MODEL ANSWER

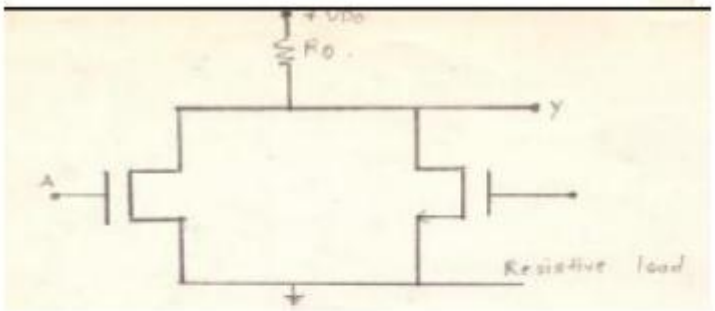
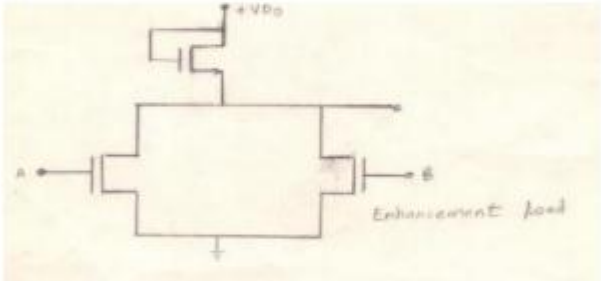
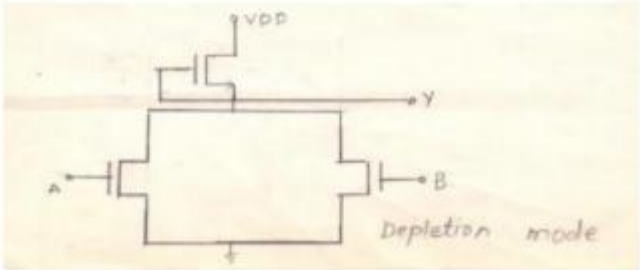
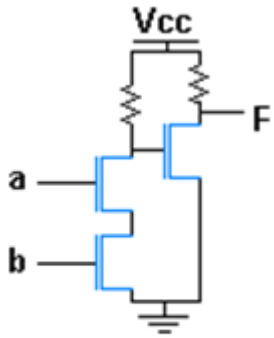
SUMMER– 19 EXAMINATION

Subject Title: Very Large Scale Integration Subject Code: 17659

Important Instructions to examiners:

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

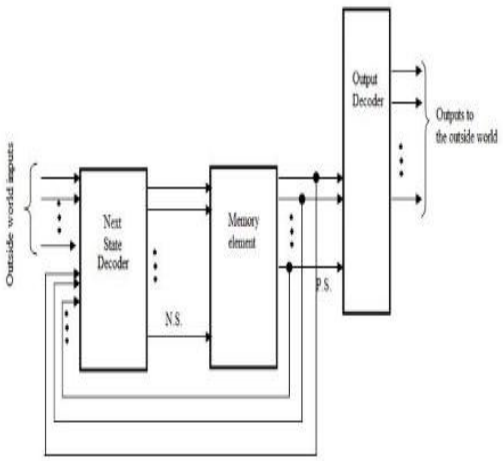
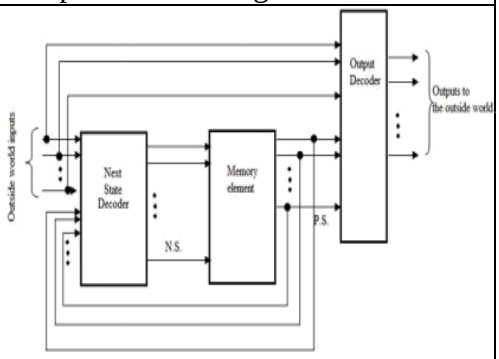
Q. No.	Sub Q.N.	Answer	Marking Scheme
Q.1		**Attempt any FIVE :**	**20M**
	a)	**Draw AND gate and NOR gate using NMOS.**	**4M**

	Ans:	**NOR gate using Resistive load**  **NOR gate using Active load**    **NMOS AND gate** **(Note: And Gate using active /passive also can be consider.)**	**2M NOR GATE** **2M AND GATE**
	b)	**Define: (i) Metastability (ii) Noise Margin**	**4M**
	Ans:	**<u>Metastability</u>:** In digital systems, when two asynchronous signals combine in such a way that their resulting output goes to an indeterminate state or unpredictable state. This state is known as metastable state. At the end of metastable state the output settles either 0 or 1. This whole process is known as Metastability.	**2M EACH**

Noise Margins: 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).

c) Compare Moore and Mealy Machine. — **4M**

Ans: — **EACH Point 1M (Any 4)**

Sr. No.	Moore Machine	Mealy Machine
1.	Moore machine is the sequential system where output depends only on present state. f (o/p) = f (P.S.)	Mealy machine is the sequential system where output depends on present input and state. f (o/p) = f (i/p, P.S.)
2.	It has more number of states than mealy machine.	It is smaller.
3.	It is faster.	It is slower than Moore machine.
4.	The output is delayed in a Moore machine. Output does not occur until the next state changes.	Output occurs in the same state by change in input.
5.	Simple to design.	Complicated to design.
6		

d) What is VHDL? Write two advantages of VHDL. — **4M**

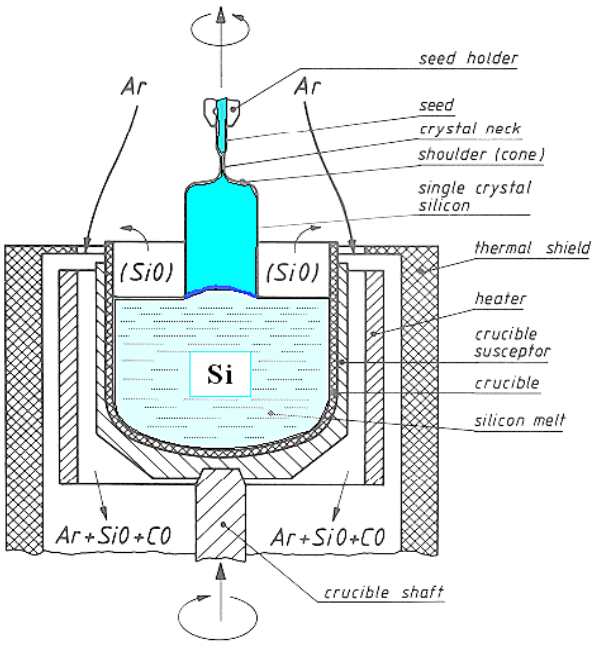
Ans: — **2M**

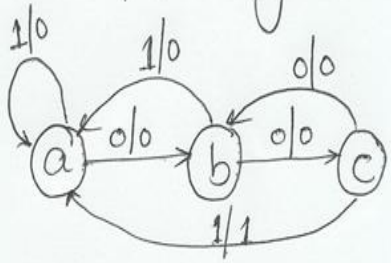
VHDL: VHDL is Hardware Descriptive Language that can be used to model digital system at many levels of abstraction, ranging from the algorithmic level to the gate level.

Advantages: — **½M Each (Any 4)**

- It is a concurrent language that is it can execute statements at same time in parallel as in hardware.
- It is a sequential language that is it can execute sequential statements one at a time in sequence.
- It supports synchronous and asynchronous timing models.
- Facilitates device independent of design portability.
- It supports design libraries.

