MODEL ANSWER

SUMMER– 17 EXAMINATION

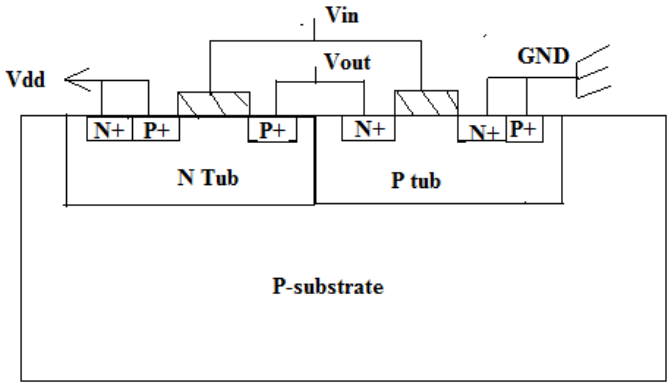
Subject Title: Very Large Scale Integration Subject Code: 17659

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 judgement 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 THREE :**	**12M**
	i)	**Define :** **1)Asynchronous sequential circuit** **2)Noise margin** **3)Fan out** **4)Skew.**	**4M**
	Ans:	**<u>Asynchronous Sequential Circuit:</u>** 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. **<u>Noise Margins</u>:** 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). **<u>Fan-Out:</u>** It is the maximum number of load gates that can be connected at output without loading with same IC family and by maintaining its output within the specified limit, **<u>Skew (Clock Skew)</u>:** skew is defined as "the magnitude of the time difference between two events that ideally would occur simultaneously.	**1M** **1M** **1M** **1M**
	ii)	**Write any two pro's and any two con's of VHDL.**	**4M**
	Ans:	**<u>Pros :</u>** • 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:	**(1M- Any two pros and cons)**

		• A VHDL state machine can be coded naturally using the actual state names (e.g. wait, acknowledge, transmit, receive, etc.), not binary state numbers (e.g. 00, 01, 10, 11). • Record types: • Define multiple signals into one type. • Natural coding style for asynchronous resets. • Easily reverse bit order of a word. • Logical statement (like case and if/then) endings are clearly marked. **Cons :** • Extremely verbose coding: • VHDL modules must be defined by a prototype and declared before they're used, causing you to change code in at least 3 places if you want to make a change to the interface. • The use of the keyword "downto" in every bit vector definition is tedious. • Sensitivity lists: Missing a single signal in the sensitivity list can cause catastrophic differences between simulation and synthesis. • Each process must have a sensitivity list that may sometimes be very long. **Type conversions:** • Signal types that are clearly related (e.g. std_logic and std_logic_vector) cannot be simply used together and must be converted to another type.	
	iii)	**What do you mean by event based and cycle based simulator ?**	**4M**
	Ans:	**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. **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. • 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, multipliers etc. are modeled 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** **2M**

iv)	**List any four Features of Spartan 3 series FPGA.**	**4M**
Ans:	**<u>Features :</u>** • **Low-cost, high-performance logic solution for high-volume, consumer-oriented applications** • Densities up to 74,880 logic cells • **SelectIO interface signaling** • Up to 633 I/O pins • 622+ Mb/s data transfer rate per I/O • 18 single-ended signal standards • 8 differential I/O standards including LVDS, RSDS • Termination by Digitally Controlled Impedance • Signal swing ranging from 1.14V to 3.465V • Double Data Rate (DDR) support • DDR, DDR2 SDRAM support up to 333 Mb/s • **Logic resources** • Abundant logic cells with shift register capability • Wide, fast multiplexers • Fast look-ahead carry logic • Dedicated 18 x 18 multipliers • JTAG logic compatible with IEEE 1149.1/1532 • **Select RAM hierarchical memory** • Up to 1,872 Kbits of total block RAM • Up to 520 Kbits of total distributed RAM • **Digital Clock Manager (up to four DCMs)** • Clock skew elimination • Frequency synthesis • High resolution phase shifting • **Eight global clock lines and abundant routing**	**(1M- Any Four Features)**
B)	**Attempt any ONE :**	**6M**
i)	**Describe the twin tub process for CMOS fabrication.**	**6M**
Ans:	**<u>Twin Tub Process:</u>** **<u>Diagram :</u>**  N Substrate	**3M**

Explanation : **3M**

- 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.
- SiO2 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.
- A thin layer of SiO2 thinox is deposited all over the surface. Using masking and etching process unrequired thinox is removed. The thinox is required only on gate areas of both the transistors.
- The polysilicon is deposited all over the surface and using a mask it is removed from areas other the gate area.
- 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.
- 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.
- The thick layer of SiO2 is grown all over the surface for isolation. This SiO2 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.

