MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2005 Certified)

Winter – 15 EXAMINATIONS

Subject Code: **17659** **<u>Model Answer</u>**

Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the answer scheme.

2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.

3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.

4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.

5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate’s answers and model answer.

6) In case of some questions credit may be given by judgment on part of examiner of relevant answer based on candidate’s understanding.

7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q.1 a) Attempt any THREE of the following 12 Marks

(i) Define the following terms.

1) Metastability 2) Noise Margin

Ans: **(any relevant correct definition – 2 marks each)**

Metastability:

Metastability in electronics is the ability of a non- equilibrium electronic state to persist for a long (and theoretically unboundable) period of time.

(OR)

A metastable state is half way between logic '0' and logic '1'. It is undefined state.

Noise Margin:

It is a measure of noise immunity of a gate or circuit (noise immunity is the ability of a gate or circuit to tolerate any noise present in a signal without performing a wrong operation).

(ii) Compare BJT and CMOS

Ans: **(any four correct points – 1 mark each)**

Sr. No.	BJT	CMOS
1.	High power dissipation	Low static power dissipation
2.	Low input impedance	High input impedance
3.	Low packing density	High packing density
4.	Low delay sensitive to load	High delay sensitive to load
5.	High output drive current	Low output drive current
6.	Essentially unidirectional	Bidirectional capability
7.	It is not an ideal switching device	It is an ideal switching device
8.	Current driven	Voltage driven

(iii) What is VHDL? State VHDL flow elements.

Ans: **(VHDL – 1 mark, any three flow elements – 1 mark each)**

VHDL:

It is a hardware description language used for modelling digital system mode of interconnection of components.

VHDL Flow elements:

- VHDL provides different types of primary constructs called design units or flow elements which are as follows:

1. Entity : It specifies the name of the entity, the ports of the entity, and other entity related information
2. Architecture: It specifies behaviour, function, interconnection or relation between input and output of an entity.
3. Configuration: The configuration describes the behaviour of each entity much like port list describing which port is to be used in which port of design.

(iv) Explain flattening and structuring with example.

Ans: **(each correct explanation – 2 marks)**

Flattening:

- The process of converting the unoptimized boolean description to a pla format is known ad flattening, because it creates a flat signal representation of only two levels: an AND level and an OR level.
- The idea is to get the unoptimized Boolean description into a format in which optimization algorithms can be used to optimize the logic.
- A pla structure is a very easy description in which to perform boolean optimization, because it has a simple structure and the algorithms are well known. An example of a boolean description is shown here:
- Original equations

A = b and c;

B = x or (y and z);

C = q or w;

- This description shows an output that has three equations describing its function. These equations use two intermediate variables b and c to hold temporary values which are then used to calculate the final value for a.
- These equations describe a particular structure of the design that contains two intermediate nodes or signals b and c.
- The flattening process removes these intermediate nodes to produce a completely flat design with no intermediate nodes.

Structuring:

- Structuring is the process of adding intermediate terms to add structure to a description.
- Structuring is usually desirable as flattened designs are bigger and slower because of the amount of fan-outs generated.
- Example if [(A AND B) OR C] occurs ten times, then the tool may assign it a variable ‘X’ and then ‘X’ is used everywhere.
- Finally the sub functions are substituted into the original equations. Comparing to the logic before structuring, the resulting area is reduced.

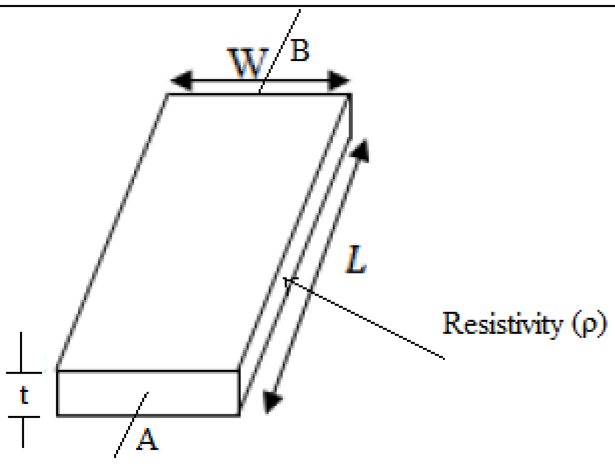
(v) Explain the estimation of resistance of channel for MOSFET (sheet resistance).

Ans: **(correct explanation with sketch – 4 marks)**

A MOS is created by superimposing a number of layers of conducting and insulating materials. It has diffusion polysilicon and metal layers separated by insulating layers. Due to these layers resistances and capacitances are introduced in the circuit, which affects the performance of the circuit. They also have inductance characteristics.

Resistance Estimation:

Consider a uniform slab of conducting material of resistivity (ρ). Let (W) be the width, (t) the thickness and *l* the length of the slab.



$$R_{AB} = R_s \cdot \frac{L}{W}\ \Omega$$

Hence, the resistance between A and B terminal is found as,

RAB =ρ.L/A ohms.

Where A = cross-sectional area.

Thus RAB =ρ.L/t.W ohms.

Consider the case in which L = W, that is a square of resistive material then

RAB =ρ/t = Rs

Where

Rs =ρ/t ohm per square or sheet resistance

Therefore, Rs = ohm per square

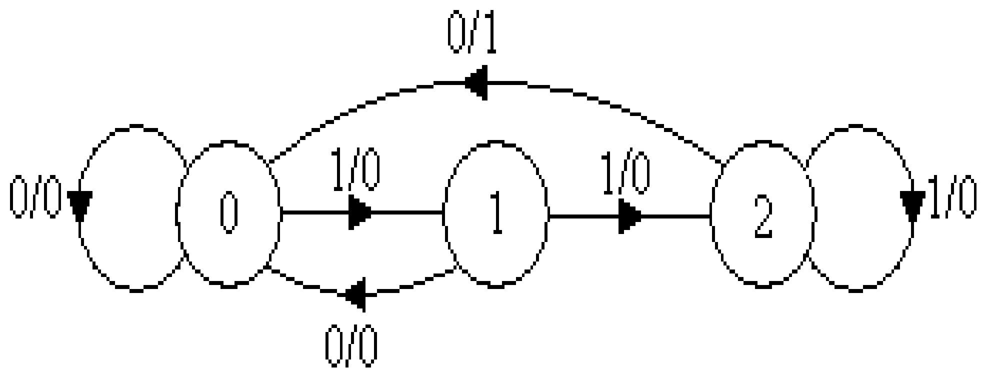
Hence Rs is completely independent of the area of the square.

Q 1. b) Attempt any ONE of the following 6 marks

(i) Design Mealy sequence detector circuit for detecting sequence of '110'using D flip-flop.

Ans: **(state diagram – 1 mark, table – 1 mark, K – maps – 2 marks, final design – 2 marks)**

The state transition diagram for the sequence 110 is as follows



(1 mark)

The state transition table for the above transition diagram is

Present State			Next State		
n			n+1		
D_{in}	A	B	A	B	D_{out}
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	1
0	1	1	x	x	x
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	x	x	x

(1 mark)

The K – maps for the above table to determine the equations are as follows

AB

	A	00	01	11	10
D_{in}	0	0	0	x	0
	1	0	1	x	1

$$D_A = D_{in}(A + B)$$

AB

	B	00	01	11	10
D_{in}	0	0	0	x	0
	1	1	0	x	0

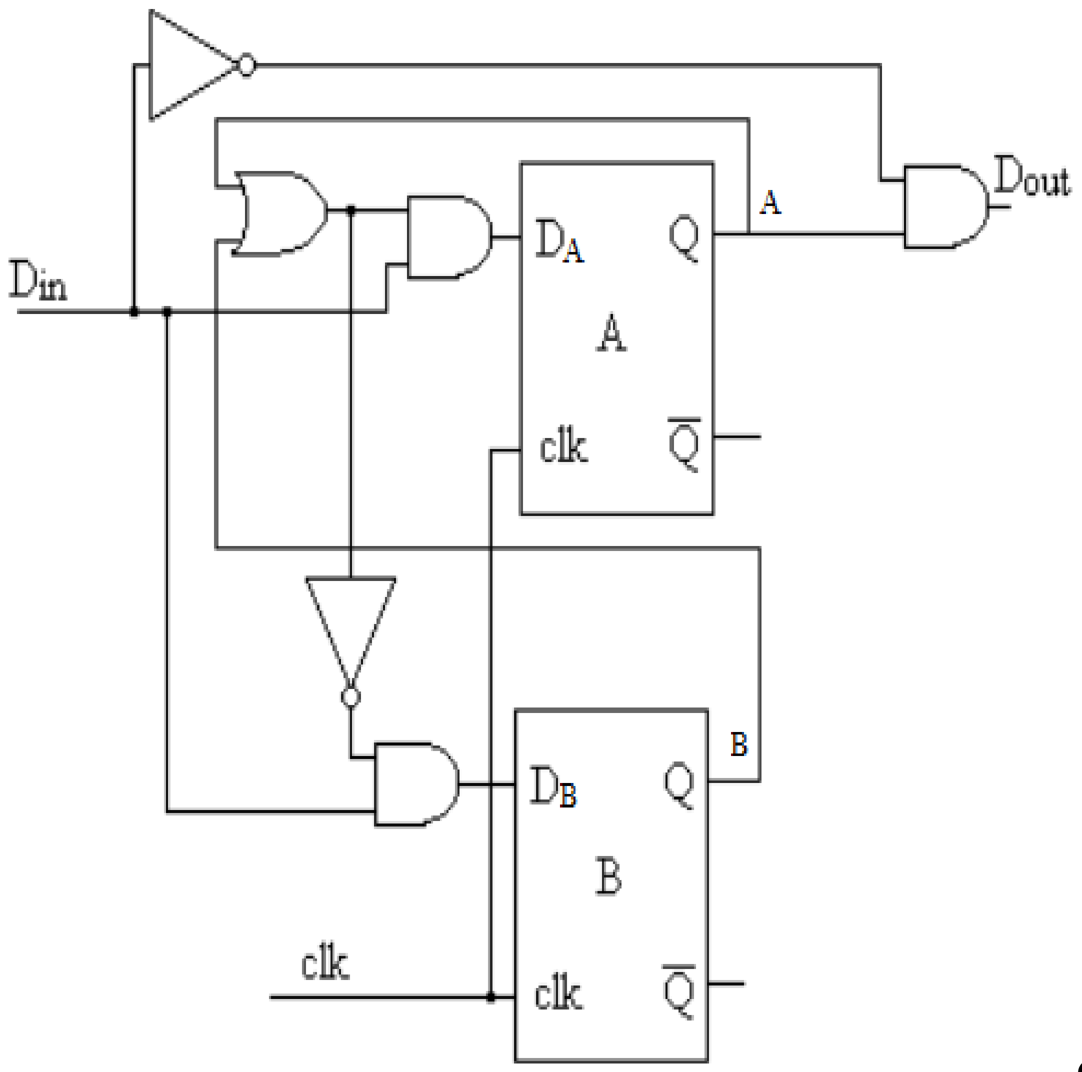
$$D_B = D_{in} \cdot \overline{A} \cdot \overline{B}$$

		AB			
	Dout	00	01	11	10
Din	0	0	0	x	1
	1	0	0	x	0

$$D_{out} = \overline{D_{in}} A$$

(2 marks)

Final implementation of Sequence detector 110 is as follows



(2 marks)

(ii) Write the VHDL program to implement JK flip-flop with positive edge trigger.