		• It has well defined interface. • Behaviour specification for simulation purpose. • Test Benches can also be generated. • Digital modelling techniques supported. • It is not technology specific. • VHDL has powerful construct language, constructs such as else if, with select, case, when etc. • VHDL supports flexible design methodologies top-down, bottom-up or mixed. • Strongly typed language: • Dealing with signed and unsigned numbers is natural, and there's less chance of making a precision mistake or assigning a 16-bit signal to a 4-bit signal. • Ability to define custom types: • Record types: Define multiple signals into one type. • Natural coding style for asynchronous resets. • Logical statement (like case and if/then) endings are clearly marked.	
	e)	**Explain: (i) Sensitivity list. (ii) Wait statement**	**4M**
	Ans:	**i) Sensitivity List** 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. Ex. F<=a and b; A and b are in the sensitivity list of f. the statement will execute only if one of these will change. Ex. Proc ess(clk, RST) The process is sensitive to RST and clk signal i.e. an event on any of these signals will cause the process toresume **ii) Wait Statement** – Wait statement suspends the execution of event or procedure until some conditions are met. If no condition is given, the process will never be reactivated again. wait on [sensitivity list]; eg. wait on clk; wait until [condition]; wait until clk="1" wait for [time out expression]; wait for 20 ns;	**2M Each**

	f)	**Explain the wafer processing with C-Z method.**	**4M**
	Ans:	**Diagram:** 	**2M**
		It consists of Quartz crucible, which is surrounded by a graphite radiator. The graphite is heated by radio frequency induction heating and temperature maintained a few degrees above the melting point of silicon (approx. 14250C), the atmosphere just above the polysilicon melt is typically helium or orgon for freezing. A polycrystalline Si is melted in the crucible and controlled amount of impurities (p type or n type) are added to the melt to provide the crystal with required electrical properties. After the seed (single crystal silicon piece) is dipped into the melt, the seed is gradually withdrawn vertically from the melt while simultaneously being rotated. The molten polycrystalline silicon melts the tip of the seed and it is withdrawn, refreezing occurs. As the melt freezes, it assumes the single crystal form of the seed. This process is continued until the melt is consumed. The diameter of the ingot (rod of silicon) is determined by the seed withdrawn rate and seed rotation rate. The produced crystalline silicon rod is then slicing into wafers using cutting tools like diamond blades. Following slicing at least one face of the wafer is polished to flat scratch free mirror finish surface.	**2M**
	g)	**Design Mealy sequence detector circuit for detecting sequence of “001”**	**4M**

	Ans:	Step : 1 : State Diagram  Step 2 : State Table :	**4M**

Present State	Next State x=0	Next State x=1	Output (Z) x=0	Output (Z) x=1
a	b	a	0	0
b	c	a	0	0
c	b	a	0	1

Step 3 : Modified Table

let a = 00, b = 01 & c = 10

Therefore,

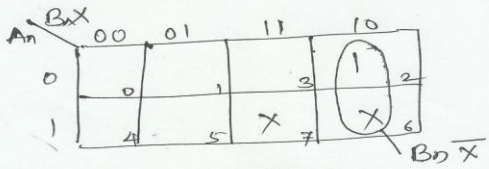
Present State		Next state x=0		Next state x=1		Output (Z) x=0	Output (Z) x=1
A	B	A	B	A	B		
0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0
1	0	0	1	0	0	0	1

Step 4 : Excitation table :

Present state		input	Next state		output	flip flop input	
A_n	B_n	X	A_{n+1}	B_{n+1}	Z	D_A	D_B
0	0	0	0	1	0	0	1
0	0	1	0	0	0	0	0
0	1	0	1	0	0	1	0
0	1	1	0	0	0	0	0
1	0	0	0	1	0	0	1
1	0	1	0	0	1	0	0
1	1	0	x	x	x	x	x
1	1	1	x	x	x	x	x

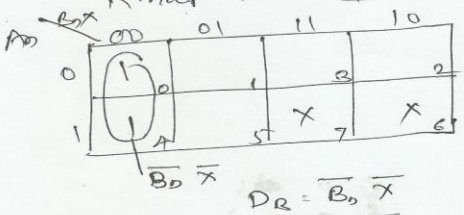
Step 5 : K maps :

Kmap for DA



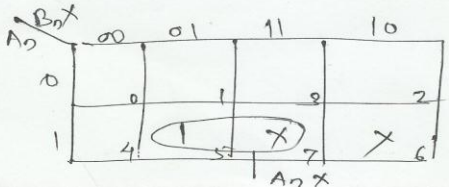
$D_A = B_n\overline{X}$

Kmap for D_B

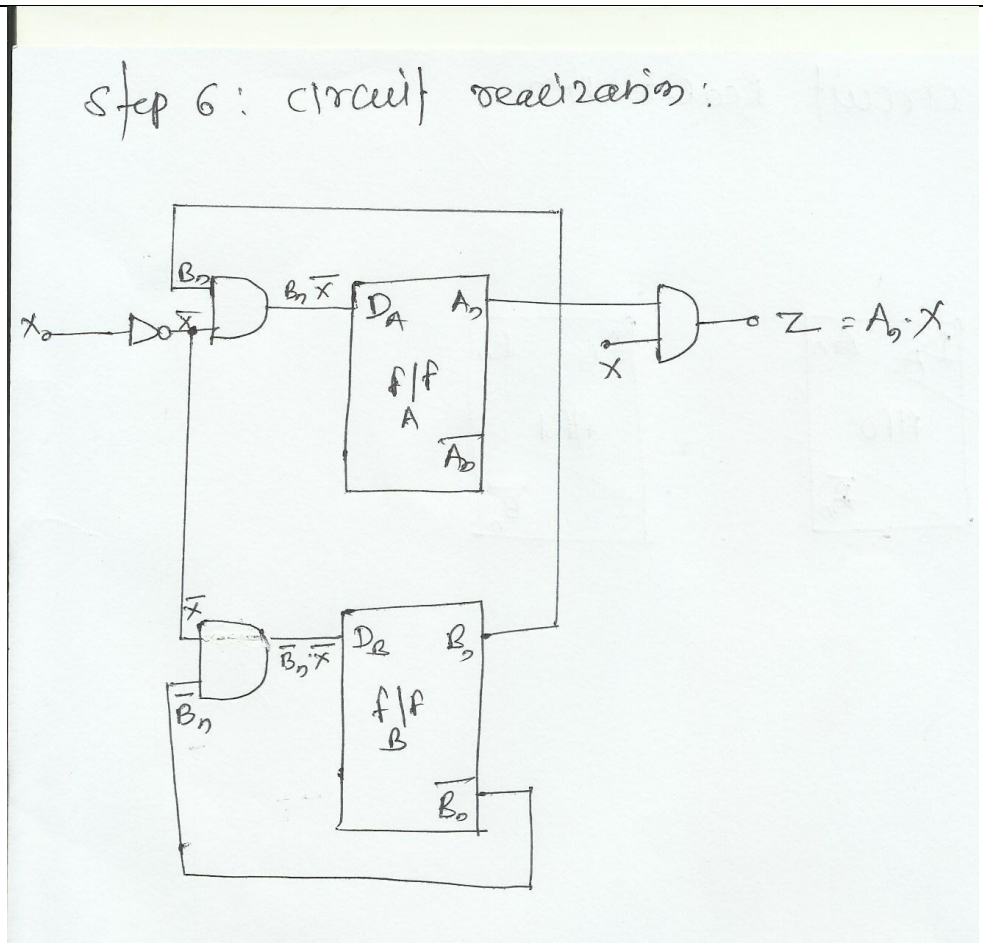


$D_B = \overline{B_n}\,\overline{X}$

Kmap for Z,



$Z = A_n X$



Note : Any F/F can be used.