ii) Write the VHDL program to implement 4 bit adder. **6M**

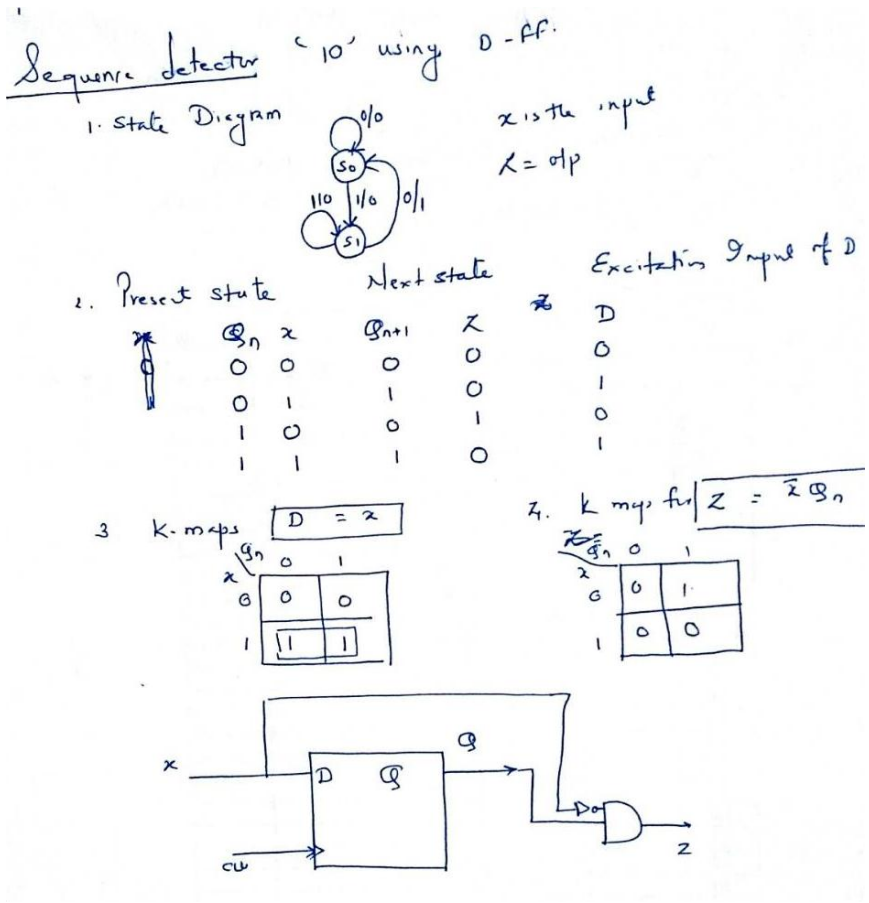
Ans:

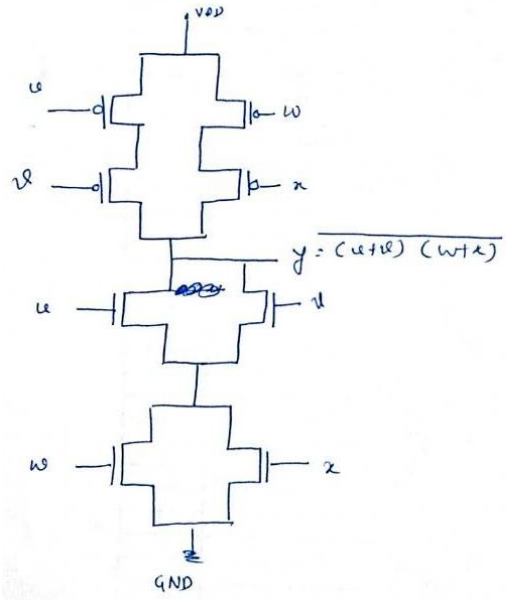
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity adder is
    Port ( a : in std_logic_vector(3 downto 0);
           b : in std_logic_vector(3 downto 0);
           s : out std_logic_vector(3 downto 0);
           c : out std_logic;
           cin : in std_logic);
end adder;

architecture behavior of adder is

begin
process(a,b,cin)
variable u:std_logic;
begin
u:=cin;
for i in 0 to 3 loop
s(i)<=a(i) xor b(i) xor u;
u:=(a(i) and b(i))or(b(i) and u) or(u and a(i));
end loop;
c<=u;
end process;