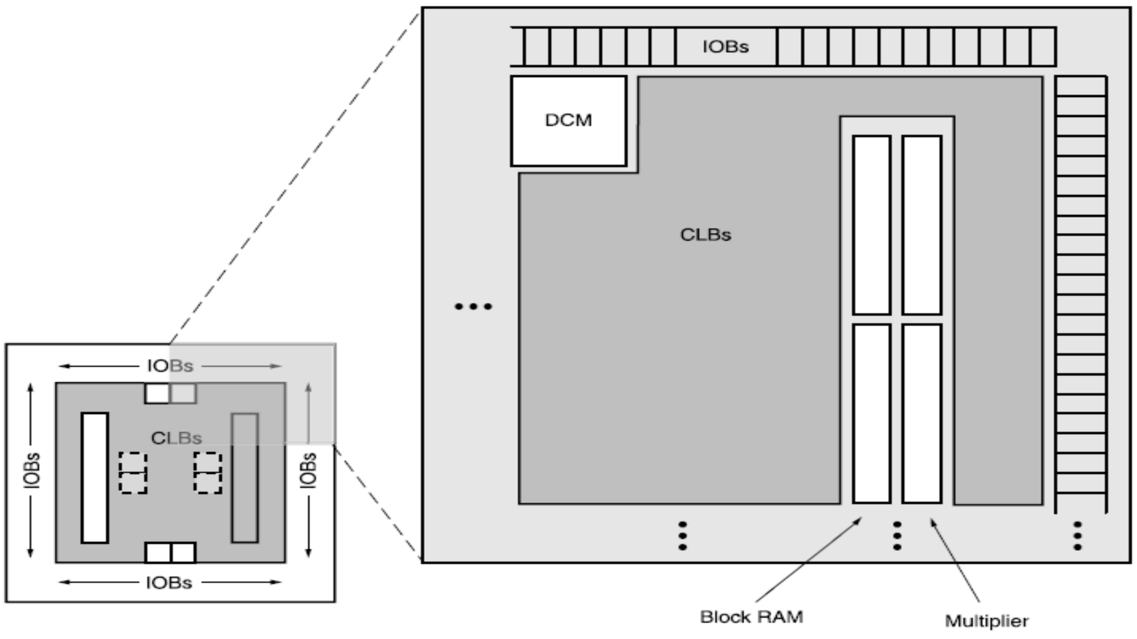
Ans: **(entity – 2 marks, architecture – 4 marks)**

Note: Program in any modelling should be considered and marks to be given

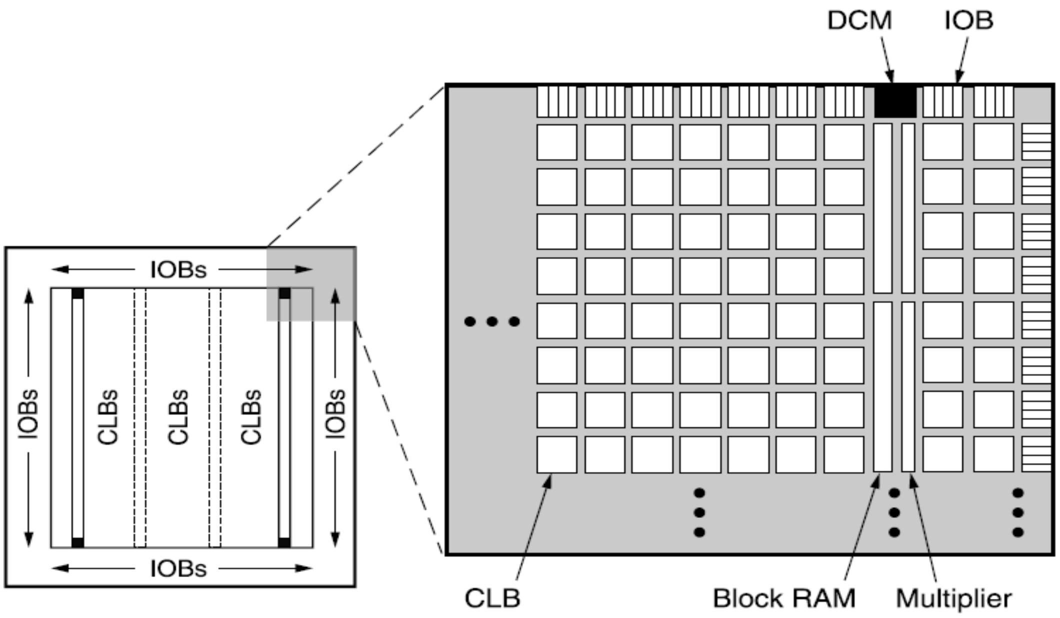
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY JKff IS
PORT (clk: IN STD_LOGIC;
          J, K: IN STD_LOGIC;
           Q, Qbar: OUT STD_LOGIC);
END JKff;
ARCHITECTURE BEHAVIORAL OF JKff_arch IS
BEGIN
     PROCESS (clk)
         BEGIN
           if (clk'event and clk = '1') then
           if (J = '0' and K = '0') then
              Q < = Q;
           elsif (J = '0' and K = '1') then
              Q < = '0';
           elsif (J = '1' and K = '0') then
              Q < = '1';
           elsif (J = '1'and K = '1') then
              Q < = not Q;
           End if
      End if
End process
Qbar < = not Q;
End behavioural JKff_arch
```

(iii) Explain the basic architecture of SPARTAN 3 FPGA series.

Ans: **(diagram – 4 marks, explanation – 2 marks)**



(OR)



The Spartan-3 family architecture consists of five fundamental programmable functional elements:

Configurable Logic Blocks (CLBs): Contain flexible Look-Up Tables (LUTs) that implement logic plus storage elements used as flip-flops or latches. CLBs perform a wide variety of logical functions as well as store data.

Input/ Output Blocks (IOBs): Control the flow of data between the I/O pins and the internal logic of the device. Each IOB supports bidirectional data flow plus 3-state operation. Double Data-Rate (DDR) registers are included.

Block RAM : Provides data storage in the form of 18-Kbit dual-port blocks.

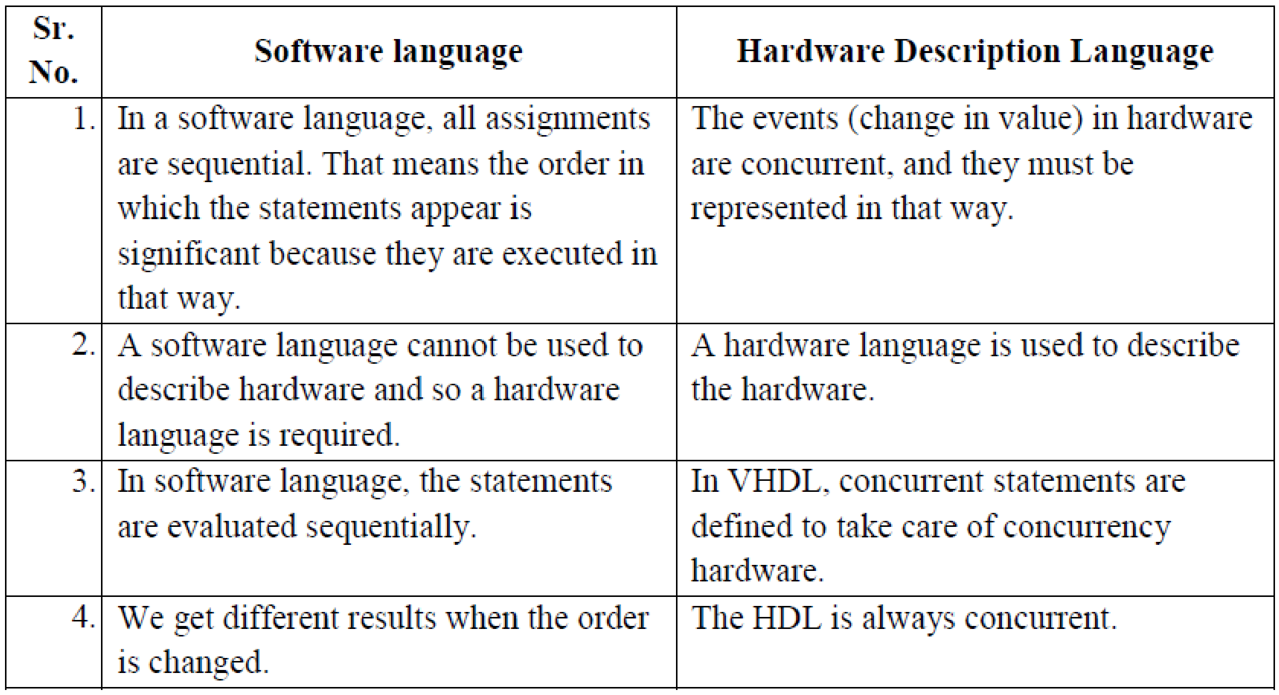
Multiplier Blocks : Accept two 18-bit binary numbers as inputs and calculate the product.

Digital Clock Manager (DCM): Blocks provide self-calibrating, fully digital solutions for distributing, delaying, multiplying, dividing, and phase-shifting clock signals.

Q 2. Attempt any FOUR of the following 16 marks

a) Compare software and hardware description language.

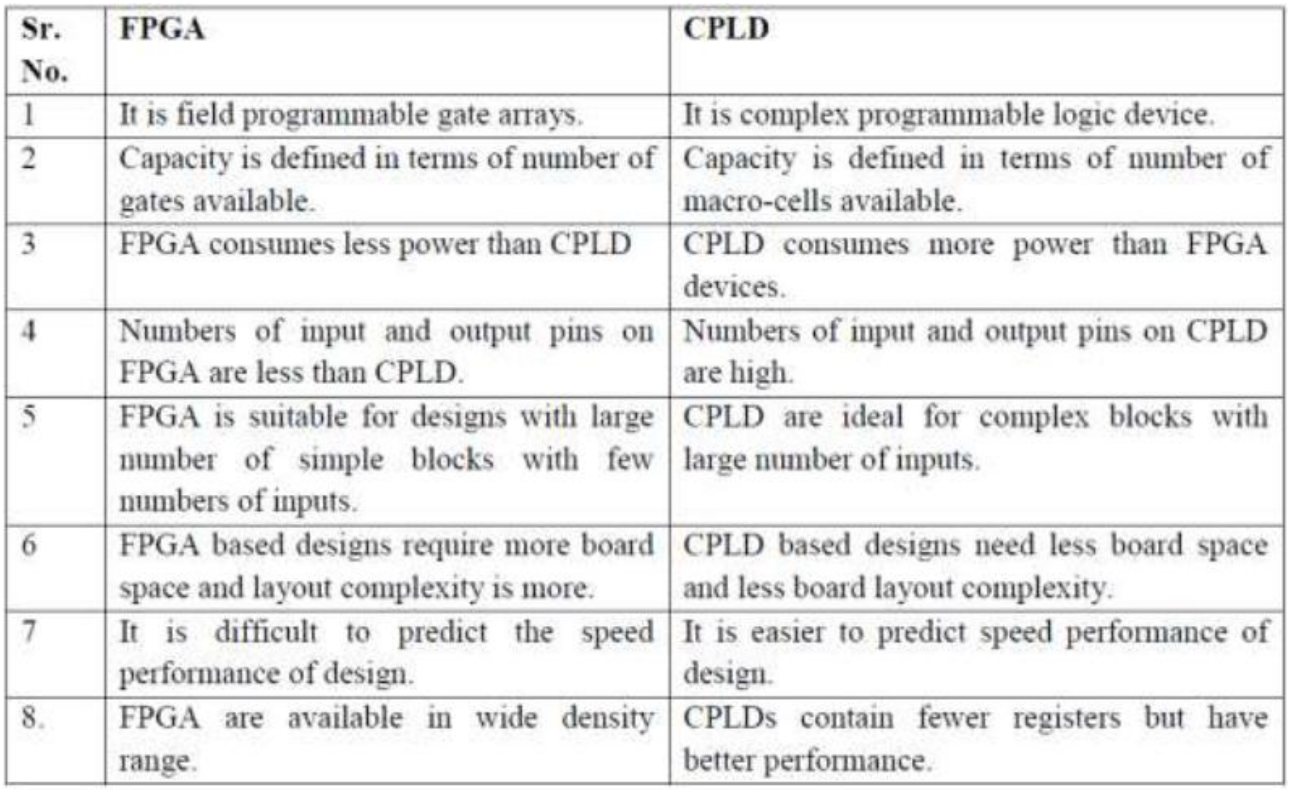
Ans: **(four correct points – 1 mark each)**



Sr. No.	Software language	Hardware Description Language
1.	In a software language, all assignments are sequential. That means the order in which the statements appear is significant because they are executed in that way.	The events (change in value) in hardware are concurrent, and they must be represented in that way.
2.	A software language cannot be used to describe hardware and so a hardware language is required.	A hardware language is used to describe the hardware.
3.	In software language, the statements are evaluated sequentially.	In VHDL, concurrent statements are defined to take care of concurrency hardware.
4.	We get different results when the order is changed.	The HDL is always concurrent.

b) Compare FPGA and CPLD

Ans: **(any four correct points – 1 mark each)**

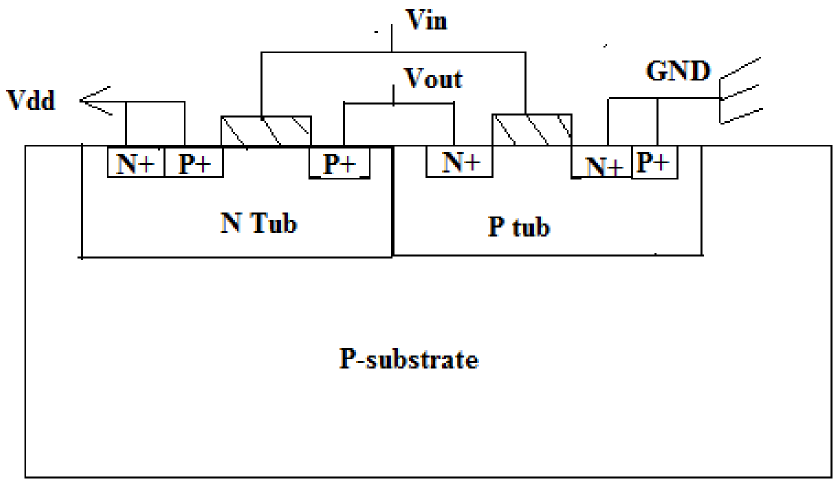


Sr. No.	FPGA	CPLD
1	It is field programmable gate arrays.	It is complex programmable logic device.
2	Capacity is defined in terms of number of gates available.	Capacity is defined in terms of number of macro-cells available.
3	FPGA consumes less power than CPLD	CPLD consumes more power than FPGA devices.
4	Numbers of input and output pins on FPGA are less than CPLD.	Numbers of input and output pins on CPLD are high.
5	FPGA is suitable for designs with large number of simple blocks with few numbers of inputs.	CPLD are ideal for complex blocks with large number of inputs.
6	FPGA based designs require more board space and layout complexity is more.	CPLD based designs need less board space and less board layout complexity.
7	It is difficult to predict the speed performance of design.	It is easier to predict speed performance of design.
8.	FPGA are available in wide density range.	CPLDs contain fewer registers but have better performance.

c) Explain Twin-tub process in CMOS fabrication.

Ans: **(diagram – 2 marks, explanation – 2 marks)**

- A logical extension of the p – well and n – well approaches is the twin – tub fabrication process.
- In this process, a substrate of high resistivity of n – type material is used and then in this n – type material both n – well and p – well regions are created.
- By using this process it is possible to preserve the performance of n – transistor without compromising the p – transistors.
- The doping control is more rapidly achieved and some relaxation in manufacturing tolerance results.
- This is particularly important as far as latch is concerned.
- The twin – tub process allows separate optimization of the n and p transistors.
- Following figure shows the CMOS inverter fabricated using twin – tub process.



d) State and explain different operators used in VHDL.