Q 2		**Attempt any FOUR :**	**16M**
	a)	**Compare synchronous & asynchronous sequential circuits.**	**4M**
	Ans:		**(Any4) 1M each**

Parameter	**Asynchronous**	**Synchronous**
Definition	Asynchronous is wherein all the flip-flops within the counter do not change state simultaneously. This is because all the flip-flops are not clocked simultaneously.	Synchronous is wherein all the flip-flops within the counter change state simultaneously. This is because all the flip-flops are clocked simultaneously.
Clock required	It does not use a clock to trigger all outputs.	It uses a clock pulse to trigger all outputs
o/p affected by	The state of circuit can change immediately when an input change occurs	A change of state occurs only in response to a synchronizing clock pulse.

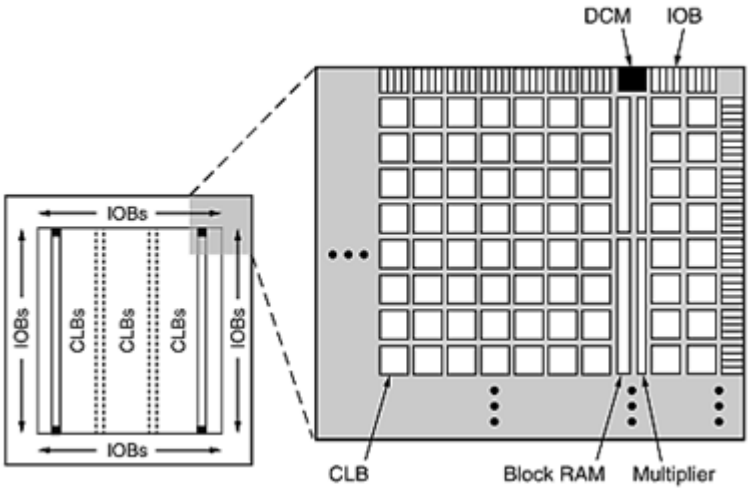
Memory element	Either latches (unclocked FF) or logic gates	Clocked FF
Design	These circuits are difficult to design	These circuits are easy to design.
Speed	They are faster	They are slower

OR

SR. NO.	ASYNCHRONOUS SEQUENTIAL CIRCUITS	SYNCHRONOUS SEQUENTIAL CIRCUITS
1	Output can be changed at any instant of time by changing the input	Output changes at discrete interval of time
2	The status of memory element will change any time as soon as input is changed. It does not use a clock	The status of memory is affected only at the active edge of clock, if input is changed. It uses a clock pulse.
3	These circuits are easy to design.	These circuits are difficult to design.
4	They are comparatively faster as no clock is used .	They are slower as clock is involved.
5	Asynchronous is wherein all the flip-flops within the counter do not change state simultaneously. This is because all the flip-flops are not clocked simultaneously.	Synchronous is wherein all the flip-flops within the counter change state simultaneously. This is because all the flip-flops are clocked simultaneously.

b) **Draw the architecture of sparatan-3 FPGA series. Explain any two blocks.** **4M**

Ans: **Architecture :** **2M**



		The Spartan-3E 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.	**Explain any 2 blocks 1M Each**
	c)	**Explain: (i) Flattening (ii) Structuring**	**4M**

	Ans:	**Flattening:**
		The process of converting the optimized 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;
		after flattering
		a = (x or (y and z)) and (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.
			2M
			2M

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.

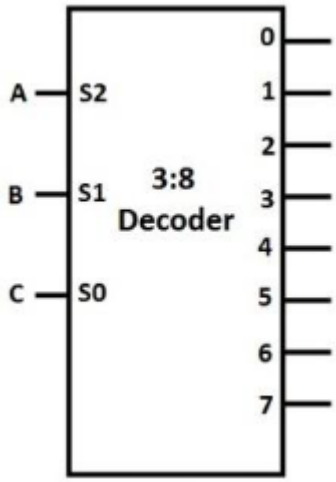
Example if [(a AND b) OR c] occurs ten times, then the tool may assign it a variable „x" and then x is used everywhere.

Finally the sub functions are substituted into the original equations. Comparing to the logic before structuring, the resulting area is reduced.

d) Write VHDL program for 3 : 8 decoder. **4M**

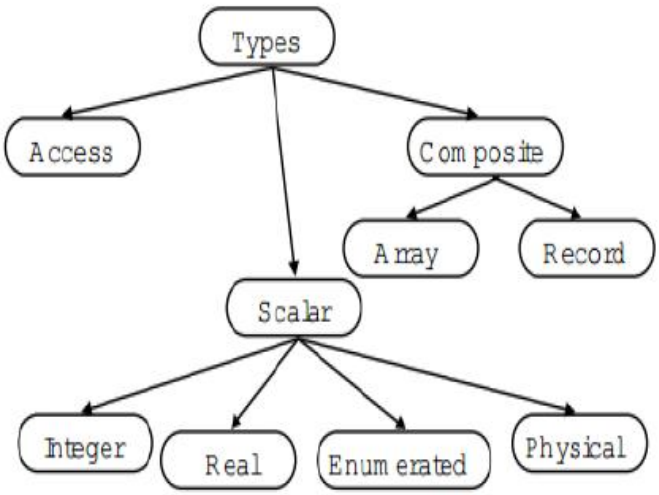
Ans: (Note Diagram is Optional)