end behavior;
```

(1M - Package Declaration)

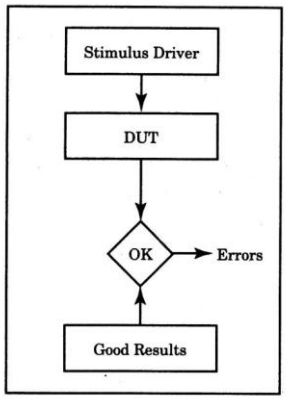
(1M – Entity)

(4M- Architecture)

Q 2		Attempt any FOUR :	16-Total Marks
	a)	**Design a sequence detector '10' using D-FF. If the sequence is valid, it gives the output z = '1' else 'z' =0.**	**4M**
	Ans:		**(1M-Excitation Table)** **(1M-k map)** **(1M-Implementation)** **(1M-Diagram)**
	b)	**Realize the equation y = $\overline{(u + v)(w + x)}$ using CMOS logic.**	**4M**
	Ans:		

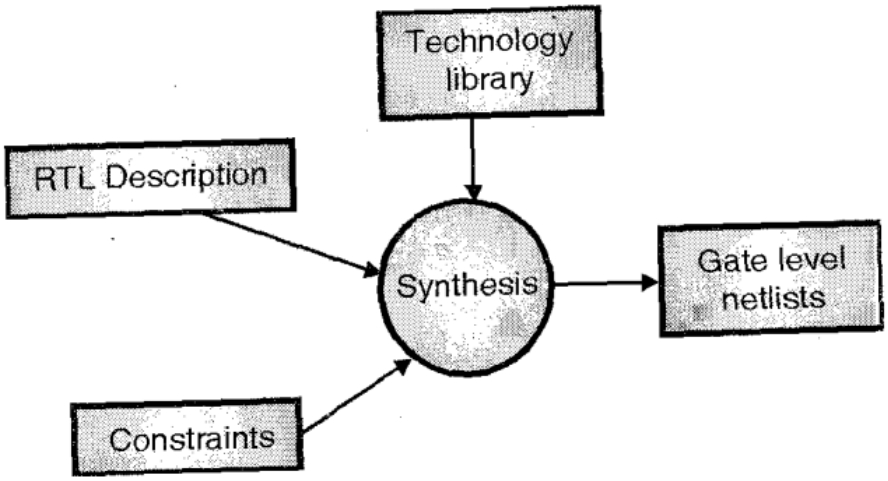
		$y = \overline{(u+v)(w+x)}$ using CMOS Logic $= \overline{(u+v)} + \overline{(w+x)}$ $= (\bar{u}.\bar{v}) + (\bar{w} + \bar{x})$  	**4M**
	c)	**What do you mean by enumerated data types ? Give the suitable example.**	**4M**
	Ans:	**<u>Enumerated:</u>** Defines the set of user defined values consisting of identifiers and character literals **<u>Example :</u>** type opcode is (load,store,add); opcode is enumerated type and supports load,store,add type mul is (‘U’, ‘0’, ‘1’, ‘Z’); set of ordered values U’, ‘0’, ‘1’, ‘Z’	**2M** **2M**
	d)	**What do you mean by test bench ? State its applications.**	**4M**
	Ans:	**<u>Definition :</u>** A test bench is HDL code that allows you to provide a documented, repeatable set of stimuli that is portable across different simulators. A test bench can be as simple as a file with clock and input data or a more complicated file that includes error checking, file input and output, and conditional testing. It 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>Applications :</u>** A test bench or testing workbench is an (often virtual) environment used to verify the correctness or soundness of a design or model, for example, that of a software product. A test bench refers to an environment in which the product under development is tested with the aid of software and hardware tools. The suite of testing tools is often designed specifically for the product under test. The software may need to be modified slightly in	**2M** **2M**

some cases to work with the test bench but careful coding can ensure that the changes can be undone easily and without introducing bugs.



e) Draw the HDL design flow for synthesis. Write the steps in the flow. **4M**

Ans: **Diagram :** **2M**



Explanation : **2M**

Synthesis is an automatic method of converting higher level of abstraction to lower level of abstraction.

- 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.
- 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 un optimized 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.

OR

Simple steps :

1. Describe your design with HDL
2. Perform RTL simulation
3. Synthesizing your design
4. Create Xilinx Netlist Files (XNF/EDIF etc)
5. Perform Functional Simulation
6. Floor planning of design (optional)
7. Placing and routing
8. Perform a timing simulation (post layout)

f) Compare FPGA and CPLD (any four). **4M**

Ans: **(1M- Any Four Points)**

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.

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

a) What is metastability ? Give the example. **4M**

Ans: **2M**

Metastability:

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.

Example:

A simple example of metastability can be found in an SR NOR latch, when both Set and Reset inputs are true (R=1 and S=1) and then both transition to false (R=0 and S=0) at about the same time. Both outputs Q and Q are initially held at 0 by the simultaneous Set and Reset inputs. After both Set and Reset inputs change to false, the flip-flop will (eventually) end up in one of two stable states, one of Q and Q true and the other false. The final state will depend on which of R or S returns to zero first, but if both transitions at about the same time, the resulting metastability, with intermediate or oscillatory output levels, can take arbitrarily long to resolve to a stable state. **2M**

b) Compare BJT and CMOS (any four). **4M**

Ans:

Sr.No.	CMOS Technology	Bipolar Technology
1.	Low static power dissipation	High power dissipation
2.	High input impedance	Low input impedance
3.	High packing density	Low packing density
4.	High delay sensitive to load	Low delay sensitive to load
5.	Low output drive current	High output drive current
6.	Bidirectional capability	Essentially unidirectional
7.	It is an ideal switching device.	It is not an ideal switching device.
8.	Voltage driven	Current driven
9.	High power application	Low power application
10.	Unipolar device	Bipolar Device
11.	High current gain	Low current gain
12.	It has less fan out	It has more fan out.

(1M- Any four point)

c) Define 1.Entity 2.Architecture in VHDL. **4M**

Ans: **2M**

Entity:-

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 describes the interface to the outside world. It specifies the number of ports, the direction of the ports, and the types of the ports.