Ans: **(any four correct operators – 1 mark each)**

The predefined operators in VHDL array of bit and

1) Logical operators: AND, OR, NOT, NAND, NOR, XOR, XNOR.

Logical operators are defined for **type bit** and Boolean, one dimensional array of bit and Boolean type.

2) Relational operators:

i) = equality

ii) /= inequality

iii) < less than

iv) < = less than or equal to

v) > greater than

vi) >= greater than or equal

3) Shift operators:

sll shift left logical

srl shift right logical

sla shift left arithmetic

sra shift right arithmetic

rll right left logical

rrl right right logical

4) Adding operators

'+' operator is used for addition

'-' operator is used for subtraction

'+' and '-' operators are predefined in VHDL for all integer operands.

'&' is the concatenation operator. It works on vector only.

5) Multiplying operators

Multiplying operators are predefined in VHDL for all integer types.

'*' is used for integer multiplication

'/' is used for integer division

'*' and / are used for floating point numbers and are defined only for integers.

6) Miscellaneous operators

The two miscellaneous operators are abs and **

'Abs' means absolute

'**' means exponential.

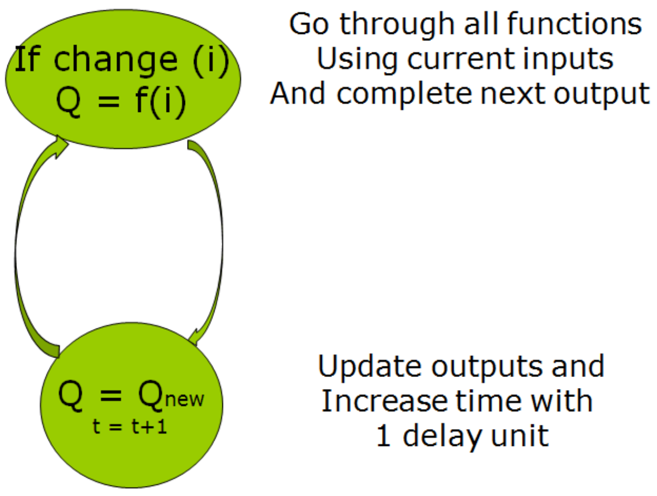
e) Explain different types of simulators.

Ans: **(correct explanation – 2 marks each; figures not compulsory but can be considered)**

There are basically two types of simulators:

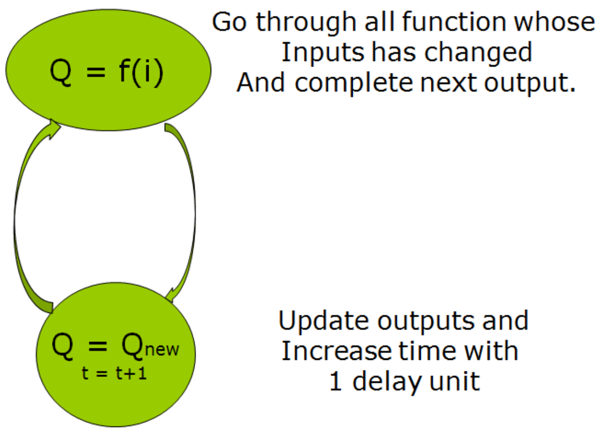
1. Event based simulator

- Event driven signal keeps track of any change in the signal in the event queue.
- The simulator starts simulation as soon as any signal in event list changes its value.
- For this the simulator has to keep record of all the scheduled events in future. This causes a large memory overload but gives high accuracy for asynchronous design. It simulates events only.
- Gates whose inputs have events are called active and are placed in activity list.
- The simulation proceeds by removing a gate from the activity list. The process of evaluation stops when the activity list becomes empty.
- The working of event based simulator is summarised in the figure below.



2. Cycle based simulator

- Cycle-based simulation ignores intra-cycle state transitions, i.e. they check the status of target signals periodically irrespective of any events.
- This can boost performance by 10 to 50 times compared to traditional event-driven simulators.
- Cycle-based technology offers greater memory efficiency and faster simulation run-time than traditional pure event-based simulators.
- Signals are treated as variables. Functions such as AND, OR etc. are directly converted to program statements.
- Cycle-based simulators work best with synchronous design but give less timing accuracy with asynchronous design.
- Signal level functions such as memory blocks, adders, multipliers etc. are modelled as subroutines.
- For every input vector, the code is repeatedly executed until all variables have attained steady value.
- Compiled code simulator is efficient when used for high-level design verification. Inefficiency is incurred by the evaluation of the design when only few inputs are changing.
- The working of cycle based simulator is summarised in the figure below.


Q = f(i)
Go through all function whose Inputs has changed And complete next output.
Q = Qnew
t = t+1
Update outputs and Increase time with 1 delay unit

Q3). Attempt any four of the following. 16 marks

a) Designed locked sequential circuit using Toggle flip-flop to count from 00 to 11 (2 bit counter).

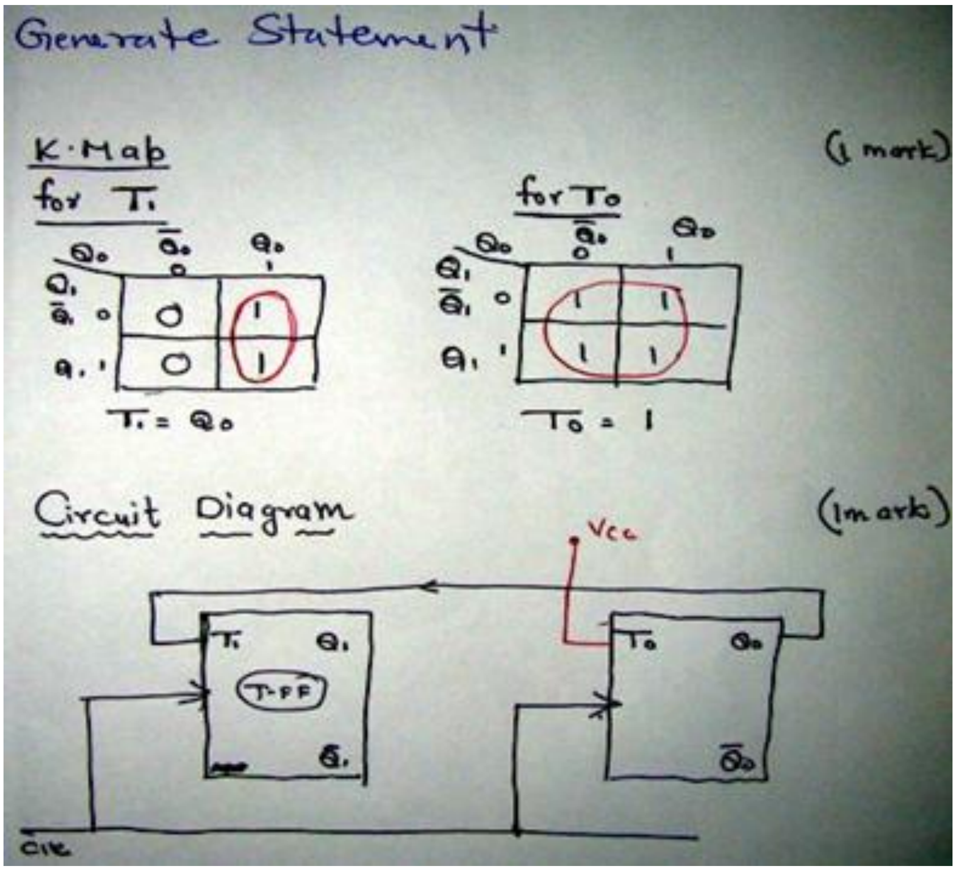
ANS:

- Number of bits= 2 (i.e. 00 – 11)
- Previous State = Q_1Q_0
- Next State = $Q_1^*Q_0^*$
- **Excitation Table for T- Flip Flop** **(1 mark)**

Transition		Toggle input
Qn	**Qn***	**T**
0	0	0
0	1	1
1	0	1
1	1	0

State Table **1 mark**

Count	Previous State		Next State		Excitation	
	Q_1	Q_0	Q_1*	Q_0*	T_1	T_0
0	0	0	0	1	0	1
1	0	1	1	0	1	1
2	1	0	1	1	0	1
3	1	1	0	0	1	1



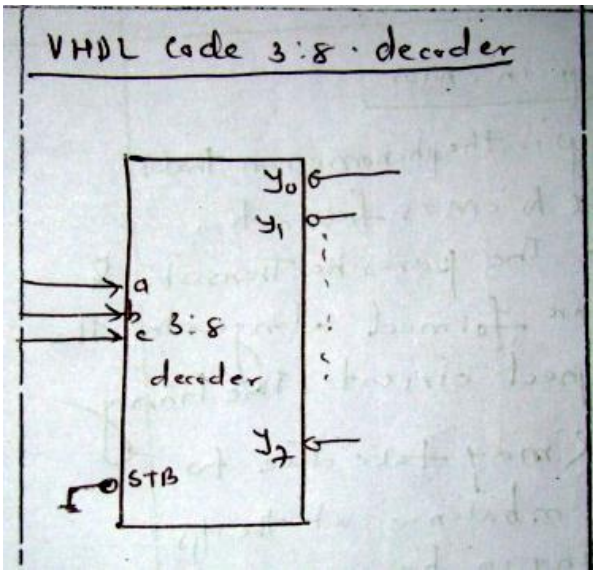
b) List the features of FPGA.

ANS: (any four correct features – 1 mark each)

1. Capacity is defined in term of number of gates available.
2. FPGA consumes less power than CPLD.
3. Numbers of input and output pins on FPGA are less than CPLD.
4. FPGA is suitable for designs with large number of simple blocks with few number of input.
5. FPGA based design requires more board space and layout complexity is more.
6. It is difficult to predict the speed performance of design.
7. FPGA are available in wide density range.
8. Gates more than 10,000.
9. Higher I/O count.
10. Complex architecture.

c) Write the VHDL code for 3:8 decoder.

ANS: (Entity – 1 mark, architecture – 3 marks) <u>Note Diagram is Optional</u>