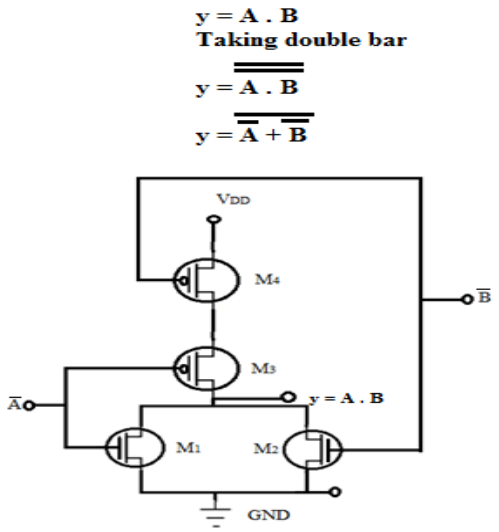
Entity 1M



Architecture 3M

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity decoder3_8 is
    port(
        din : in STD_LOGIC_VECTOR(2 downto 0);
        dout : out STD_LOGIC_VECTOR(7 downto 0)
        );
end decoder3_8;
architecture decoder3_8_arc of decoder3_8 is
begin
   dout<= ("10000000") when (din="000") else
          ("01000000") when (din="001") else
          ("00100000") when (din="010") else
          ("00010000") when (din="011") else
          ("00001000") when (din="100") else
          ("00000100") when (din="101") else
          ("00000010") when (din="110") else
          ("00000001") ;
```

		end decoder3_8_arc **(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)**	
	e)	**List and explain data types used in VHDL.**	**4M**
	Ans:	 There are mainly two types: 1. Scalar 2. Composite **Scalar Types:** The scalar data types describe objects that can hold at the most one value at a time. There are four different kinds of scalar types. These types are: 1. **Integer:** An integer type defines a type who's set of values fall within a specified integer type. Integer is the only predefined integer type of the language. The range of the Integer number is from -2^{31}+1 to $+2^{31}$-1(-2,147,483,647 to 2,147,483,647). **E.g. 568** 2. **Real**: A floating point type has a set of values in a given range of real numbers. The only predefined floating point type is REAL. The range of REAL is from -1.0E38 to +1.0E38 **e.g. 16.26** 3. **Enumerated:** Defines the set of user defined values consisting of identifiers and character literals	**List 1M** **Explain 3M**

		This type values are represented by enumeration literals (either identifiers or character literals). For example consider the following enumeration type declaration. Type CAR_STATE is (STOP, SLOW, MEDIUM, FAST); The predefined enumeration types of the language are CHARACTER, BIT, BOOLEAN, SEVERITY_LEVEL, FILE_OPEN_KIND and FILE_OPEN_STATUS. **4. Physical:** A physical types are used to represent physical quantities such as length, voltage, time and current. **Composite data types:** A composite type represents a collection of values. There are two composite types: **Array:** Contain many elements of same type. **Record:** Contain elements of different types	
	f)	**Draw and explain CMOS AND gate.**	**4M**
	Ans:	**Diagram:**  **Case 1:** When both the inputs A and B are at logic zero. As inputs A' and B' are logic 1 both the n-MOS transistors M_1 and M_2 are at logic 1, they are ON. And both the switches are closed. At the same time, p-MOS transistors M_3 and M_4 are OFF and both the switches are open. As a result of which, the output is shorted to ground. Hence output y = 0. (Logic 0) **Case 2:** When any one of the inputs is at logic 1, Say A = 0 and B =1 As input A' =1 and B'=0, transistor M_1 is ON and transistor M_3 is in cutoff and transistor	**2M** **Explain** **OR** **Truth table**

M_2 is OFF and M_4 is in ON. **2M**

Case 3: When A=1 and B=0
As input A' =0 and B'=1 transistor M_1 is OFF and transistor M_3 is ONand transistor M_2 is ON and M_4 is in OFF.
In the above two conditions, the output is shorted to ground. Hence output y = 0. (Logic 0)
Case 4: When both the inputs A and B are at logic 1.
As input A' =0 and B'=0 transistor M_1 is OFF and transistor M_3 is ON and transistor M_2 is OFF and M_4 is in ON.
In the above two conditions, the output is shorted to VDD. Hence output y = 1. (Logic 1)

The working of CMOS NAND gate is summarized in truth table as below-The working of CMOS NAND gate is summarized in truth table as below-

Input		CMOS				Output
A	B	M1	M2	M3	M4	
0	0	ON	ON	OFF	OFF	0
0	1	ON	OFF	OFF	ON	0
1	0	OFF	ON	ON	OFF	0
1	1	OFF	OFF	ON	ON	1

Q.3 **Attempt any FOUR :** **16M**

a) **List and explain features of CPLD.** **4M**

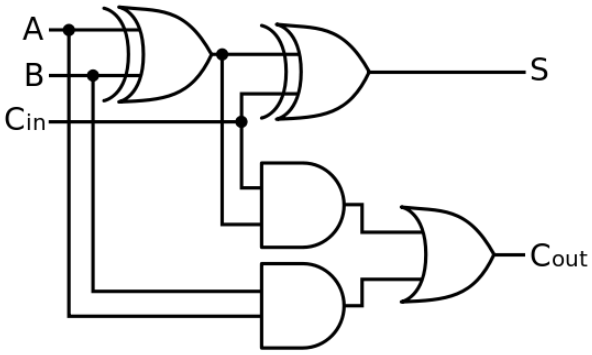
Ans: **4M**

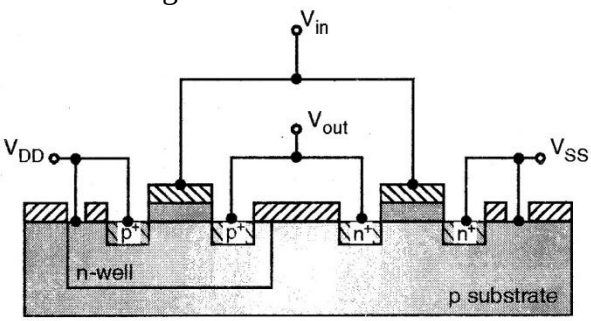
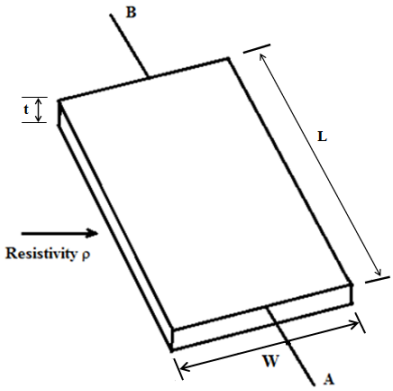
1. Product terms generated in programmable macrocells.
2. Typically one dedicated flip-flop per macrocell
3. Many macrocells per logic-block
4. Typically all logic-blocks identica
5. Minimum two logic-blocks per device
6. Routing between logic-blocks via global switch matrix

OR

1. High-performance Fully CMOS, Electrically-erasable Complex Programmable Logic Device.
2. In-System Programming (ISP) Supported
3. Flexible Logic Macrocell
4. Fully Green (RoHS Compliant)
5. 1 Static Current
6. Power Saving Option During Operation
7. Programmable Pin-keeper Option on Inputs and I/Os
8. Programmable Schmitt Trigger Option on Input and I/O Pins
9. Programmable Input and I/O Pull-up Option (per Pin)
10. Unused Pins Can Be Configured as Ground (Optional)
11. Available in Commercial and Industrial Temperature Ranges
12. Advanced Digital CMOS Technology
13. Security Fuse Feature
14. Hot-Socketing Supported

	b)	**State and explain delta delay.**	**4M**
	Ans:	All digital circuit elements have a delay (propagation delay) which is very small in terms of nano sec. This nano sec delta delay will have little impact while writing the VHDL code. But for circuit realization this delay must be incorporated. The physical circuit always has finite delay. The ordering of zero delay events is handled with a fictitious unit called delta time. Delta time represents the executionof a simulation cycle without advancing Simulation time. The simulator models zero-delay events using delta time.Events scheduled at the same time are simulated in specific order during a delta time step.Related logic is then re-simulated to propagate the effects for another delta timestep. Delta time steps continue until there is no activity for the same instant of simulated time. **OR** **<u>Delta Delay:</u>** The delta delay is introduced to achieve concurrency and order independence. The simulator freezes simulation time until all scheduled assignments in current simulation time is finished and there are no more events on the sensitivity list. Thus real and simulation time are different. Several logic changes occur simultaneously in a circuit. But a simulator cannot process events concurrently. Hence time is frozen within the time. Events are processed and logic values are updated one after another till no more events take place. This is known as one simulation cycle. The real time that the simulator takes to execute one simulation cycle is known as delta delay for simulation delta with zero simulation time.	**4M**