One entity can be associated with much architecture.

Architecture:-

All entities that are declared have an architecture associated with it. Architecture describes the behavior, functionality, interconnections or relationship between inputs and outputs. Architecture contains only concurrent statement. **2M**
Architecture is always related to an entity and describes the behavior of that entity.
First part of the architecture may contain declaration of types, signals, constants, subprograms etc.

d) What the different measure should be taken to write the efficient code? **4M**

Ans: **4M**

Efficient Coding Styles:-

The style of writing VHDL code is very important. Effective VHDL coding techniques can make all the difference between designs that meet tough synthesis targets and verification schedules.

- Coding should be easy to read and maintain.

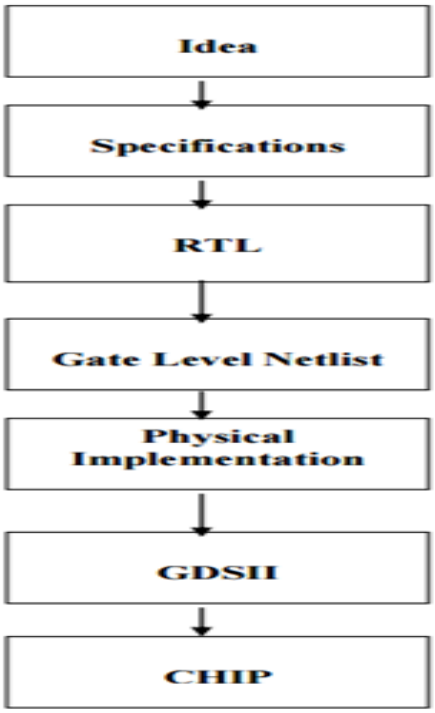
- It must yield expected results.
- It must follow VHDL language rules.
- It should have common look to enhance familiarity between different models.
- Out dated VHDL should be avoided.
- The common functions should be lumped in common packages, partitions or architectures.
- Behavioral and structural coding should be kept separate to reduce the debugging time
- Use one line for each signal in declarations instantiations and mappings which maintains clarity, more readable and understandable code.
- To avoid accidental mixing of signal use named Association
- Use active voice signals throughout the VHDL which makes the code easier to debug, test and reduce complication
- For combinatorial processes, never assign to a signal and read from the same signal in the same process. This will eliminate infinite loops when performing behavioral simulation.
- For sequential processes, never assign to a signal outside of control of the rising edge clock statement.
- Within a combinational process, all signals that are read must be in the sensitivity list.
- Within a sequential process only asynchronous set/ reset and clock should be in the sensitivity list.
- Always make an assignment to a variable before it is read. Otherwise, variables will infer either latches are registers to maintain their previous value.

Use subprograms wherever possible.

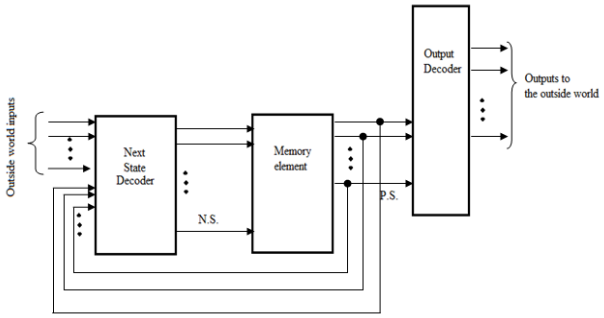
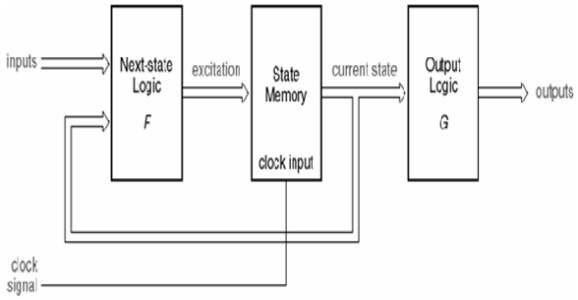
e) Draw the ASIC design flow and explain it. **4M**

Ans: ASIC DESIGN FLOW: **2M**

Diagram :



		Explaination : **Specifications:** In this step all the functionality and features are defined, such as power consumption, voltage reference, timing restrictions and performing criterion. Chip planning is also performed in this step. The next step is to decide the architecture for the design from the specification. **RTL Coding:** This is beginning of the ASIC design flow. The micro architecture is transformed into RTL code, RTL is expressed usually in Verilog or VHDL, by using a HDL one can describe any hardware (digital) at any level. **Simulation:** Functional/Logical Verification is performed at this stage to ensure the RTL designed matches the idea. **Synthesis:** Once Functional Verification is completed, the RTL is converted into an optimized Gate Level Netlist. This step is called Logic/RTL synthesis. This is done by Synthesis Tools such as Design Compiler (Synopsys), Blast Create (Magma), RTL Compiler (Cadence) etc... A synthesis tool takes an RTL hardware description and a standard cell library as input and produces a gate-level netlist as output. The resulting gate-level netlist is a completely structural description with only standard cells at the leaves of the design. At this stage, it is also verified whether the Gate Level Conversion has been correctly performed by doing simulation. **Physical Implementation:** The next step in the ASIC flow is the Physical Implementation of the Gate Level Netlist. The Gate level Netlist is converted into geometric representation. The geometric representation is nothing but the layout of the design. The layout is designed according to guidelines based on the limitations of the fabrication process. The Physical Implementation step consists of three sub steps; Floor planning, Placement, Routing The file produced at the output of the Physical Implementation is the **GDSII** file. It is the file used by the foundry to fabricate the ASIC. Physical Verification is performed to verify whether the layout is designed according the rules. For any design to work at a specific speed, timing analysis has to be performed. We need to check whether the design is meeting the speed requirement mentioned in the specification. This is done by Static Timing Analysis Tool; it validates the timing performance of a design by checking the design for all possible timing violations for example; set up, hold timing. After Layout, Verification, Timing Analysis, the layout is ready for Fabrication. The layout data is converted into photo lithographic masks. After fabrication, the wafer is diced into individual chips. Each Chip is packaged and tested.	**2M**