```
Library IEEE ;
Use IEEE .std _logic =1164.all ;
Entity  decoder is
Port ( a,b,c: in std _logic;
          Stb  :in std_logic ;
             Y : out std_logic _vector(7 downto 0) );
End decoder;
Architecture beh of decoder is
 signal temp : std_ logic_vector(3 downto 0);
   begin
temp <=stb & a & b &c ;
Y  <=”01111111” when temp =” 0000” else
       “10111111” when temp = “0001” else
     ( and so on for each input )

        “11111110” when temp=”0111”else
         “zzzzzzzz;
End beh;
```

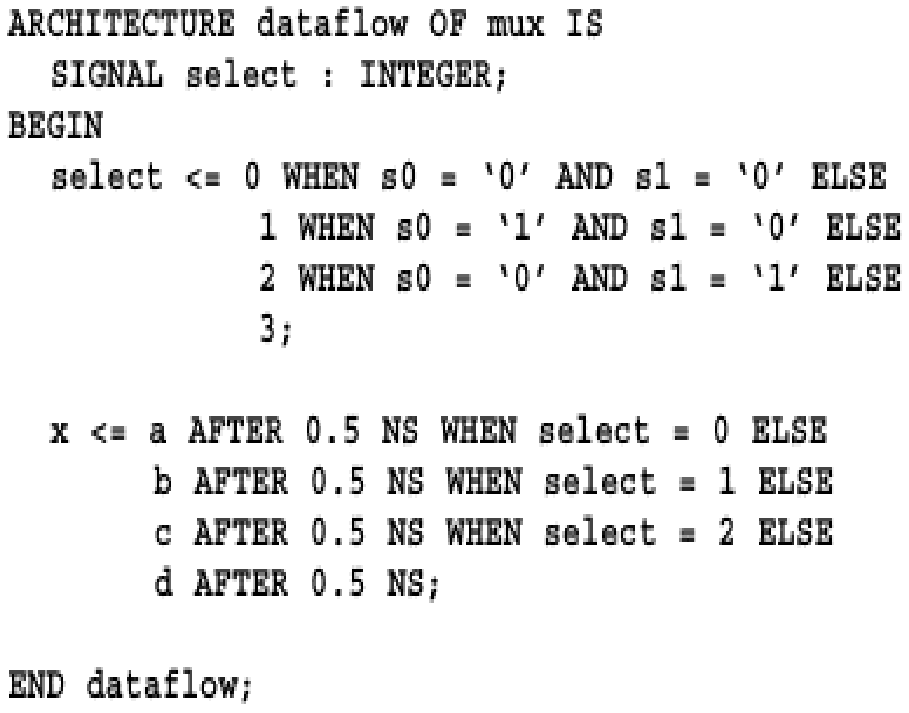
(Any logic using with ----select or case statement or if statement can be used for program. Give marks to entity declaration and interpretation of syntax)

d) Explain event scheduling and sensitivity list.

ANS: (each correct explanation – 2 marks)

Event Scheduling:

1. Event is nothing but change on target signal which is to be updated.
2. Ex. X<= a after 0.5ns when select=0 else
   X<= b after 0.5ns
3. The assignment to signal x does not happen instantly. Each of the values assigned to x contain an after clause.
4. The mechanism for delaying the new value is called scheduling an event. By assigning port x a new value, an event was scheduled 0.5ns in the future that contains the new value for signal x. when the event matures, signal receives a new value.
5. Example:



```
ARCHITECTURE dataflow OF mux IS
  SIGNAL select : INTEGER;
BEGIN
  select <= 0 WHEN s0 = '0' AND s1 = '0' ELSE
            1 WHEN s0 = '1' AND s1 = '0' ELSE
            2 WHEN s0 = '0' AND s1 = '1' ELSE
            3;

  x <= a AFTER 0.5 NS WHEN select = 0 ELSE
       b AFTER 0.5 NS WHEN select = 1 ELSE
       c AFTER 0.5 NS WHEN select = 2 ELSE
       d AFTER 0.5 NS;

END dataflow;
```

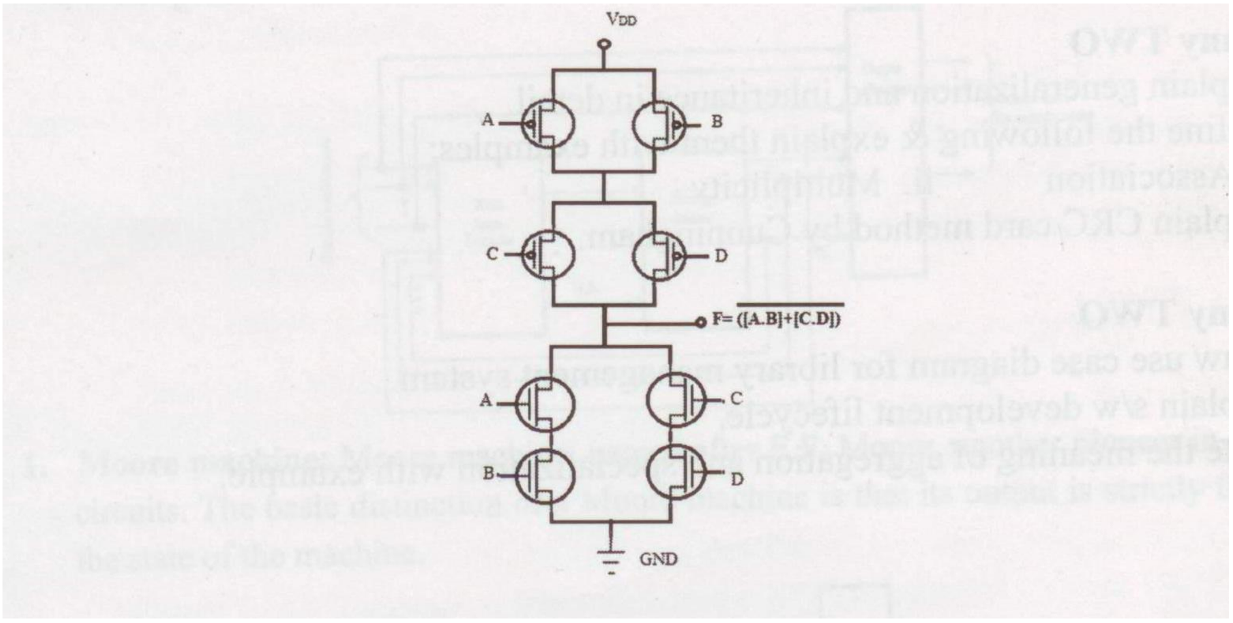
Sensitivity List:

1. Every concurrent statement has a sensitivity list. Statements are executed only when there is an event or signal in the sensitivity list, otherwise they are suspended.
2. Ex. F<=A and B;
3. A and B are in the sensitivity list of F.
4. The statement will execute only if one of these will change.
5. Ex. Process(CLK, RST)
6. The process is sensitive to RST and CLK signal i.e. an event on any of these signals will cause the process to resume.

e) Implement the logic circuit using CMOS.

$\gamma = \overline{([A.B] + [C.D])}$

ANS: (Any relevant correct Diagram 4M)



Q4) a). Attempt any three of the following: 12 marks

i. Explain with syntax.
   1. Entity. 2. Architecture.

ANS: (Each correct explanation – 2 marks) (structure is optional)

Entity:

- A VHDL Entity specifies the name of the entity, the ports of the entity, and other entity related information.
- All designs are created using one or more entities.
- **Syntax:**

  Entity <entity_name> **is**

  Generic (<generic_list>);

  Port (<port_list>);

  End <entity_name>

- The keyword entity signifies that this is the start of the entity statement. The standard type provided is BIT.
- **Example**

```
ENTITY mux is
PORT (a, b, c, d: IN_BIT;
          s0, s1: IN_BIT;
          Y: OUT_BIT);
END mux;
```

- Name of the user created object is mux. The name of the entity is mux.

- The entity has seven ports in the PORT Clause.
- Six of them are input ports and one is output port which is notified as IN and OUT respectively.
- The four data input ports (a, b, c, d) and two select input ports (s0, s1) and one output port (y) are of type BIT.
- The entity describes the interface to the outside world. It specifies the number of ports, direction of the ports, type of ports, etc.

Architecture:

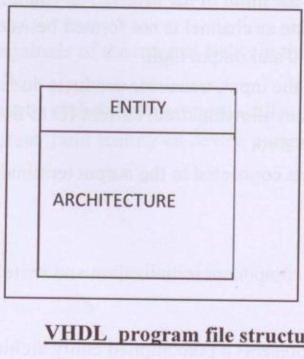
- The entity describes the interface to the VHDL model.
- The architecture specifies behaviour, function, interconnection or relation between input and output of an entity.
- **Syntax:**

```
Architecture architecture_name of entity_name is
architecture_declarations
Begin
concurrent_statements
End [architecture] [architecture_name];
```

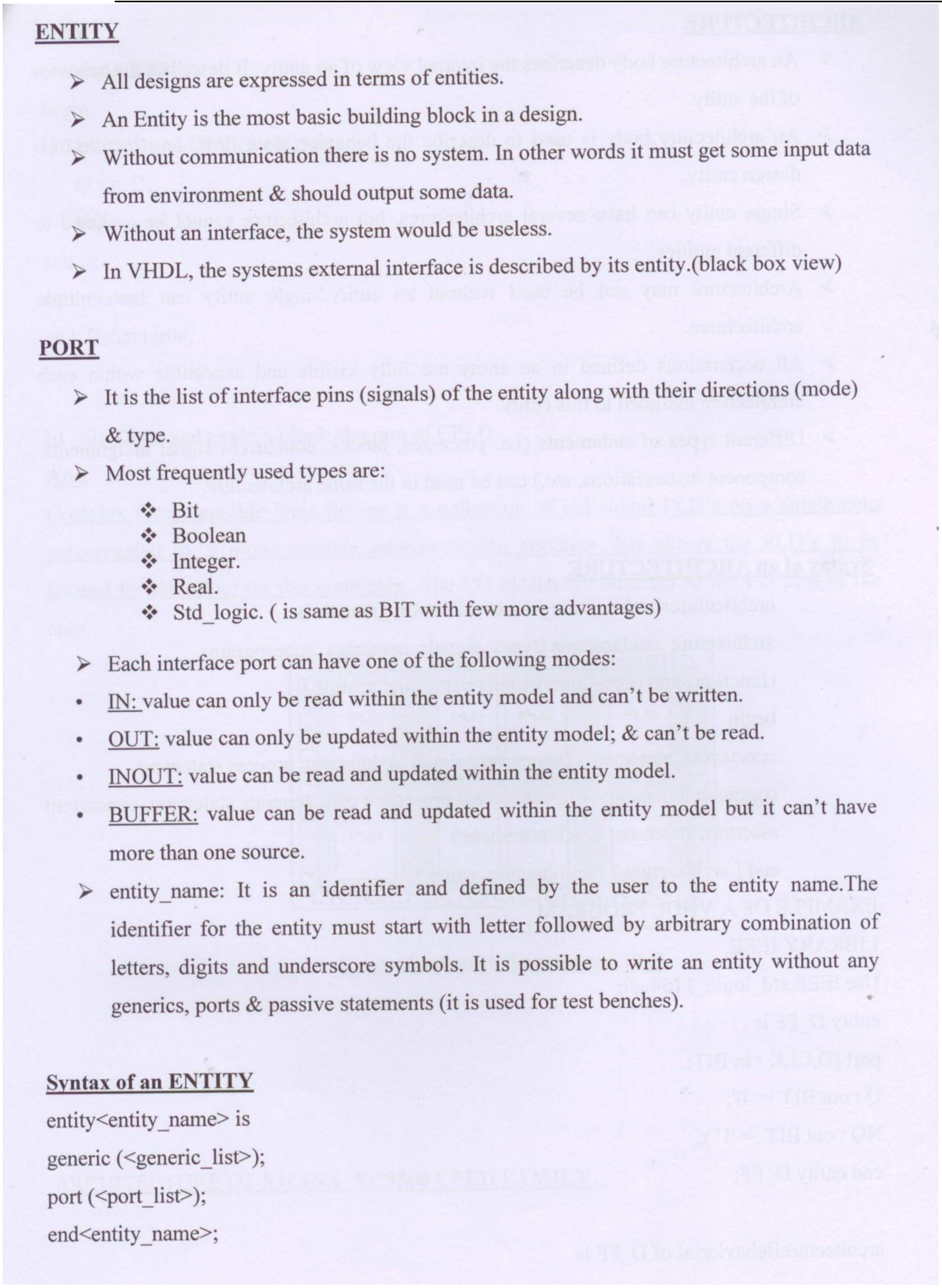
- It describes the contents of an entity. The reason for connection between the architecture and the entity is that an entity can have multiple architectures describing the behaviour of the entity.
- The keyword ARCHITECTURE signifies that this statement describes architecture for an entity.
- The statement of the architecture starts with the keyword BEGIN and ends with the END netlist statement.
- **Example:**

```
Architecture AND1 of ANDGATE is
--declarations
Begin
--statements
Y <= A AND B;
End architecture AND1;
```

(OR)



VHDL program file structure



ENTITY

- All designs are expressed in terms of entities.
- An Entity is the most basic building block in a design.
- Without communication there is no system. In other words it must get some input data from environment & should output some data.
- Without an interface, the system would be useless.
- In VHDL, the systems external interface is described by its entity.(black box view)

PORT

- It is the list of interface pins (signals) of the entity along with their directions (mode) & type.
- Most frequently used types are:
    - Bit
    - Boolean
    - Integer.
    - Real.
    - Std_logic. (is same as BIT with few more advantages)
- Each interface port can have one of the following modes:
- IN: value can only be read within the entity model and can't be written.
- OUT: value can only be updated within the entity model; & can't be read.
- INOUT: value can be read and updated within the entity model.
- BUFFER: value can be read and updated within the entity model but it can't have more than one source.
- entity_name: It is an identifier and defined by the user to the entity name.The identifier for the entity must start with letter followed by arbitrary combination of letters, digits and underscore symbols. It is possible to write an entity without any generics, ports & passive statements (it is used for test benches).

Syntax of an ENTITY

