive its own programmable AND array without going outside the functional block.

Q.4(a) Design a 2 bit comparator using gates. **[12]**

(A) To design 2–bit comparator using gates. Truth table is as follows :

Inputs				Outputs		
A_1	A_0	B_1	B_0	A > B	A = B	A < B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

K-map Simplification (A > B)

A_1A_0 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

$$A>B = A0\bar{B}_1\bar{B}_0 + A_1\bar{B}_1 + A_1A_0\bar{B}_0$$

(A=B)

A_1A_0 \ B_1B_0	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
0	0	0	0	1

$$(A=B)= \bar{A}_1\bar{A}_0\bar{B}_1\bar{B}_0 + \bar{A}_1 A_0 \bar{B}_1 B_0 + A_1 A_0 B_1 B_0 + A_1 \bar{A}_0 B_1 \bar{B}_0$$

$$= \bar{A}_1\bar{B}_1\left(\bar{A}_0\bar{B}_0 + A_0 B_0\right) + A_1 B_1\left(A_0 B_0 + \bar{A}_0\bar{B}_0\right)$$

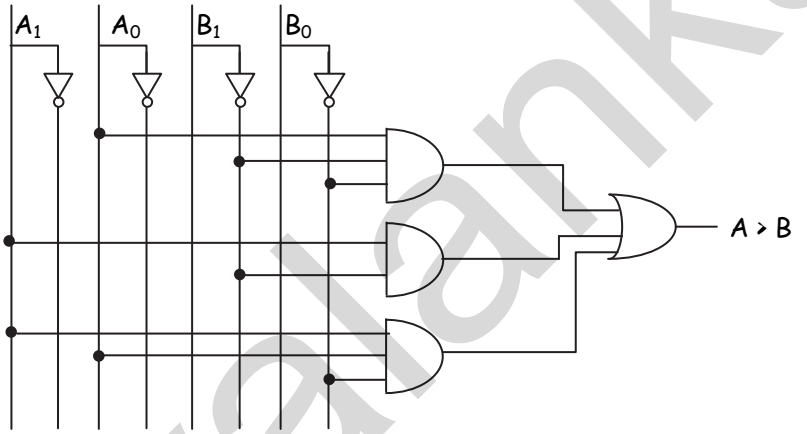
$$= \left(A_0 \odot B_0\right)\left(A_1 \odot B_1\right)$$

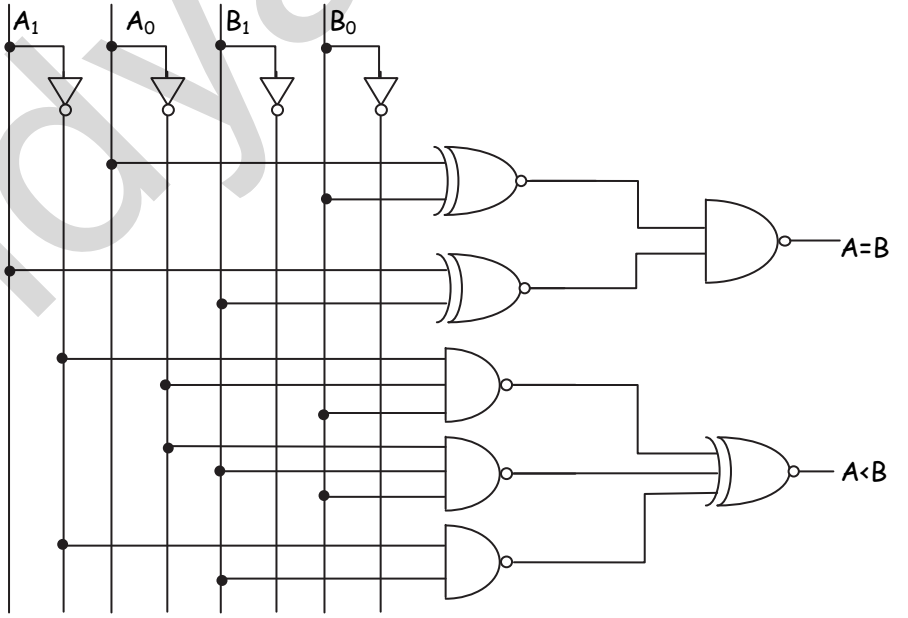
(A<B)

A_1A_0 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

$$(A<B) = \bar{A}_1 \bar{A}_0 B_0 + \bar{A}_0 B_1 B_0 + \bar{A}_1 B_1$$

Logic diagram





Q.4(b) Explain JTAG and BIST. [8]