	c)	**Write VHDL code for Full ADDER.**	**4M**

	Ans:	
		Library ieee;
		use ieee.std_logic_1164.all;
		
		entity full_adder is port(a,b,c:in bit; sum,carry:out bit);
		end full_adder;
		
		architecture data of full_adder is
		begin
		sum<= a xor b xor c;
		carry <= ((a and b) or (b and c) or (a and c));
			4M

		end data;	
	d)	**Explain N-well process with diagram.**	**4M**
	Ans:	**<u>N-Well process:</u>** The N-well CMOS circuits are getting more popular because of the lower substrate bias effect on transistor threshold voltage and lower parasitic capacitances associated with source and drain regions.  The fabrication steps are as follows: • Thick SiO_2 layer is grown on p-type silicon wafer. • 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. • Oxide in the n transistor region is removed and thin oxide layer is grown all over the surface to insulate gate and substrate. • The polysilicon is deposited and patterned on thin oxide regions 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. • Using n^+ mask and complementary n^+ mask, source and drain of both nMOS and pMOS transistors are formed one after another using respective diffusion processes. These same masks also include the V_{DD} and V_{SS} contacts. • The contacts are made using proper masking procedure and metal is deposited and patterned on the entire chip surface. • An overall passivation layer is formed and the openings for accessing bonding pads are defined.	**4M**
	e)	**Explain Resistance Fabrication.**	**4M**
	Ans:	**<u>Resistance Estimation:</u>** Consider a uniform slab of conducting material of resistivity ρ. Let W be the width, t the thickness and l is the length of the slab. Hence the resistance between A and B terminal:  	**4M**

$$R_{AB} = \frac{\rho L}{A} \text{ ohms}$$

Where A is the cross section area.

$$R_{AB} = \frac{\rho L}{t\ x\ W}$$

Consider the case in which L = W, which is square of resistive material base then:

$$R_{AB} = \frac{\rho}{t} = Rs$$

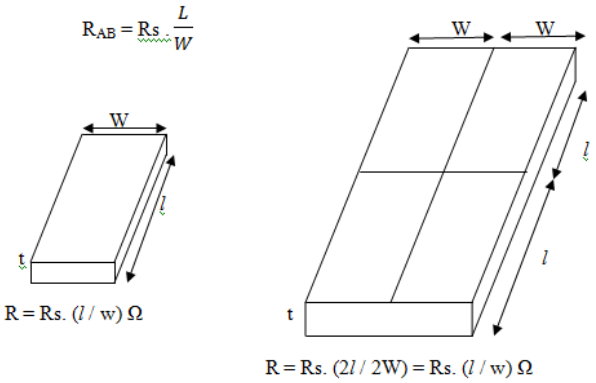
Where Rs= ohm per square or sheet resistance.

$$\therefore Rs = \frac{\rho}{t}$$

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

$$\text{Thus: } R_{AB} = Rs \times \frac{L}{W}$$

Thus to obtain the resistance of a conductor on a layer multiply by the sheet resistance Rs, by the ratio of length to width of the conductor.



For ex: resistances of the two shapes shown in figure are same as the length to width ratios of both the slabs are same, even though the sizes are different.

This channel has length L = 8λ and width W = 2λ

$$R = R_S\left(\frac{L}{W}\right)$$

$$\therefore R = R_S\left(\frac{8\lambda}{2\lambda}\right)$$

$$\therefore R = 4R_S$$

In other words there are 4 squares in series length wise. Hence the total resistance is 4 Rs.

	f)	**Design parity checker using Moore logic or Mealy logic.**	**4M**

	Ans:		**4M**

Parity checker:-

a) Serial I/p string

- Out = 1 if odd of 1's is input
- Out = 0 if even of 1's is input

b) State diagram (Moore machine)

c) State Transition Table

I/p	Present state	Next state	O/P
0	Even	Even	0
1	Even	odd	0
0	odd	odd	1
1	odd	even	1

d) State Assignment:-

Even = 0; odd = 1

e) State Encoding:-

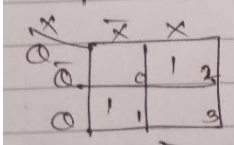
I/P	Present state	Next state	O/P
0	0	0	0
1	0	1	0
0	1	1	1
1	1	0	1

f) State Excitation Table:-

(X) I/p	(PS) Q	NS Q^+	D F/F I/P	O/P
0	0	0	0	0
1	0	1	1	0
0	1	1	1	1
1	1	0	0	1

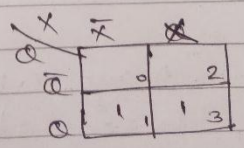
g) Kmap:-

(a) Dinput

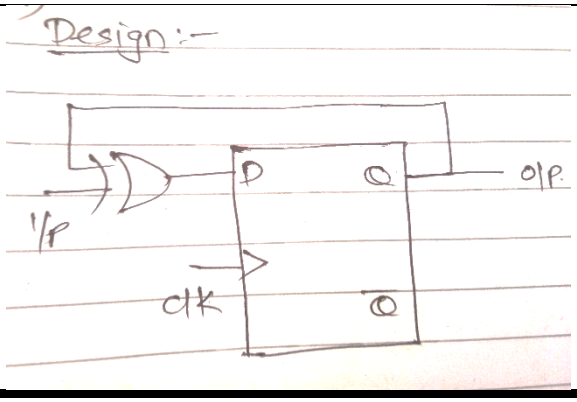


$D_{in} = X\bar{Q} + \bar{X}Q$

(b) O/P



$Y = Q$

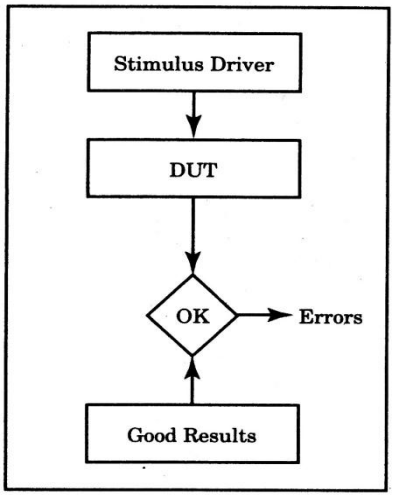


Q.4	**A)**	**Attempt any FOUR :**	**16M**
	a)	**Compare FPGA and CPLD.**	**4M**
	Ans:		**4M**

Sr. No	FPGA	CPLD
1.	It is field programmable gate array.	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
4.	Number of input and output pins on FPGA are less than CPLD.	Number of input and output pins on CPLD are more than FPGA.
5.	FPGA is suitable for designs with large number of blocks with few number of inputs.	CPLD is ideal for complex blocks with large number of inputs.
6.	FPGA based designs require more board space and layout is more complex.	CPLD board designs need less board space and layout is less complex.
7.	It is difficult to predict the speed performance of design.	Speed performance can be easily predicted.
8.	FPGA are available in wide density range.	CPLD contain fewer registers but have better performance.

	b)	**Explain cycle based and event based simulators.**	**4M**
	Ans:	**1. <u>Event based simulator:</u>**	**2M**

- 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.

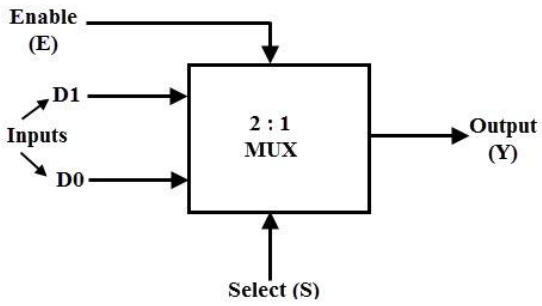
		2. <u>Cycle-based Simulator:</u> • Cycle-based simulation ignores intra—cycle state transitions. i.e. they check the Status of target signals periodically irrespective of any events. This can boost performance by 10 to 50 times compared to traditional event-driven simulators. • Cycle-based technology offers greater memory efficiency and faster simulation run-time than traditional pure event-based simulators. • Cycle-based simulators work best with synchronous design but give less timing accuracy with asynchronous design. • Signals are treated as variables. Functions such as AND. OR etc. are directly converted to program statements. • Signal level functions such as memory blocks, adders. Multiplier's 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.	**2M**
	c)	**Describe verification using Test Bench.**	**4M**
	Ans:	Test bench encapsulates the stimulus driver, known good results, and DUT and contains internal signals to make the proper connections. The stimulus driver drives the input into DUT which responds and produces results. Finally a compare function within the test bench compares the result from the DUT against those known good results and reports any errors  <u>Typical Test Bench Format is:</u>	**4M**