```
entity<entity_name> is
generic (<generic_list>);
port (<port_list>);
end<entity_name>;
```

ARCHITECTURE

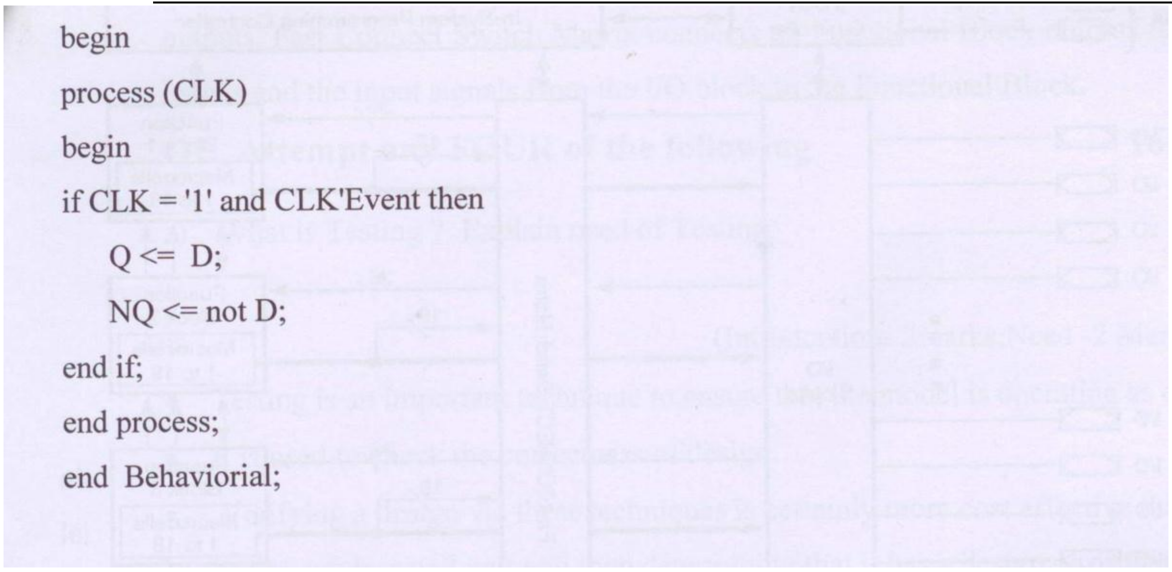
- An architecture body describes the internal view of an entity. It describes the behavior of the entity.
- An architecture body is used to describe the behavior, data flow, or structure of a design entity.
- Single entity can have several architectures, but architecture cannot be assigned to different entities.
- Architecture may not be used without an entity.Single entity can have ultiple architectures.
- All declarations defined in an entity are fully visible and accessible within each architecture assigned to this entity.
- Different types of statements (i.e. processes, blocks, concurrent signal assignments, component instantiations, etc.) can be used in the same architecture.

Syntax of an ARCHITECTURE

```
architecture<architecture_name> of <entity_name> is
architecture_declarations (types, signals, constants, subprograms
(functions and procedures),components, and groups.)
begin
concurrent_statements (concurrent signal assignment, process statement,
component instantiation, concurrent procedure call, generate statement, concurrent
assertion statement block statement.)
end [ architecture ] [ architecture_name ];
```

EXAMPLE OF A VHDL PROGRAM

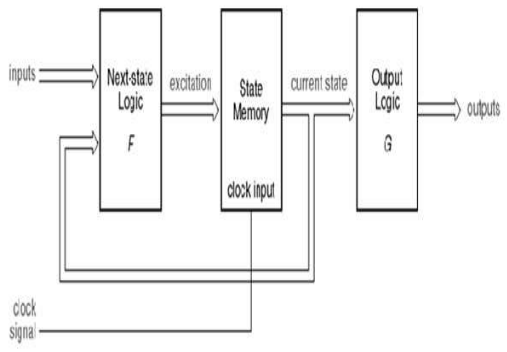
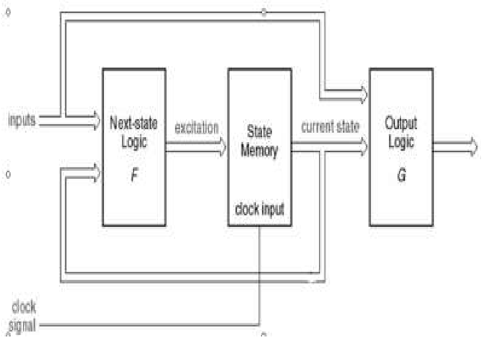
```
LIBRARY IEEE;
Use IEEE.std_logic_ 1164.all;
entity D_FF is
port (D,CLK : in BIT;
Q : out BIT := '0';
NQ : out BIT := '1' );
end entity D_FF;

architectureBehaviorial of D_FF is
```



```
begin
process (CLK)
begin
if CLK = '1' and CLK'Event then
      Q <=  D;
      NQ <= not D;
end if;
end process;
end  Behaviorial;
```

ii. Compare Mealy Machine with moor machine.

ANS: (any four correct points – 1 mark each)

Moore machine	Mealy Machine
Definition :output is function of state of machine	Definition :output is function of state of machine and present input condition
Requires more number of states	Requires less number of states
Faster	slower
Design simple	Design complex
Output in state	Output is at the time of state transition
Block diagram  inputs Next-state Logic F excitation State Memory clock input current state Output Logic G outputs clock signal	Block diagram  inputs Next-state Logic F excitation State Memory clock input current state Output Logic G clock signal

iii. Explain sharing of complex operators in VHDL with suitable example.

ANS: **(any relevant correct explanation with suitable example – 4 marks)**

The efficiency of a synthesized design depends primarily on how you describe its component structure. The optimization of individual components, especially those made from random logic, produces similar results from two very different descriptions.

Therefore, concentrate the majority of your design effort on the implied component hierarchy, rather than on the logical descriptions.

VHDL compiler supports many shorthand VHDL expressions. There is no benefit with a verbose syntax when a shorter description is adequate.

Example shows four equivalent groups of statements.

Example equivalent statements

```
Signal A, B, B: BIT_VECTOR (3 downto 0);
……
C < = A and B;
```

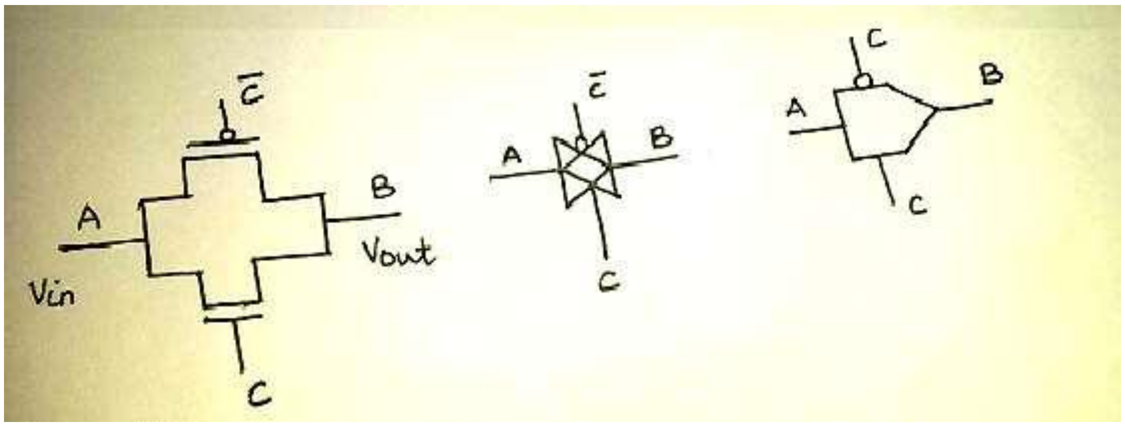
```
C (3 downto 0) < = A (3 downto 0) and B (3 downto 0);
```

```
C (3) < = A (3) and B (3);
C (2) < = A (2) and B (2);
C (1) < = A (1) and B (1);
C (0) < = A (0) and B (0);
```

```
For I in 3 downto 0 loop
C (I) < = A (I) and B (I);
End loop;
```

iv. Draw and explain working of CMOS transmission gates.

ANS: **(diagram – 2 marks, explanation – 2 marks)**

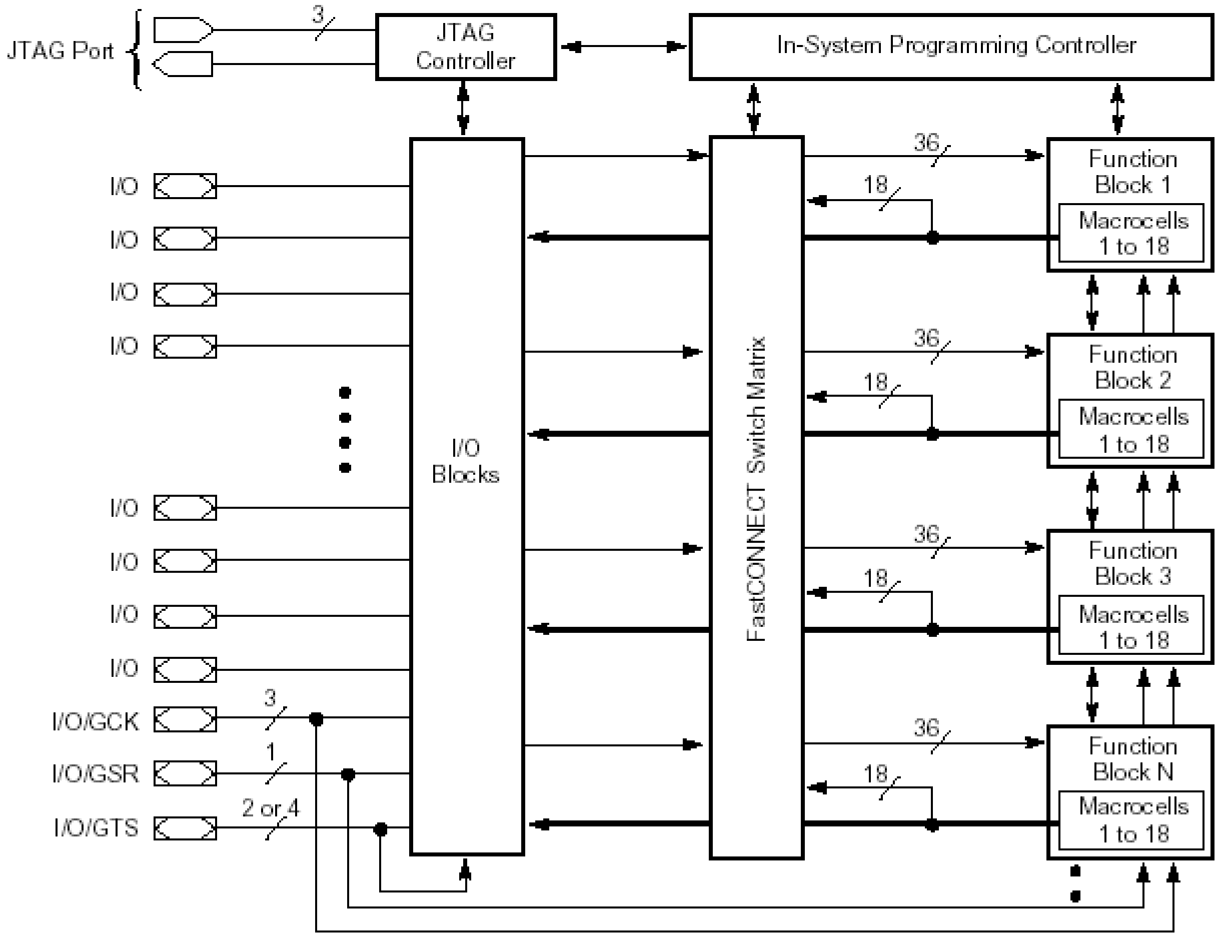


- It consists of one nMOS and one pMOS transistor in parallel.
- The gate voltages, applied to these two transistors are also set to be complementary signals.
- The CMOS Transmission gate operates as a bidirectional switch between the nodes A & B which is controlled by C.
- If the control signal C is logic high, VDD, then both the transistors are turned ON and provides a low resistance current path between the nodes A & B.
- If C is low, then both the transistors are off & path between A & B is open circuit.
- This condition is called high impedance state.

b) Attempt any one of the following: 6 marks

i) Explain the architecture of Xilinx family of CPLD.

ANS: (Diagram 3M Explanation 3M)



Explanation:

- Each external I/O pin 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 a “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 as “Global Three State Controls”(GTS).One of the signals can be selected in each macrocell to output enable the corresponding output driver when the macrocells”s output is hooked to an external I/O pin.
- Only four Functional Blocks (FB) are shown but XC9500 scales to accommodate 16 FB‟s in the XC95288.Regardless of the specific family member each FB programmable receives 36 signals from the switch matrix. The inputs to the switch matrix are the 18 macrocell outputs from each of the functional blocks and the external inputs from the I/O pins.
- Each Functional block also has 18 outputs that run under the switch matrix and connect to the I/O blocks. These are the output-enable signals for the I/O block output drives; they’re used when FB macrocells”s output is hooked up to 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 the I/O blocks and the input signals from the I/O block to the Functional Block.

ii) Write the VHDL code for 8:1 MUX.

ANS: (entity – 2 marks, architecture – 4 marks)

Note: Any type of modeling should be considered

Any type of statements used should be considered

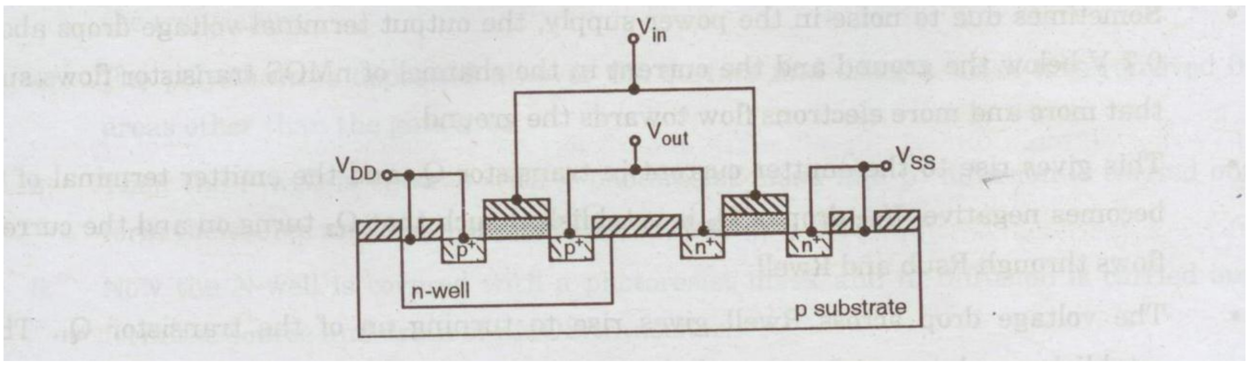
```
LIBRARY IEEE;//standard library.
USE IEEE.STD_LOGIC_1164.ALL;//importing standard library.
USE IEEE.STD_LOGIC_ARITH.ALL;
//entity declaration
ENTITY 8mux1 IS
PORT (I: IN STD_LOGIC_VECTOR (7 downto 0);
S: IN STD_LOGIC_VECTOR (2 downto 0);
Q: OUT STD_LOGIC);
END 8mux1;
//end of entity declaration
ARCHITECTURE behave OF 8mux1 IS
BEGIN
PROCESS (I, S) //sensitivity list.
BEGIN
Y < = I (0) when (S = "000") else
        I (1) when (S = "001") else
        I (2) when (S = "010") else
        I (3) when (S = "011") else
        I (4) when (S = "100") else
        I (5) when (S = "101") else
        I (6) when (S = "110") else
        I (7);
END PROCESS;
END behave OF 8mux1;//end of architecture.
```

(OR)