(A) JTAG

The Joint Test Action Group (JTAG) is an electronics industry association formed for developing a method of verifying designs and testing printed circuits boards after manufacture. The Boundary-Scan Method (JTAG Boundary Scan) is a method of testing modern Printed Circuit Boards after assembly. Using the dedicated test logic built into many of today's integrated circuits, boundary scan checks if each device is correctly inserted. A JTAG interface is a special interface added to a chip. Depending on the version of JTAG, two, four or five pins are added. A test probe only connects to a single 'JTAG port' to have access to all chips on a circuit board. To simplify the test architecture with a PCB it is common to connect the devices in a serial (daisy chain) formation so that the first device's TDO connects to the next device's TDI (and so on) to form a so–called scan chain. JTAG boundary scan technology provides access to many logic signals of a complex integrated circuit, including device pins. The signals are represented in the boundary scan register (BSR) accessible via the TAP (Test access port). This permit testing as well as controlling the states of the signals for testing and debugging. Hence, both software and hardware faults may be located and an operating device may be monitored. When combined with build-in self- test (BIST), the JTAG scan chain enables a low overhead, embedded solution to testing an IC for certain static faults (shorts, opens and logic errors).

BIST:

Modern day ICs based on deep sub–micro technology may develop failures after initial fault checking and even during operation within expected life time. to cater to this problem sometimes redundant circuitry are kept on – chip which replace the faulty parts. To enable replacement of faulty circuitry, the ICs are tested before each time they startup. If a fault is found, a part of the circuit (having the fault) is replaced with a corresponding redundant circuit part (by re–adjusting corrections). Testing a circuit every time before they startup, is called the Built–in Self-test (BIST). BIST is basically same as off–line testing using ATE where the test pattern generator and test response analyzer are on-chip circuitry (instead of equipment). As equipment are replaced by circuitry, So it is obvious that compressed implementations of test pattern generator and response analyzer are to be designed. BIST is designed to meet requirements such as high reliability and low repair cycle times.

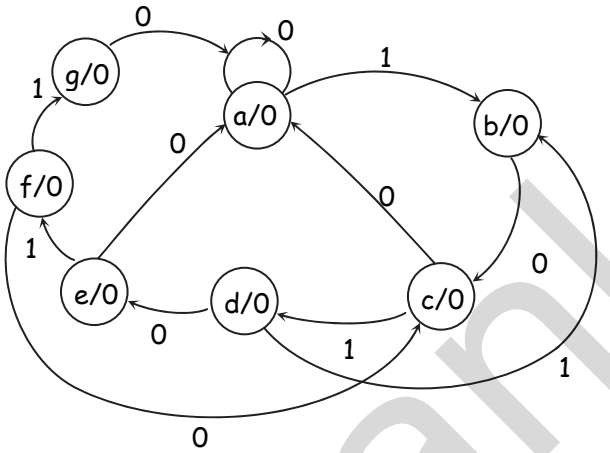
Q.5(a) Design a sequence detector to detect the sequence of 101011. [10]

(A) To design sequence detector to detect sequence 101011. State diagram is as follows :–

State assignment

a = 000 b = 001 c = 010 d = 011

e = 100 f = 101 g = 110



Excitation Table :

Present state			Input	Next state			Output
A	B	C	X	A+	B+	C+	Y
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0
0	0	1	1	0	0	1	0
0	1	0	0	0	0	0	0
0	1	0	1	0	1	1	0
0	1	1	0	1	0	0	0
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	0
1	0	0	1	1	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	1	1	0	0
1	1	0	0	0	0	0	1
1	1	0	1	0	0	1	1
1	1	1	0	×	×	×	×
1	1	1	1	×	×	×	×