Q. 4	**A)**	**Attempt any THREE :**	**12M**
	i)	**Draw the Moore machine and write its o/p equation.**	**4M**
	Ans:	**Diagram :**	**2M**

		 **OR**  **Moore Machine:** Moore machine is the sequential system where output depends only on present state. f (o/p) = f (Present State) **OR** Next state = F (Current state, input) Output = G (Current state)	**2M**
ii)		**List the any four logical operators in VHDL.**	**4M**
Ans:		**Logical Operators:**These are defined for type bit and Boolean, one dimensional array of bit and Boolean type. The logical operators are: • AND • OR • NAND • NOR • XOR • XNOR • NOT	**(1M- Any Four Types)**
iii)		**What do you mean by delta delay ? Give the example.**	**4M**
Ans:		**Definition :** The real time that the simulator takes to execute one simulation cycle is known as delta delay for simulation delta with zero simulation time.A delta delay is very small and does not correspond to any real delay and actual simulation time does not advance. Delta delay is introduced to achieve concurrency and order independency .The simulator freezes simulation time until all scheduled assignments in current simulation time is finished and there are no more events in the sensitivity list.	**2M**

Example : **2M**

B<= NOT A; C<= B NAND CLK; D<= C AND B;	B<= NOT A; D<= C AND B; C<= B NAND CLK;
If A changes '1' to '0'	
B <=1	B <=1
Evaluate AND (C=1)	Evaluate NAND
D<=1	C<=0
Evaluate NAND	Evaluate AND
C<=0	D<=0
Evaluate AND	
D<=0	

Both the syntax gives the same values and achieved concurrency with independent order of syntax.

iv) **Describe the following statement with syntax :** **4M**

i) wait

ii) assert.

Ans: **1. <u>wait:</u>**statement suspends the execution of a process or procedure until some conditions are met. **1M**

<u>Syntax :</u> **1M**

wait on [sensitivity list]; eg. wait on clk;

wait until [condition]; wait until clk='1''
wait for [time out expression]; wait for 20 ns;

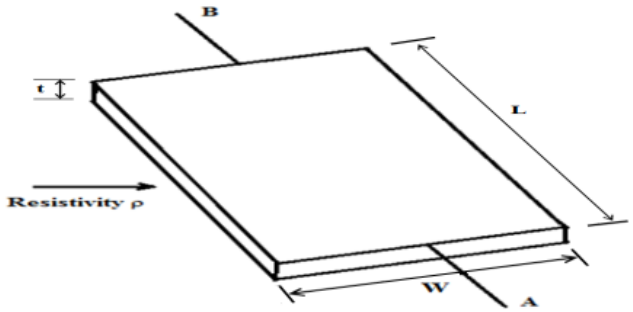
2. <u>assert</u> : Assert is a **non-synthesizable** statement whose purpose is to write out messages on the screen when problems are found during simulation **1M**

<u>Syntax :</u>
ASSERT condition **1M**
[REPORT "message"]
[SEVERITY severity_level];
The message is written when the condition is FALSE.
Severity_level can be: **Note, Warning, Error (default), or Failure.**

B) **Attempt any ONE ;** **6M**

i) **Describe the resistance estimation of conducting material of uniform sheet with 'L' length. 'ρ' resistivity ,'w' width and 't' thickness.** **6M**

Ans: **<u>Diagram :</u>** **2M**



Formulation :

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. **2M**

Hence the resistance between A and B terminal.

RAB = ohms.

Where A = cross-sectional area.

Thus RAB = ohms.

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

RAB = = Rs

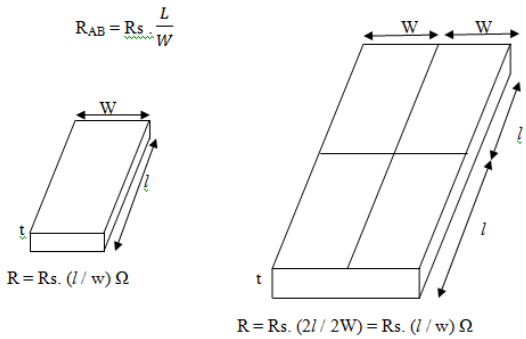
Where

Rs = ohm per square or sheet resistance

Therefore, Rs = ohm per square

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

Thus,

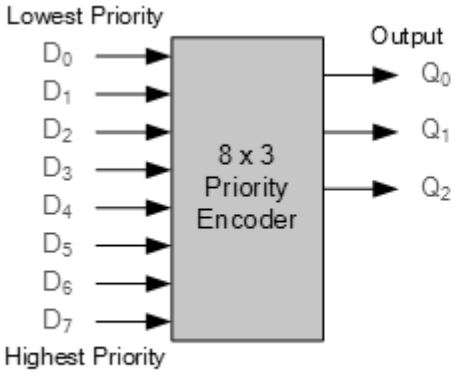


Explanation :

Thus to obtain the resistance of a conductor on a layer multiply the sheet resistance Rs, by the ratio of length to width of the conductor as shown in the equation. For examples, resistances of the two shapes shown in the above figure are same because the length to width ratio of both the slabs is same, even though the sizes are different. Although the voltage – current characteristics of a MOS transistor are generally non-linear, it is used to approximate its behaviour in terms of a 'change resistance' to estimate the performance. **2M**

ii) Write the VHDL program to implement 8 : 3 encoder. **6M**

Ans: Diagram and truth table : **1M each**



Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

X = dont care

Program :

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity encoder8_3 is
    port(
        din : in STD_LOGIC_VECTOR(7 downto 0);
        dout : out STD_LOGIC_VECTOR(2 downto 0)
        );
end encoder8_3;
architecture encoder8_3_arc of encoder8_3 is
begin
   dout <= "000" when (din="10000000") else
          "001" when (din="01000000") else
          "010" when (din="00100000") else
          "011" when (din="00010000") else
          "100" when (din="00001000") else
          "101" when (din="00000100") else
          "110" when (din="00000010") else
          "111";
end encoder8_3_arc;
```