```
LIBRARY IEEE;//standard library.
USE IEEE.STD_LOGIC_1164.ALL;//importing standard library.
USE IEEE.STD_LOGIC_ARITH.ALL;
//entity declaration
ENTITY 8mux1 IS
PORT (I: IN STD_LOGIC_VECTOR (7 downto 0);
S: IN STD_LOGIC_VECTOR (2 downto 0);
Q: OUT STD_LOGIC);
END 8mux1;
//end of entity declaration
ARCHITECTURE behave OF 8mux1 IS
BEGIN
PROCESS (I, S) //sensitivity list.
BEGIN
Case S is
when "000" = >Y < = I (0);
when "001" = >Y < = I (1);
when "010" = >Y < = I (2);
when "011" = >Y < = I (3);
when "100" = >Y < = I (4);
when "101" = >Y < = I (5);
when "110" = >Y < = I (6);
when "111" = >Y < = I (7);
when others = > null;
 END case;
END PROCESS;
END behave OF 8mux1;//end of architecture.
```

iii) Explain the steps involved in fabrication of n-well process.

ANS: (diagram – 3 marks, explanation – 3 marks)



Explanation

The N- well CMOS circuit are getting more popular because of the lower substrate bias effects on transistor threshold voltage and lower parasitic capacitances associated with source and drain regions. The typical N- well fabrication steps are similar to p-well fabrication steps.

1.Thick SiO_2 layer is grown on p-type silicon wafer.
2.After defining the area for N- well diffusion, using a mask, the SiO_2 layer is etched off and n- well diffusion process is carried out.
3.Oxide in the n transistor region is removed and thin oxide layer is grown all over the surface to insulate gate and substrate.
4.The polysilicon is deposited and pattern on thin oxide region using a Mask to form gate of both the transistors. The thin oxide on source and drain regions of both the transistors is removed by proper masking steps.
5.Using n^+ mask and complimentary n^+ mask, source and drain of both nMOS and pMOS transistors are formed on after the other using respective diffusion processes. These same masks also include the V_{DD} and V_{SS} contacts.
6.The contacts cuts are made using proper masking procedure and metal is deposited and patterned on the entire chip surface.
7.An overall passivation layer is formed and the openings for accessing bonding.

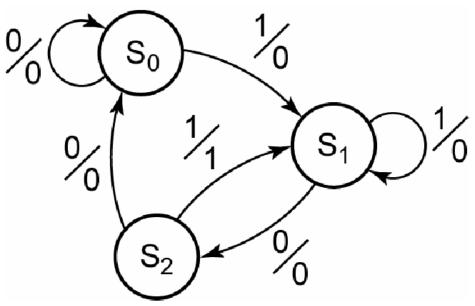
Q5). Attempt any four of the following: 16 marks

a) Draw and Implement the T flip-flop using Moor machine.

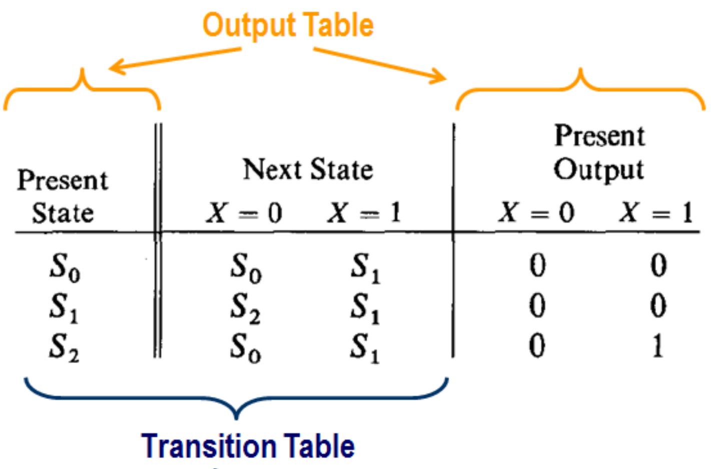
ANS: (Proper relevant stepwise design – 4 marks)

Let's consider a sequence detector 101 which can be implemented with T flip flop using Moore machine.

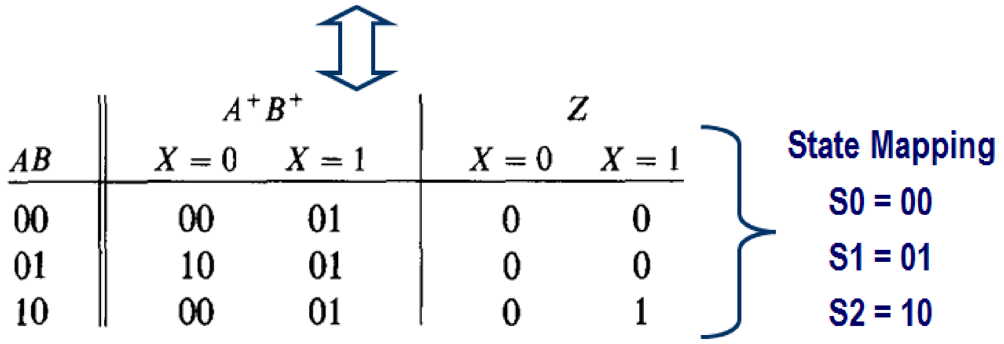
The state diagram will be (1 mark)



Transition table will be (1 mark)

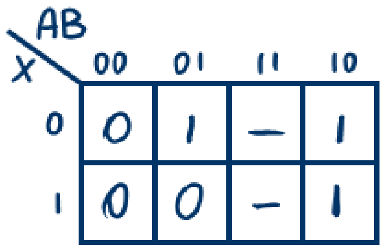

Output Table
Transition Table

Present State	Next State		Present Output	
	$X = 0$	$X = 1$	$X = 0$	$X = 1$
S_0	S_0	S_1	0	0
S_1	S_2	S_1	0	0
S_2	S_0	S_1	0	1

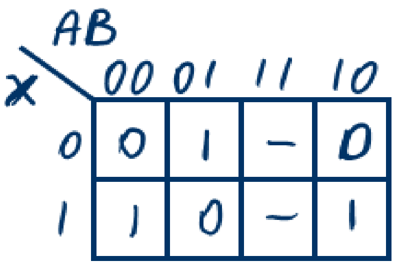

State Mapping
S0 = 00
S1 = 01
S2 = 10

AB	A^+B^+		Z	
	$X = 0$	$X = 1$	$X = 0$	$X = 1$
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1

K – maps for implementing design using T flip-flop (1 mark)



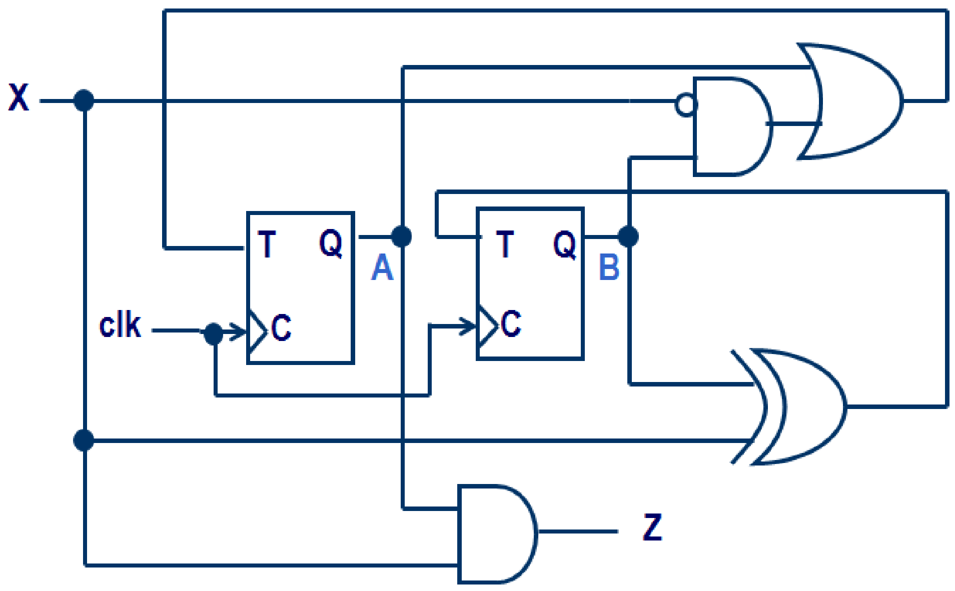
X \ AB	00	01	11	10
0	0	1	–	1
1	0	0	–	1



X \ AB	00	01	11	10
0	0	1	–	0
1	1	0	–	1

$T_A = A + \overline{X} \cdot B$ $\qquad$ $T_B = \overline{X} \cdot B + X \cdot \overline{B} = X \oplus B$

Final Design using T flip-flops (1 mark)



b) Define the following terms related to fabrication process.

i) Oxidation.
ii) Diffusion.
iii) Icon – implantation.
iv) Deposition.

ANS: (each correct definition – 1 mark)

(i) Oxidation:

Oxidation is a process by which a layer of silicon dioxide is grown on the surface of a silicon wafer. The oxidation of silicon is necessary throughout the modern integrated circuit fabrication process.

(ii) Diffusion:

It is the process by which impurities may be introduced into selected region of a semiconductor. In the silicon technology diffusion allows formation of sources and drains for metal-oxide-semiconductor devices. It is extensively used because it is ideally adopted to batch processes where many slices are handled in single operation. It does not produce crystal damage, thus high quality junctions with minimum leakage current can be made easily by this method. It is achieved by heating the wafer to a high temperature and passing a gas containing the desired impurity.

(iii) Ion- Implantation:

Ion implantation can be defined as the process by which impurity ions are accelerated to high velocity and physically lodged into the target material.

(iv) Deposition:

Deposition is a process followed by an implantation step to reduce poly resistance.

c) Describe the following statement with syntax.

i) Process statement
ii) Case statement.

ANS: (each correct statement with syntax – 2 marks)

i) Process statement:

- The process statement is the primary concurrent VHDL statement used in sequential behavior. The statements within process statement are sequential statements.
- The process statement can exist anywhere in the architecture and define region in the architecture where all statements are sequential, i.e. they are executed one after another, in the order in which they are written.

- Process places only one driver on the signal. The value that the signal updated with, is the last value assigned to it within the process execution.
  e.g. : Z <= A and B ;
  Z < = C and D;
  It wants create two drivers, Z will get the value generated by the statement 'C and D'.
- The process statement can have explicit sensitivity list. This list defines the signals, that causes the statements inside the process statement to execute whenever one or more elements of the list change value i.e. the sensitivity list is a list of the signals that the process is sensitive to changes on.
- The process should either have 'sensitivity list' or a 'wait' statement at the end.
- These statements should to be executed in the given sequence, otherwise the signal value will be different.

Syntax:

```
process (sensitvity_list)
begin
  sequential_statements
end process ;
e.g.   process (B, C, D)
       begin
          A <= B ;
          B <= C ;
          C <= D ;
        end process ;
```

ii) case statement:

Case Statement:

A sequence statement selects for execution one of the several alternative sequences is called case statement.

Eg. VHDL program for 4:1 Multiplexer using case statement**.**

Program:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity multiplexer4_1 is
port (
i : in std_logic_vector(3 downto 0);
sel : in std_logic_vector(1 downto 0);
y : out std_logic );
end multiplexer4_1;
architecture Behavioral of multiplexer4_1 is
begin
process(i0,i1,i2,i3,sel)
begin
```

```
case sel is
when "00" => y <= i(0);
when "01" => y <= i(1);
when "10" => y <= i(2);
when others => y <= i(3);
end case;
end process;
end Behavioral;
```

d) List the advantages and disadvantages of VHDL.