```
entity TEST_BENCH is
end;

architecture TB_BEHAVIOR of TEST_BENCH is
   component ENTITY_UNDER_TEST
      port ( list-of-ports-their-types-and-modes );
   end component;
   Local-signal-declarations ;
begin
   Generate-waveforms-using-behavioral-constructs ;
   Apply-to-entity-under-test ;
   EUT : ENTITY_UNDER_TEST port map ( port-associations ) ;
   Monitor-values-and-compare-with-expected-values ;
end TB_BEHAVIOR;
```

d) Write VHDL code for 2:1 MUX using if….. else statements. **4M**

Ans: **4M**

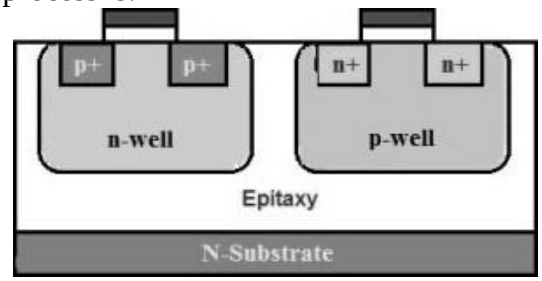


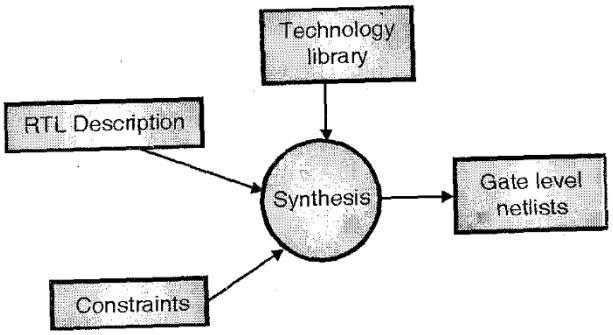

```
library ieee;
use ieee.std_logic_1164.all;

entity mux2to1 is
  port (w0, w1, s : in std_logic;
     f : out std_logic);
end mux2to1;

architecture behaviour of mux2to1 is
begin
  process (w0, w1, s)
  begin
    if s = '0' then
      f <= w0;
    else
      f <= w1;
    end if;
  end process;
end behaviour;
```

e) Explain Twin-Tab Process in CMOS fabrication with diagram. **4M**

	Ans:	In this process the substrate can be of any type. Consider n type silicon substrate. The twin tub fabrication process is:  1. The process is carried out on N type silicon substrate with lower doping or higher resistivity so that the lesser current flows through the substrate. On this, the n^+ Si substrate is grown further i.e. epitaxial layer of required thickness is grown. 2. SiO_2 layer is grown all over the surface and the areas of P well and N well are defined. P well is diffused by masking N well area and N well is diffused by masking P well area. 3. A thin layer of SiO_2 thin ox is deposited all over the surface. Using masking and etching process unrequired thin ox is removed. The thin ox is required only on gate areas of both the transistors. 4. The polysilicon is deposited all over the surface and using a mask it is removed from areas other than the gate area. 5. Then the P well is covered with a photoresist mask and p^+ diffusion is carried out to form the source and drain of pMOS transistor. 6. Now the N well is covered with a photoresist mask and n^+ diffusion is carried out to form the source and drain of nMOS transistor. 7. The thick layer of SiO_2 is grown all over the surface for isolation. This SiO_2 layer is etched off to expose all the terminals. The metal is deposited and patterned all over the wafer surface so that it makes contact with source, drain and gate terminals.	**4M**
	f)	**Explain HDL design flow for synthesis.**	**4M**
	Ans:	Synthesis = Translation + Optimization. Synthesis is an automatic method of converting higher level of abstraction to lower level of abstraction. i.e. • The process that converts user, hardware description into structural logic description. Synthesis is a means of converting HDL into real world hardware. It generates a gate level net list for the target technology. The synthesis tool converts register transfer level (RTL) description to gate level netlist. These gate level netlists consist of interconnected gate level macrocells. • The inputs to the synthesis process are RTL (register transfer level) VHDL description, circuit constraints and attributes for the design, and a technology library.	**4M**

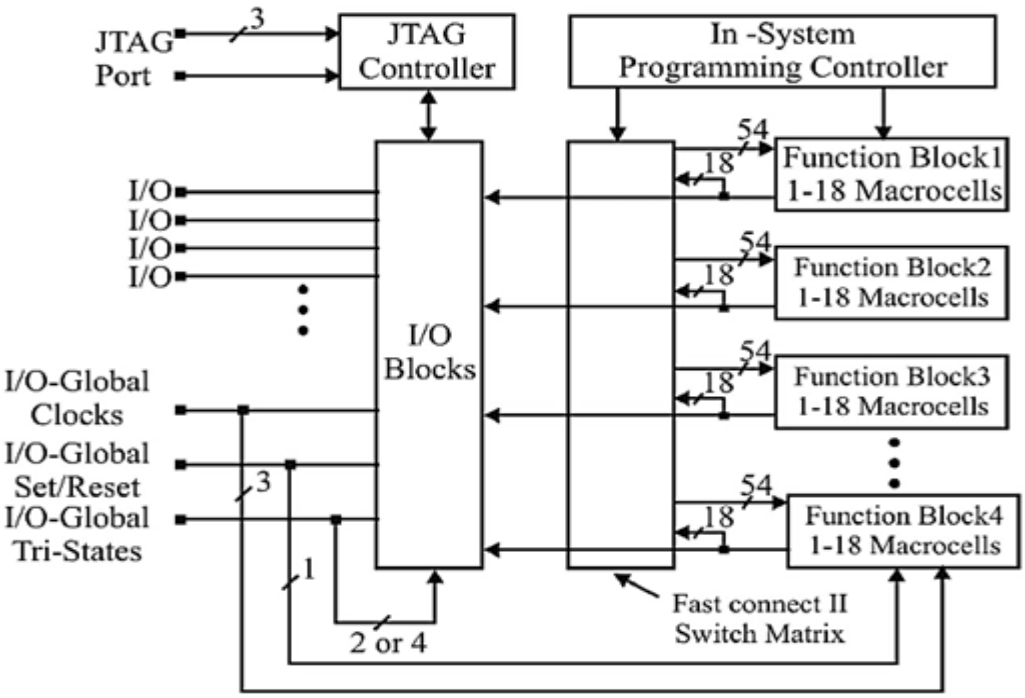


- The synthesis process produces an optimized gate level net list from all these inputs. The translation from RTL description to Boolean equivalent description is usually not user controllable.

The intermediate form that is generated is a format that is optimized for a particular tool and may not even be viewable by the user. All the conditional signal assignments and selected signal assignment statements are converted to their boolean equivalent in this intermediate form. The optimization process takes an unoptimized Boolean description and converts it to an optimized Boolean description. For this it uses number of algorithm and rules. This process aims to improve structure of Boolean equations by applying rules of boolean algebra. This removes the redundant logic and reduces the area requirement

Q.5		**Attempt any FOUR :**	**16M**
	a)	**Explain: (i) Sensitivity list (ii) Zero modeling**	**4M**
	Ans:	**Sensitivity List :-** This list defines the signals that cause the statements inside the process statement to execute whenever one or more elements of the list change value. Sensitivity list is the list of the signals that will cause the process to execute. 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. Ex. Process (CLK, RST) The process is sensitive to RST and CLK signal i.e. an event on any of these signals will cause the process to resume. **Zero Modeling:-** All digital circuit elements have a delay (propagation delay) which is very small in terms of nano sec. This nano sec delta delay will have little impact while writing the VHDL code. But for circuit realization this delay must be incorporated. The physical circuit always has finite delay. In VHDL zero delay circuits and designs that depend on zero delay components can never be build. Simulation deltas are used to order some types of events during simulation. Specifically zero delay events must be ordered to produce consistent results. If they are not properly ordered results can vary between different simulation runs.	**2M Each**

	b)	**Draw the architectures of Xilinx 9500 family CPLD. Explain any two blocks.**	**4M**
	Ans:	**Architecture 9500 :**	**2M**



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

Explain (Any2) blocks 2M Each

from the I/O block to the Functional Block.

c) What is instantiation in VHDL? Write one example. **4M**

Ans: **Instantiation** :

The instantiation means the precompiled entity architecture component is declared in another VHDL entity program. A component instantiated in a structural description is declared using component declaration. A component declaration declares the name and the interface of a component. **2M**

Example

Note: Any suitable example. NAND gate using AND and NOT

Any relevant example 2M Each

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity INVE is
   Port ( X : in  STD_LOGIC;
          Z : out  STD_LOGIC);
end INVE;

architecture Behavioral of INVE is

begin
 z <= not x;
end Behavioral;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity NANDG is
   Port ( a : in  STD_LOGIC;
          b : in  STD_LOGIC;
          y : out  STD_LOGIC);
end NANDG;
architecture Behavioral of NANDG is
 component INVE
  port  ( X : in  STD_LOGIC;
          Z : out  STD_LOGIC);
end component;
 signal w : std_logic;
begin
  w <= a and b;
a0 :INVE
port map ( w, y);
end Behavioral;
```