K-map simplification

AB \ CX	00	01	11	10
00	0	0	0	0
01	0	0	0	1
11	0	0	×	×
10	0	1	1	0

$A+ = A\bar{B}X + BC\bar{X}$

AB \ CX	00	01	11	10
00	0	0	0	1
01	0	1	0	0
11	0	0	×	×
10	0	0	1	1

$B+ = \bar{A}B\bar{C}X + AC + \bar{B}C\bar{X}$

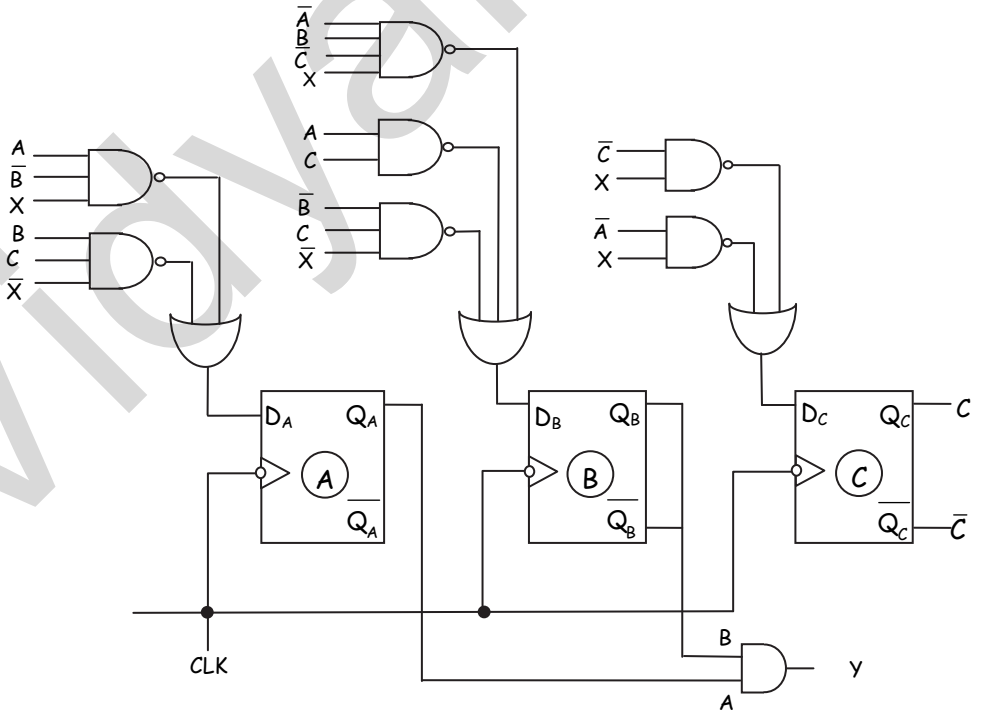
AB \ CX	00	01	11	10
00	0	1	1	0
01	0	1	1	1
11	0	1	×	×
10	0	1	0	0

$C+ = \bar{C}X + \bar{A}X$

AB \ CX	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	×	×
10	0	0	0	0

$Y = AB$

Logic Diagram



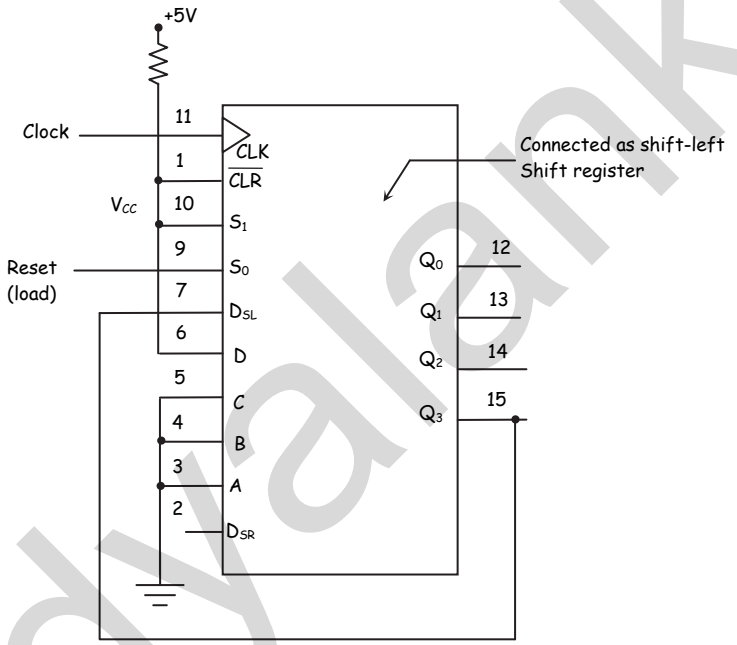
Q.5(b) Compare Moore and Mealy models. **[10]**