ANS: (any two advantages – 2 marks, two disadvantages – 2 marks)

Advantages of VHDL:

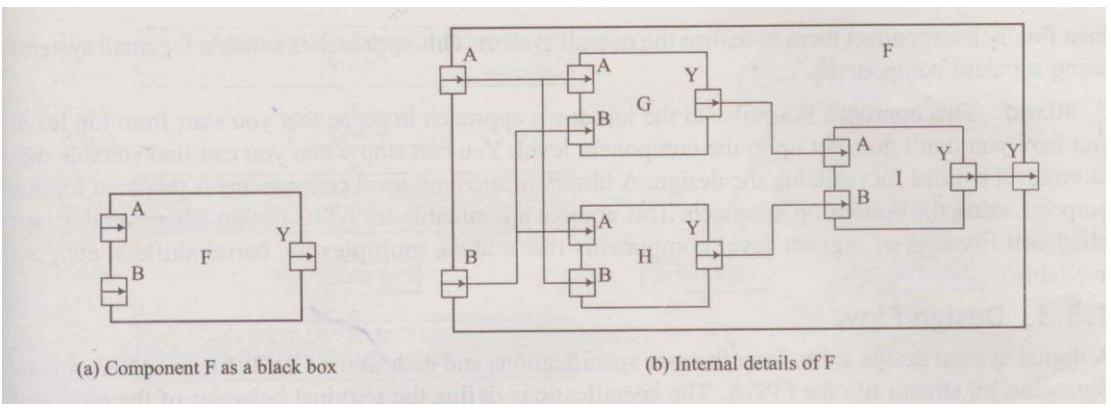
1. Standard language.
2. Fully expressive language.
3. Hierarchical.
4. Configurable.
5. Tool availability.
6. Consistency and completeness checks.
7. Tight coupling to lower levels of design.
8. Supports hybrid modeling.

Disadvantages of VHDL:

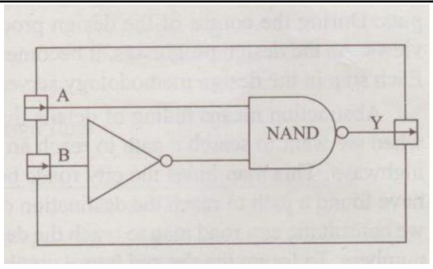
1. Extreme verbose coding
   - VHDL modules must be defined by a prototype and declared before they are used, causing you to change code in atleast three places if you want to make a change to the interface.
2. Sensitivity lists
   - Missing a single signal in the sensitivity list can cause major differences between simulation and synthesis.
   - Each process must have a sensitivity list that may sometimes be very long.
3. Type conversions
   - Signal types that are clearly related (e.g. std_logic and Std_logic_vectors) cannot be simply used together and must be converted into another type.

e) State different coding styles in VHDL and explain any one.

ANS: (List of coding styles – 2marks, any one explanation – 2 marks)



(a) Component F as a black box (b) Internal details of F



Coding style: the design process starts with the description of the desired hardware. A hardware circuit description can be broadly classified into two categories-structural and behavioral.

1. **Structural description**: In this method, the circuit is described as a interconnection of known components. In figure the component F is shown as a black box and in figure, it is shown as interconnection of three components G, H, and I. the component I is a 2-input NAND gate and components G and H are two instances of a device whose internal structure are shown in figure. The components G and H have further structures as shown in figure. As this examples shows, the structural description is very useful for a person who build the circuit from basic components but very difficult for a person who wants to understand its working.
2. **Behavioral description:** In this method the circuit is describes by its behavior, i.e. by means of input-output relation in terms of Boolean equations or a set of sequential instructions, the above circuit can be analyze to study its. Behavior. Let us denote output of a device by its instance name.

Thus G = (A.B')' and H = (B.A')'
Therefore, I = (G.H)' = ((A.B.')'. (B.A'))' = A.B' + A'.B = A XOR B

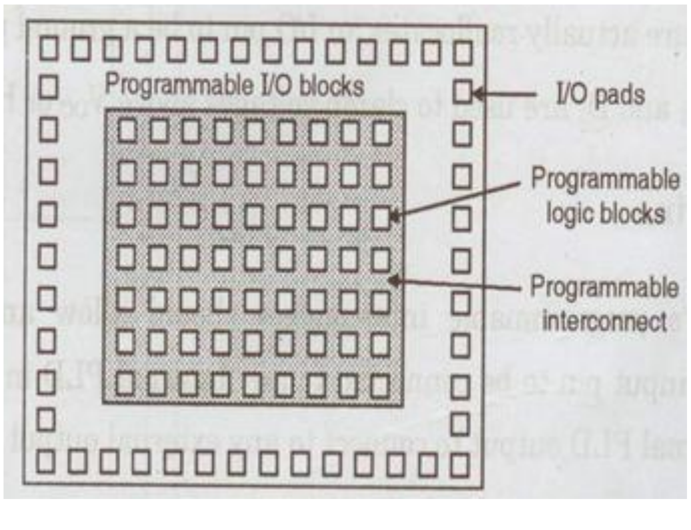
The above circuit can be describes in a behavioral way as follows: Y = A XOR B

The behavior description is closer to human thinking and structural description is closer to physical implementation. The process of synthesis converts the behavioral description to structural description.

Behavior description has some more flavors. If it consists of concurrent signal assignments representing flow of data through the circuit, it is called Data flow description. The data flow operation take placed in a single clock cycle. If the processing operation is spread over several clock cycles with intermediate results being stored in registers, it is called Register Transfer level (RTL) description. Pure behavioral description is like a C program, with no through given to its synthesis.

f) Draw the general FPGA chip architecture and explain the same.

ANS: (diagram – 2 marks, explanation – 2 marks)



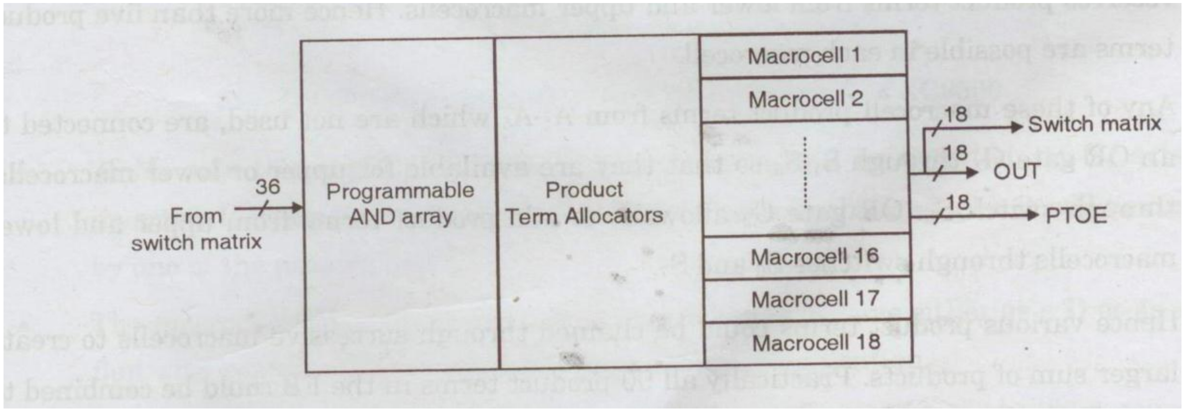
Field programmable Gate Arrays (FPGA):

- A field programmable gate array (FPGA) ha large number of programmable logic blocks that are individually smaller than a PLD. The basic structure of a FPGA is sown in figure.
- The programmable logic blocks are arrange in the matrix form with programmable interconnections and the entire array is surrounded by programmable I/O blocks. Each logic block is less capable than a typical PLD, but it has lot more logic blocks than a CPLD of the same size.

Q6). Attempt any four of the following: 16 marks

a) Explain the product term allocator of Xilinx CPLD family.

ANS: (diagram – 2 marks, explanation – 2 marks)



- The programmable AND array has just 90 product terms i.e. only five AND terms per macrocell. But it does not bring down the efficiency of the FB as it has product term allocator.
- If you are finding it a little difficult to understand, then take the example of sum of product equation
  $y = A\bar{B}C + \bar{A}\bar{B}C + AB\bar{C}$.
- Each term in the equation is a product terms, we need and requires and gate, (AND array is CPLD) and finally to add up all this terms, we need an OR gate, which is in the macrocell.
- The product term allocator allows a macro cell's unused product terms to be used by another nearby macrocells in the same FB.

b) Write VHDL code for FULL ADDER.

ANS: (Entity – 1 mark, Architecture – 3 marks) Truth table and K map optional

<u>Note: Any type of modeling should be considered</u>

<u>Any type of statements used should be considered</u>

FULL ADDER:

A	B	CIN	SUM	COUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

SUM	00	01	11	10
0		1		1
1	1		1	

CARRY	00	01	11	10
0			1	
1		1	1	1

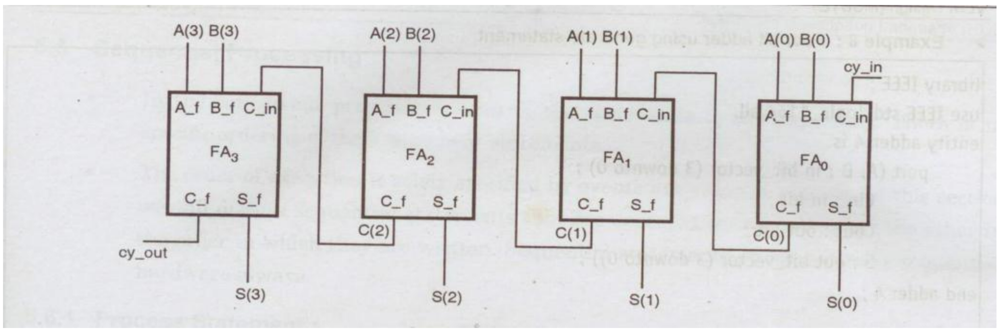
SUM = A XOR B XOR C;

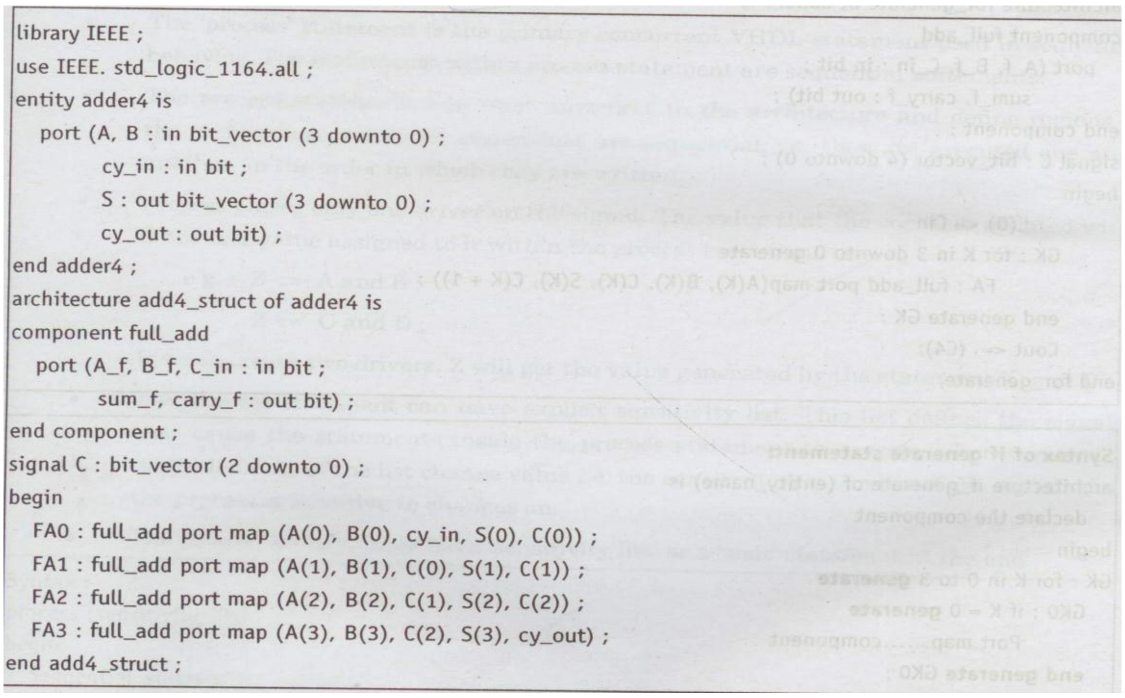
CARRY = AB + AC + BC;