d) List and explain different types of operators in VHDL. **4M**

Ans: **Logical Operators:** The logical operators are defined for BIT and BOOLEAN types, as well as for one-dimensional arrays containing the elements of BIT and BOOLEAN. The logical operators are: AND, OR, NAND, NOR, XOR, XNOR, NOT **4M**

Relational operators: The relational operators are used to check the conditions. Both operands must be of the same type, and the result received is always of the Boolean type. The relational operators are:

=	Equality
/=	Inequality
<	Ordering „less than"
<=	Ordering „less than or equal"
>	Ordering „greater than"
>=	Ordering „greater than or equal"

Shift Operators: These are defined for one dimensional array with elements of the type bit and Boolean. The various shift operators are:

sll	Shift left logical
srl	Shift right logical
sla	Shift left arithmetic
sra	Shift right arithmetic
rol	Rotate left logical
ror	Rotate right logical

Adding Operators: The three adding operators are +, - and &. The + and – operators must be of same numeric type and the result is of the same numeric type. Whereas & (concatenation) operator can be one dimensional array type as an element type.

Multiplying operators: The multiplication and division operators are predefined for both operands being of the same integer or real type. The result is also of the same type.
The rem (reminder) and mod (modulus) operators operates on operands of integer types, and the result is also of the same type. The result of a rem operation has the sign of its first operand and is defined as:
A **rem** B = A-(A/B)*B
The result of mod operator has the sign of the second operand and is defined as:
A **mod** B= A-B*N

Miscellaneous operators: The two miscellaneous operators are **abs** [absolute] and ** [exponential]. The abs is defined for any numeric type and ** is defined for integer or floating point number.
Unary operators: These are sign operators and there are two sign operators positive and negative.

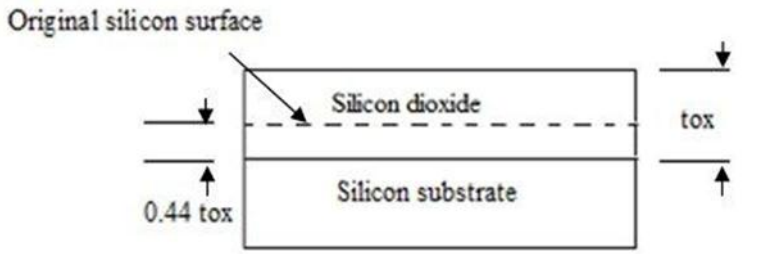
Operator Type							
logical	and	or	nand	nor	xor	xnor	not
relational	=	/=	<	<=	>	>=	
shift	sll	srl	sla	sra	rol	ror	
addition	+	-					
unary	+	-					
multiplying	*	/	mod	rem			
other	**	abs	&				

	e)	**State and explain efficient coding styles.**	**4M**
	Ans:	• There may be more than one method to model a particular design part but only a few	**4M**

		would yield better performance. • The essence of VHDL coding lies in understanding which style yields the ultimate performance under the given set of specifications. • The key to higher performance is to avoid writing code that needlessly creates additional work for the HDL compiler and synthesizer, which, in turn, generates designs with greater number of gates. • o Basically, any coding style that gives the HDL simulator information about the design that cannot be passed onto the synthesis tool is a bad coding style.	
	f)	**Write the VHDL code for 4-bit adder without instantiation.**	**4M**

	Ans:	library IEEE;
		use IEEE.STD_LOGIC_1164.ALL;
		use IEEE.STD_LOGIC_arith.ALL;
		use IEEE.STD_LOGIC_unsigned.ALL;
		entity Add is
		Port (X : in STD_LOGIC_VECTOR (3 downto 0);
		Y : in STD_LOGIC_VECTOR (3 downto 0);
		Sum : out STD_LOGIC_VECTOR (3 downto 0);
		Carry : out STD_LOGIC);
		end Add;
		architecture Behavioral of Add is
		signal s1,s2 : std_logic_vector (4 downto 0);
		signal temp : std_logic_vector (4 downto 0);
		begin
		s1 <= '0' & X;
		s2 <= '0' & Y;
		temp <= s1 + s2;
		Sum <= temp(3 downto 0);
		Carry<= temp(4);
		end Behavioral;
		(NOTE : Any other logic can be used for program)
			Entity 1M
			
			Architecture 3M

	Q.6	**Attempt any FOUR :**	**16M**
	a)	**Explain the following process :** **(i) Oxidation Process (ii) Diffusion Process**	**4M**
	Ans:	**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. Therefore the reliable manufacture of SiO2 is extremely important.	**2M Each**



Oxidation of silicon is achieved by heating silicon wafers in an oxidizing atmosphere such as oxygen or water vapour.

Two common methods of oxidation are:

Wet Oxidation: When oxidizing atmosphere contains water vapour the temperature is usually between9000C and 10000C. This is rapid process.

Dry Oxidation: When the oxidizing atmosphere is pure oxygen temperature are in the region of 12000C, toachieve an acceptable growth rate.

The oxidation process consumes silicon. Since SiO2 has approximately twice the volume of silicon, the SiO2 layer grows almost equally in both vertical directions. This layer of oxide (SiO2) is called as field oxide (FOX).