4M

Q.5		**Attempt any FOUR :**	**16M**
	a)	**Draw the state diagram and state table for 3 bit binary counter.**	**4M**
	Ans:	**Diagram :**	**2M**

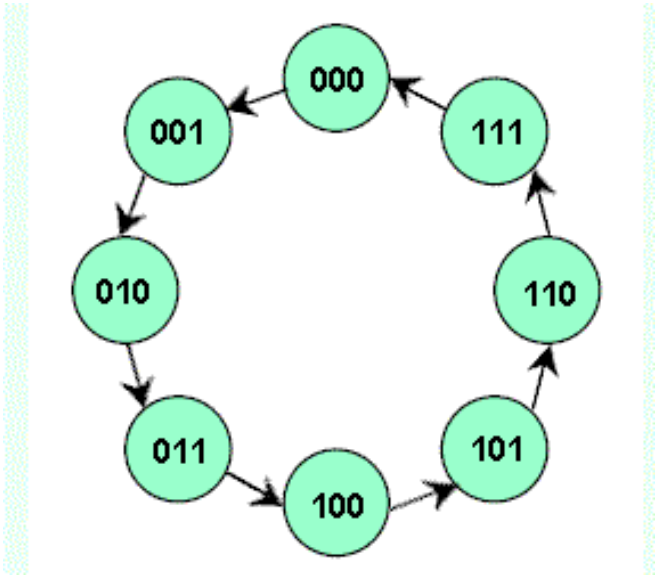


Table: **2M**

Present State Q2 Q1 Q0	Next State Q2 Q1 Q0
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	0 1 1
0 1 1	1 0 0
1 0 0	1 0 1
1 0 1	1 1 0
1 1 0	1 1 1
1 1 1	0 0 0

b) Explain with the syntax : 1.Signal 2.Variable **4M**

Ans: **1. Signal:** **1M**

• Signal objects are used to connect entities together to form models.
• A signal declaration looks like;

Syntax : **1M**

SIGNAL signal_name : signal_type [:= initial value];

2. Variables: **1M**

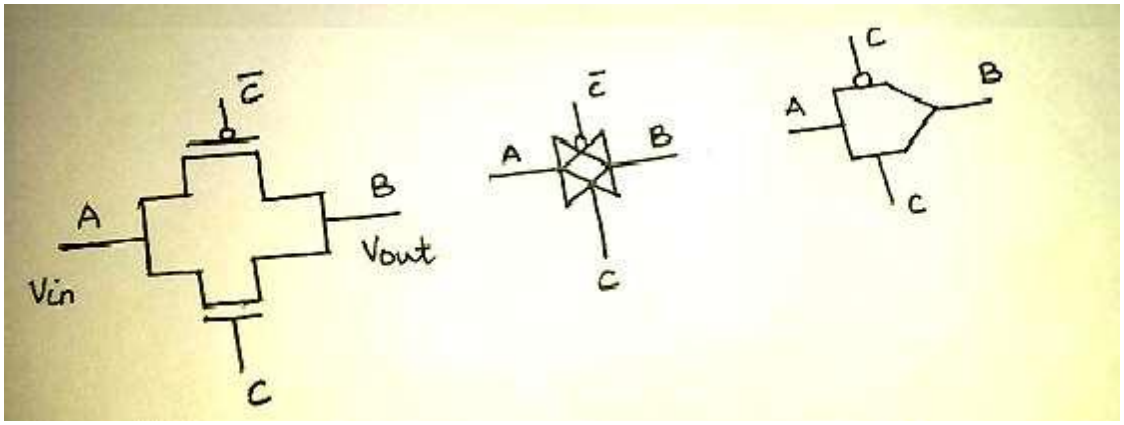
• Variables are used for local storage in process statements and subprograms.
• A variable declaration looks like this

Syntax :

VARIABLE variable_name {,variable_name} : variable_type [:=value]; **1M**

c) Describe the transmission gate with neat sketch. **4M**

Ans: **Diagram :** **2M**



Explanation : **2M**

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.

d) Write the VHDL program for 4 : 1 Mux. **4M**

Ans:

```
Library IEEE;
Use IEEE. Std_logic_1164.all;
Entity MUX4_1 is
                    Port(I : in std_logic_vector (3 downto 0);
                         S: in std_logic _vector (1 downto 0);
                         Y: out std_logic);
                    end MUX4_1;
architecture MUX of MUX4_1 is
begin
                    with S Select
                    Y <=  I(0) when "00"
                          I(1) when "01"
                          I(2) when "10"
                          I(3) when "11";
                    '0' when others; -- optional
end MUX;
```

(Any other 4:1 program with different cases marks to be given)

e) Differentiate software programming language and hardware descriptive language. **4M**

Ans:

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.