<u>--VHDL code for DATA FLOW model of Full Adder:</u>

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
Entity FA_DF is
Port (A, B, C: in BIT;
        SUM, CARRY: out BIT);
End FA_DF;
Architecture FA_dataflow of FA_DF is
Begin
SUM < = A XOR B XOR C;
CARRY < = (A AND B) OR (B AND C) OR (A AND C);
End FA_dataflow;
```

(OR)





```
library IEEE ;
use IEEE. std_logic_1164.all ;
entity adder4 is
   port (A, B : in bit_vector (3 downto 0) ;
         cy_in : in bit ;
         S : out bit_vector (3 downto 0) ;
         cy_out : out bit) ;
end adder4 ;
architecture add4_struct of adder4 is
component full_add
   port (A_f, B_f, C_in : in bit ;
         sum_f, carry_f : out bit) ;
end component ;
signal C : bit_vector (2 downto 0) ;
begin
   FA0 : full_add port map (A(0), B(0), cy_in, S(0), C(0)) ;
   FA1 : full_add port map (A(1), B(1), C(0), S(1), C(1)) ;
   FA2 : full_add port map (A(2), B(2), C(1), S(2), C(2)) ;
   FA3 : full_add port map (A(3), B(3), C(2), S(3), cy_out) ;
end add4_struct ;
```

c) State the following data types.

I. Scalar data types.
II. Composite data types.

ANS: (relevant explanation – 2 marks each)

I. Scalar data types:

The scalar data types describe objects that can hold at most one value at a time.

The type itself can contain multiple values.

- Integer types.
- Real types.
- Enumerated types.
- Physical types.

Integer types:

They are like mathematical integers. An integer type defines a type whose set of values fall within a specified integer range.

Real types:

They are used to declare objects that emulate mathematical real numbers. It has a set of values in the given range of real numbers.

The predefined real data type covers the range -1.0E38 to +1.0E38, and it must provide atn least six decimal digits of precision.

Enumerated types:

This declaration defines a set of user defined values consisting of identifier and character literals.

e.g. type micro_op is (load, store, add, sub, mul, div);

Hence micro_op is enumerated type and supports type values load, store, add, sub, etc.

type MUL is (‘U’, ‘O’,’1’,’Z’);

similarly MUL is an enumerated type that has the set of ordered values ‘U’,’0’,’1’ and ‘Z’. the order in which values appear in an enumeration type declaration, defines their ordering i.e.

store < div is true

sub > mul is false

Physical types:

A physical type contain value that represent measurements of some physical quantity, like time, length, voltage or current. Values of this type are expressed as integer multiples of a base unit.

The base unit is nano ampere (nA), while all athers are derived units.

II.Composite data types:

A composite type represents a collection of values. There are two composite types:

- Array type.
- Record type.

An array type represents a collection of values all belonging to a single type; on the other hand, a record type represents a collection of values that may belong to different types.

Array type:

An object of an array type consists of elements that have same type.

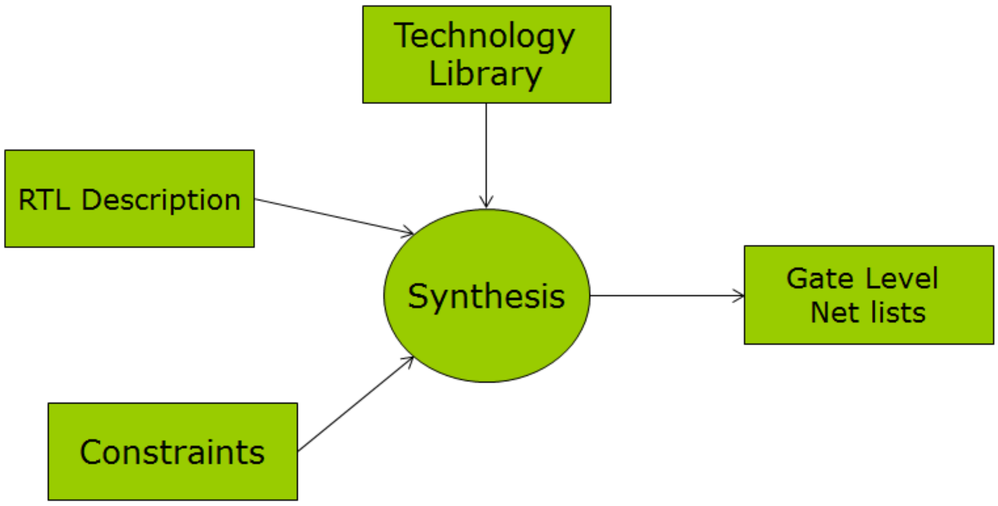
Record type:

An object of a record type is composed of elements of same or different types.

d) Draw HDL design flow for synthesis.

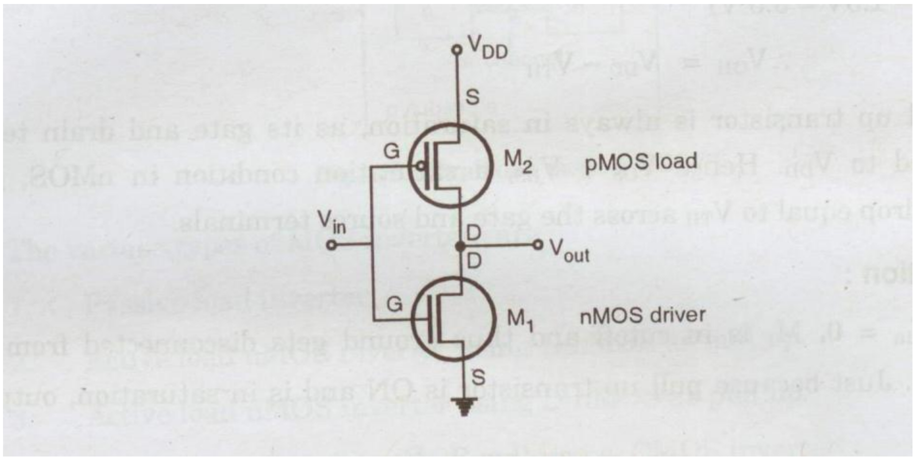
ANS: (correct diagram – 4 marks)

Synthesis = Translation + Optimization.



e) Draw and explain working of CMOS Inverter.

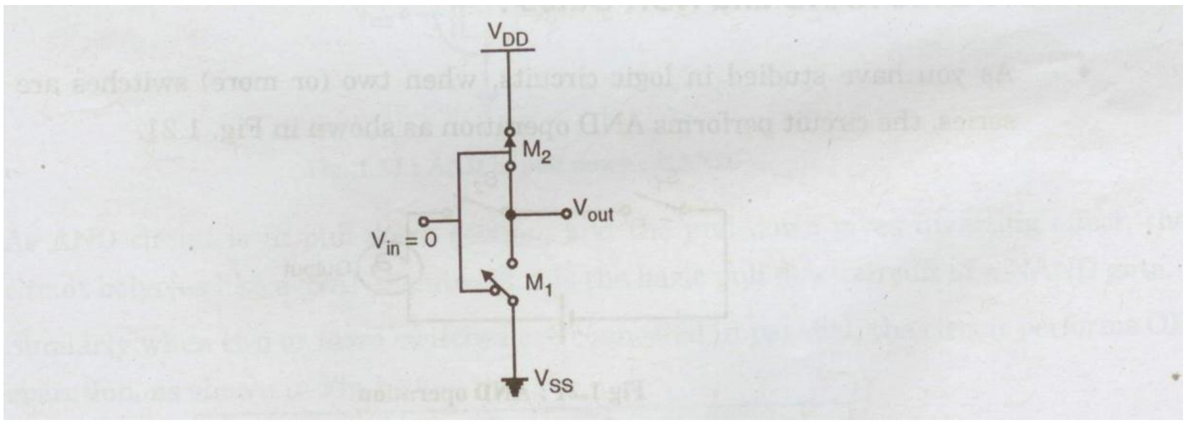
ANS: (diagram – 2 marks, explanation – 2 marks)



- For CMOS inverter, the driver is an nMOS enhancement type transistor, but the load is not nMOS, it is a pMOS transistor.
- The drain and gates of both the transistor are tied together to form the output and input of the inverter respectively. V_{DD} is applied at the source of pMOS and V_{SS} = ground is applied at the source of nMOS.
- The working of pMOS transistor is exactly the same as that of nMOS. The only difference is that pMOS requires negative voltage (i.e. negative V_{GS}) to form the p channel between its source and drain.

Working of CMOS inverter:

- We will begin with the case where input to the inverter is zero, i.e. when $V_{in} = 0$, gate of both the transistors M_1 (nMOS) and M_2 (pMOS) are at logic zero.
- In nMOS transistor, the input voltage $V_{GS,n}$ is smaller than the threshold voltage $V_{TH,n}$ of nMOS and M_1 is in cutoff. At the same time M_2 (pMOS) is ON, as input voltage i.e. $V_{GS,p.}$ is more negative than its threshold voltage $V_{TH,p.}$ We can verify this by substituting the value in equation.



- In case of pMOS by substituting values in equation (1.3)
  V_{GS} = Vin - V_{DD} . Therefore, $V_{GS} = 0 – 5 = - 5V$

  $V_{GS} = - 5V$
- We can visualize this circuit as two switches connected in series form V_{DD} to ground with upper switch closed and lower one open, as shown in figure. The output voltage, $V_{out} = V_{OH} = V_{DD}$, as output terminal gets connected to V_{DD}.
- At this instant we shall examine the second case where input to the inverter is high.
  i.e. $Vi_n = 5V$

As Vi_n is high gate of both the transistors are at logic 1 (i.e. 5V).

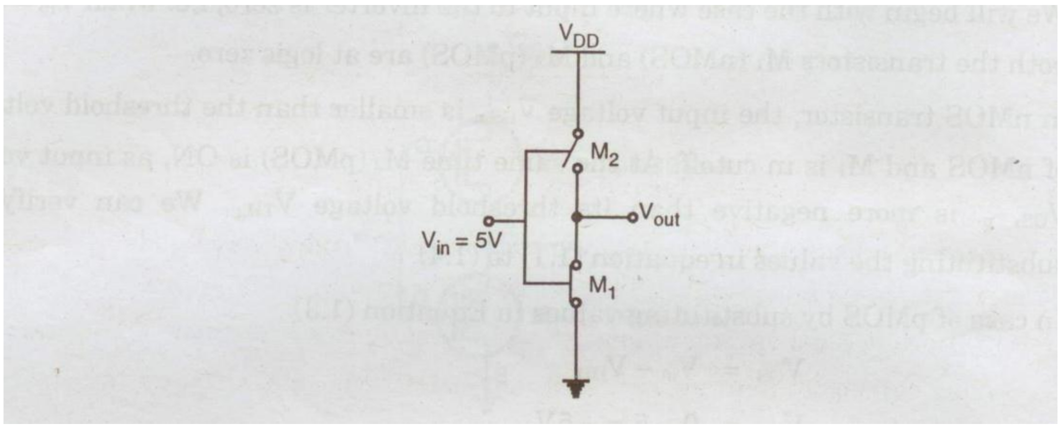
Transistor M_1 is ON as $V_{GS,n} = 5V$ which is greater than the threshold voltage of nMOS transistor.

- Substituting value in equation (1.3)
  $V_{GS,p} = V_{in} - V_{DD}$

  $V_{GS,p} = 5 – 5 = 0V$

  $V_{GS,p} = 0V$

  And transistor M_2 is in cutoff.
- Hence for $V_{in} = 5V$ the upper switch is open and the lower switch is closed as shown in figure. Thus output voltage $V_{out} = V_{OL} = 0V$
- We have discussed inverters with different types of pull up of logics. These pull up logics are also used in designing of NAND and NOR gates with basic pull down circuit using nMOS transistors.


V_{DD}
M_2
V_{out}
$V_{in} = 5V$
M_1