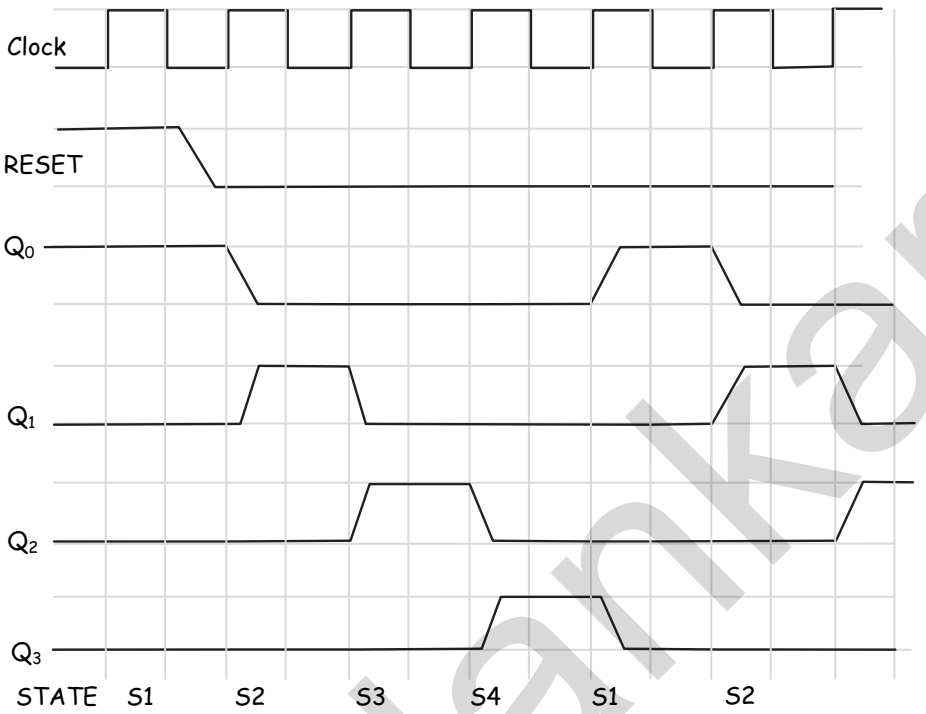
(A)

	Moore Model	Mealy Model
1	Its output is a function of present state only	Output is a function of present stat as well as present input
2	Input changes do not affect the output	Input changes may affect output of the circuit
3	It requires more number of states for implementing same function	It requires less number of states for implementing same function

Q.6(a) Design a 4 bit, 4 state ring counter using IC 74194. **[10]**

(A)





(Timing diagram for 4 bit ring counter)

The ring counter with a single circulating 1 is the IC74X194 universal shift register is connected so that it normally performs a left shift. However, when RESET is asserted it loads 0001. Once RESET is negated, the 74194 shifts left on each clock pulse. The D_{SL} serial input is connected to the leftmost output (Q_3: MSB) so the next states are 0010, 0100, 1000, 0001, 0010,….Thus, the counter visits four unique state before reporting

Q.6(b) Write VHDL codes for the following : [10]

(i) SR flip flop (ii) 4 bit shift register

(A) (i) SR flip flop

```
Library ieee;
use ieee.std_Logic_1164.all;
use ieee.std_Logic_arith.all;
use ieee.std_Logic_unsigned.all;
entity  SR_FF is
PORT (S, R, CLOCK: in std_logic;
Q, QB: out std_logic);
end SR_FF;
Architecture behavioral of SR_FF is
begin
```

```
PROCESS (CLOCK)
Variable tmp; std_Logic;
begin
if(CLOCK = '1' and CLOCK' EVENT) then
if(S = '0' and R = '0') then
tmp: = tmp;
elsif(s = '1' and R = '1') then
tmp:= 'z';
elsif(s= '0' and R= '1') then
tmp: = '0' ;
else
tmp: = ' l ';
end if;
end if;
Q < = tmp;
QB < = not tmp;
end PROCESS;
end behavioral;
```

(ii) 4 bit shift register

(1) Parallel in Parallel out shift register

(2) Serial in – parallel out shift register

(1) Parallel in – Parallel out shift register (VHDL code)

```
library ieee;
use ieee. std_logic_1164.all;
entity pipo is
port (clk; in std_logic; D: in std_logic_vector Q; out
std_logic_vector  (3 down to 0) (3 down to 0);
);
end pipo;
architecture arch of pipo is
begin
process (clk)
begin
if (clk' event and clk = ' l ') then
Q < = D;
end if ;
end process;
end arch;
```

(2) Serial in – serial out shift register (VHDL code)

```
library ieee;
use ieee. std_logic_1164.all;
entity sipo is
port (clk, clear; in std_logic; input_data : in std_logic;
Q; out std_logic_vector (3 dowon to 0))
end sipo;
architecture arch of sipo is
begin
process (clk)
begin
if clear = ' l ' then
Q < = "0 0 0 0"
else if(clk' event and CLK = '1') then
Q(3 down to 1) < = Q (2 downto 0);
Q(0) < = Input_Data;
end if;
end process;
end arch;
```

❑ ❑ ❑ ❑ ❑