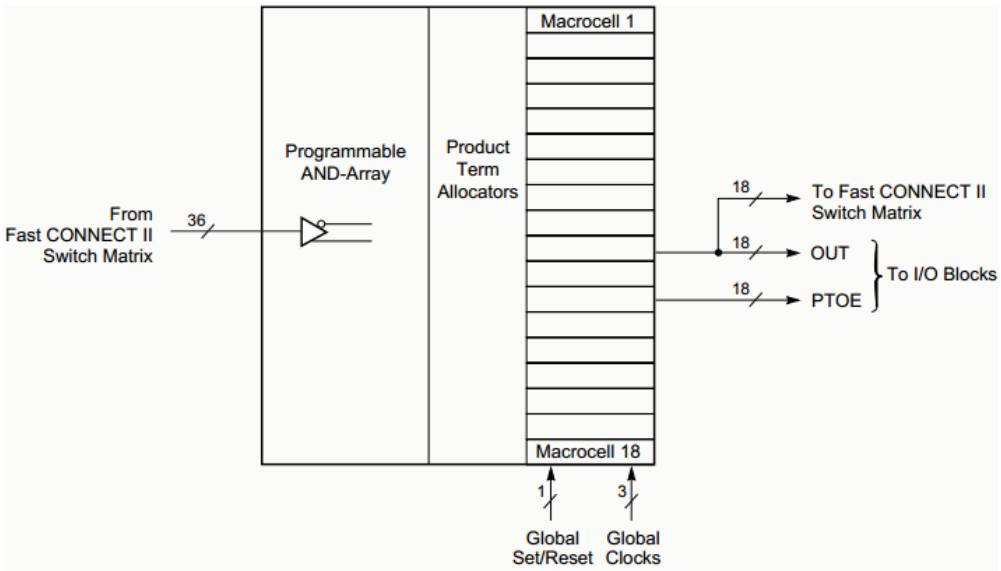
Note:Any other relevant points marks to be given.

(1M Each Points)

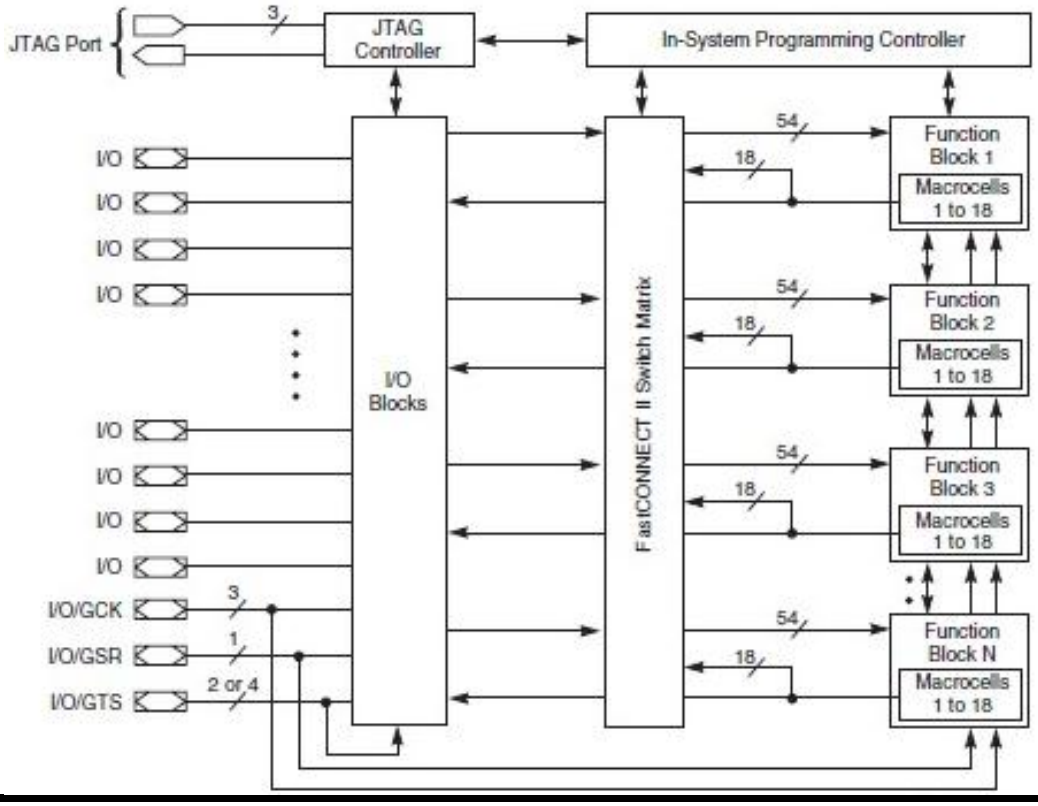
f) Draw the functional block architecture of Xilinx CPLD. **4M**