Doping of Silicon

Process of adding impurities to silicon.

N type = Phosphorus

P type = Boron

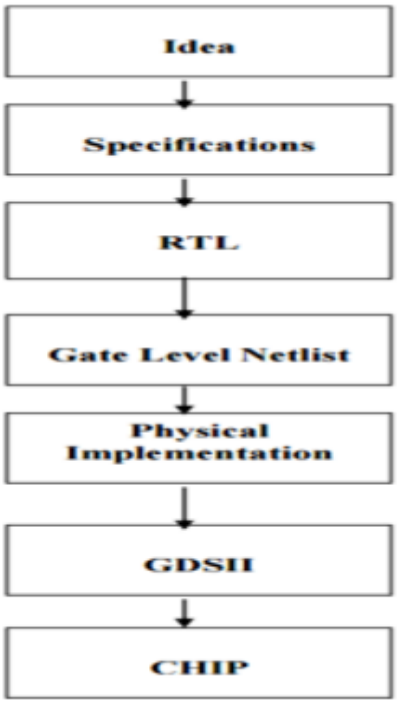
Diffusion:

Diffusion is a process of introducing controlled amount of dopants into semiconductors. Using diffusion conductivity of silicon is being altered by producing either n type or p type region.

Diffusion of impurity atoms into silicon crystal takes place only at elevated temperature, typically 900 to 11000C. Impurity atoms are introduced onto the surface of silicon wafer and diffuse into the lattice because of their tendency to move from regions of high to low concentration.

Diffusion is used to form base, emitters and resistors in bipolar device technology, to form source and drain regions and to dope polysilicon in MOS device technology.

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.

	b)	**Draw ASIC design flow.**	**4M**
	Ans:		**4M**
	c)	**Compare software & hardware description language.**	**4M**

Ans:

Sr. No.	Software Language	Hardware Descripting Language
1	It is High Level Language	It is used for implementing hardware circuit.
2	It handles sequential instruction.	VHDL allows both sequential and concurrent executions.
3	It can be written with our logical or arithmetic thinking.	VHDL programmer needs knowledge of Hardware circuit.
4	These programs ran on computer with powerful processor with high speed so easy to implement image processing algorithm.	Difficult to implement image processing algorithm in VHDL
5	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.
6	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.
7	In software language, the statements are evaluated sequentially.	In VHDL, concurrent statements are defined to take care of concurrency hardware.

Any 4 points 1M Each

d) Write VHDL program for 4:1 MUX using case statement. **4M**

Ans:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity multiplexer4_1 is
port (
     i0 : in std_logic;
     i1 : in std_logic;
     i2 : in std_logic;
     i3 : in std_logic;
    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 <= i0;
  when "01" => y <= i1;
  when "10" => y <= i2;

when "11" => y <= i3; -------------optional
  when others => y <= 'Z';  ---------------optional  or  y <= i3;
```

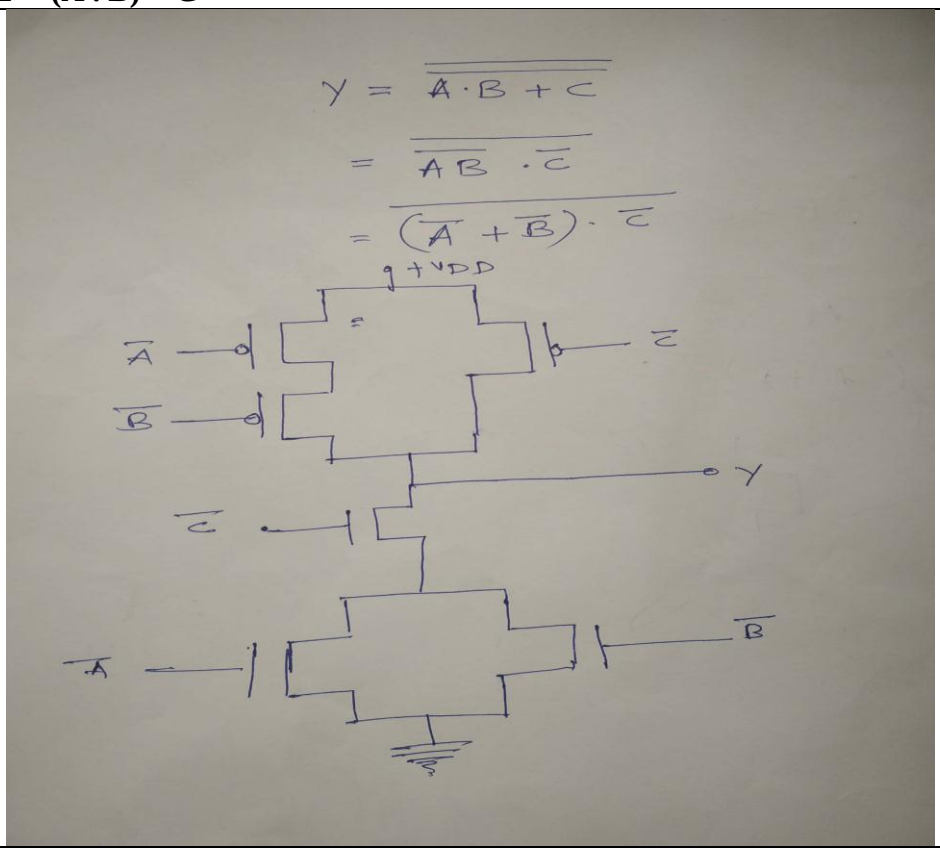
Entity 1M

Architecture 3M

		end case; **end process**; **end** Behavioral;	
	e)	**Explain Entity and Architecture with suitable example.**	**4M**

	Ans:	**Entity declaration:**
		
		• Entity is the description of inputs and outputs of the design. An entity is the most basic building block in the design. A design can have more than one entity block.
		• The entity statement declares the design name. Then it defines input output parameters andports of the design entity.
		Syntax:
		
		entity entity _ name is
		
		Port declaration.
		
		end entity_name
		
		Architecture:
		
		• All entities that are declared have an architecture associated with it. Architecture describes the behavior of the entity. An entity can have multiple architectures.
		• Architecture assigned to an entity describes internal relationship between input and output of the entity. First part of the architecture may contain declaration of types, signals, constants, subprograms etc.
		
		Syntax:
		
		architecture architecture _name of entity_ name
		
		Architecture_ declaration_ name;
		
		begin
		
		Statement;
		
		end architecture_ name;
		
		Example -
		
		library IEEE;
		use IEEE.STD_LOGIC_1164.**ALL**;
		
		entity multiplexer4_1 **is**
		port (
		i0 : **in** std_logic;
		i1 : **in** std_logic;
		i2 : **in** std_logic;
		i3 : **in** std_logic;
		sel : **in** std_logic_vector(1 **downto** 0);
		y : **out** std_logic);
			Entity 2M
			
			Architecture 2M

		end multiplexer4_1;
		
		architecture Behavioral **of** multiplexer4_1 **is**
		begin
		process(i0,i1,i2,i3,sel)
		begin
		case sel **is**
		  when "00" => y <= i0;
		  when "01" => y <= i1;
		  when "10" => y <= i2;
		  when others => y <= i3;
		end case;
		end process;
		end Behavioral;
		
		(NOTE: Any relevant example can be considered (complete program not required))	

	f)	**Design the following function using CMOS:** **Y = (A . B) + C**	**4M**
	Ans:		**4M**