Ans: **Diagram : .** **4M**

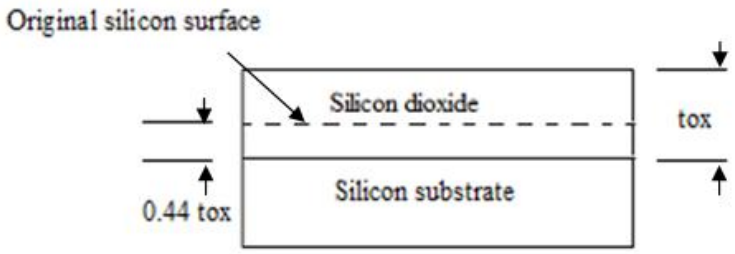
Note:Any One relevant diagram



OR

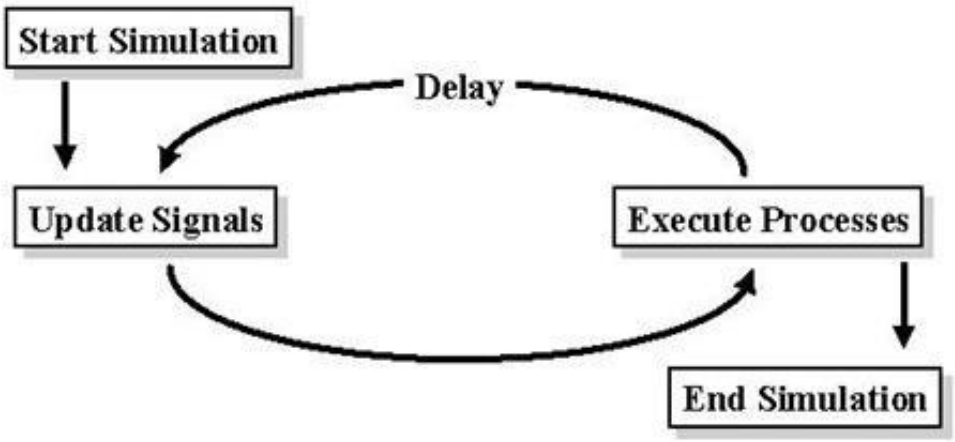


Q.6		**Attempt any FOUR:**	**16M**
	a)	**Describe the process of oxidation.**	**4M**
	Ans:	**<u>Oxidation :</u>** Oxidation is a process by which a layer of silicon dioxide is grown on the surface of a silicon wafer. The oxidation of silicon is necessary throughout the modern integrated circuit fabrication process. Therefore the reliable manufacture of SiO2 is extremely important.	**4M**

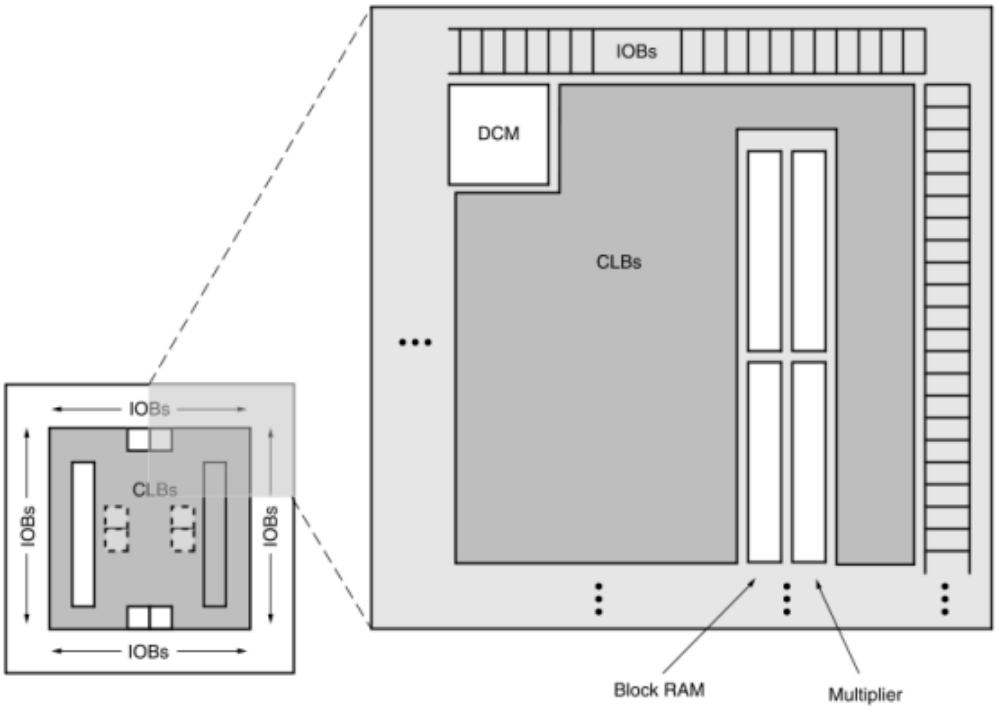


Oxidation of silicon is achieved by heating silicon wafers in an oxidizing atmosphere such as oxygen or water vapour.
Two common methods of oxidation are:
Wet Oxidation: When oxidizing atmosphere contains water vapour the temperature is usually between 9000C and 10000C.
This is rapid process.
Dry Oxidation: When the oxidizing atmosphere is pure oxygen temperature are in the region of 12000C, to achieve an acceptable growth rate.

b)	**Define sensitivity list. State any two VHDL syntax in which sensitivity list is defined.**	**4M**
Ans:	**Definition :** 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. **Syntax :** **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. 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.	**2M** **1M** **1M**
c)	**Draw the simulation cycle and label it.**	**4M**
Ans:	**Diagram :**	**4M**



d)	**Draw the FPGA configurable logic block with neat label.**	**4M**
Ans:	**Diagram :**	**4M**

			
	e)	**Write two examples of CPLD and FPGA. Write one application of CPLD and FPGA.**	**4M**
	Ans:	**<u>Example of CPLD:</u>** 1. Xilinx XC9500 CPLD 2. CPLD ATF1502ASV **<u>Applications of CPLD:</u>** 1. Complex programmable logic devices are ideal for high performance, critical control applications. CPLD can be used in digital designs to perform the functions of boot loader CPLD is used for loading the configuration data of a field programmable gate array from non-volatile memory. Generally, these are used in small design applications like address decoding CPLDs are frequently used many applications like in cost sensitive, battery operated portable devices due to its low size and usage of low power. **<u>Examples of FPGA:</u>** 1. 5K – 50K Gates Coprocessor FPGA with Free RAM 2. XILINX XC 4000 **<u>Applications of FPGA:</u>** 1. Aerospace and Defense 2. Consumer Electronics 3. Medical 4. ASIC Prototyping	**2M** **(2M Any other relevant application and examples marks to be given)**