USN | | | | | | | | | |

Internal Assessment Test 1 – Sept. 2019

Scheme of Evaluation

CMR Institute of Technology, Bangalore		
Department(s): Telecommunication Engineering		
Semester: 05	**Section(s): TCE & ECE B**	Lectures/week: 04
Subject: Verilog HDL		Code: 17EC53
Course Instructor(s): prof.Sophiya Susan S/prof.Sunil Kuamr H/Prof.Monika Singh.Prof.Jyoti M R		
Course duration: 01 Aug. 2019– 25Nov. 2019		
Course Site: https://sites.google.com/a/cmrit.ac.in/sophiya-susan/home/courses		

Sub:	Verilog HDL					Sub Code:	17EC53	Branch:	ECE/TCE
Date:	7/9/2019	Duration:	90 min's	Max Marks:	50	Sem / Sec:	V		OBE

Answer any FIVE FULL Questions — MARKS — CO — RBT

1 Design and write a 4-bit Ripple Carry Counter(RCC) HDL top level module and test bench/stimulus block for the same top level module [10]

- **Program For 4-bit ripple counter top level with necessary comments -5M**
- **test bench/stimulus block for the top level module with necessary comments -5M**

2 Write a 4-bit ripple carry adder top level module and test bench /stimulus block for the top level module. [10]

- **Program For 4-bit ripple carry adder top level with necessary comments -5M**
- **test bench/stimulus block for the top level module with necessary comments -5M**

3 Explain the typical VLSI IC design flow with the help of flow chart. [10]

- VLSI IC design flow diagram **6M**
- Explanation **4M**

4 What are the components of SR latch? Write Verilog HDL module of SR latch along with the test bench/stimulus block? [10]

- components of SR latch—**2M**
- Verilog HDL module of SR latch **with necessary comments** —3M
- test bench/stimulus block **with necessary comments** ---5M

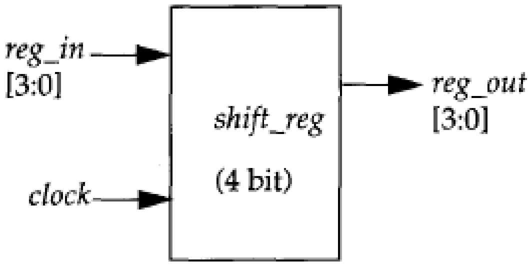
5 a. Describe why Verilog HDL has evolved as popular HDL in digital circuit design? [4]

- **Any 4 Popularity of Verilog HDL 1 mark each- Total 4M**

b. With a neat diagram discuss the components of Verilog HDL module. [6]

- Components of Verilog HDL Module – **3M**

- Expalantion **3M**

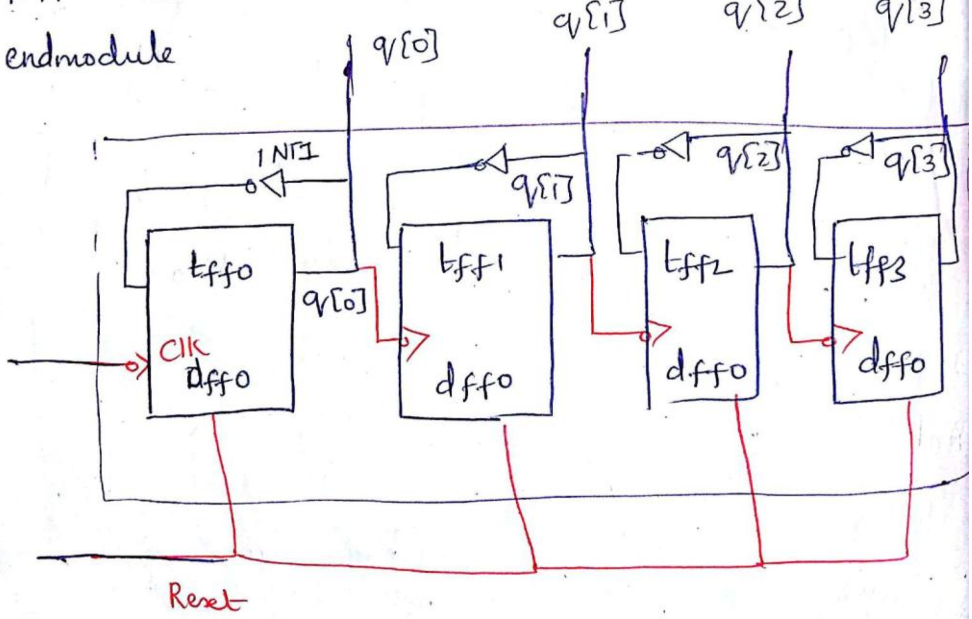
6	a. A 4-bit parallel shift register has I/O pins as shown in the figure below. Write the module definition for this module *shift_reg.* Include the list of ports and port declarations. You do not need to show the internals. **2M**  1. Declare a top-level module *stimulus.* Define *REG_IN* (4 bit) and CLK (1 bit) as reg register variables and *REG_OUT* (4 bit) as wire. Instantiate the module *shift_reg* and call it *srl.* Connect the ports by ordered list. **2M** 2. Connect the ports in Step 4 by name. **2M**	[6]	
	b. Briefly Explain the trends in HDLs? • 4 Points on Trends and Explain **4M**	[4]	

1. Design and write a 4-bit Ripple Carry Counter(RCC) HDL top level module and test bench/stimulus block for the same top level module

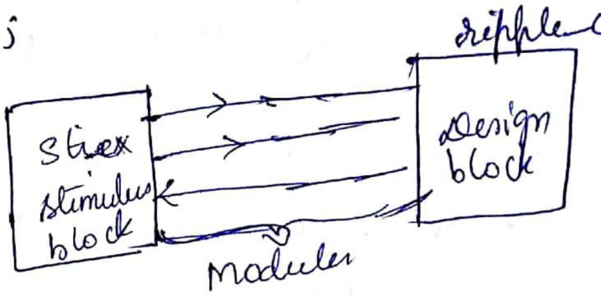
- **Program For 4-bit ripple counter top level with necessary comments -5M**

Code for 4-bit Ripple Carry Using T-ff designed intuen Using d-ff & NOT.

```
Module Ripple-carry ( q, Clk, reset);
output [3:0]   q;
 input  Clk, reset;

T-ff   dff0( q[0], clk, reset);
T-ff   dff1( q[1], q[0], reset);
T-ff   tff2 ( q[2], q[1], reset);
T-ff   tff3 (q[3], q[2], reset);
endmodule
```



```
T_ff tff2 (q[2], Q[1], reset);
T_ff tff3 (Q[3], Q[2], reset);
```



stimulus block

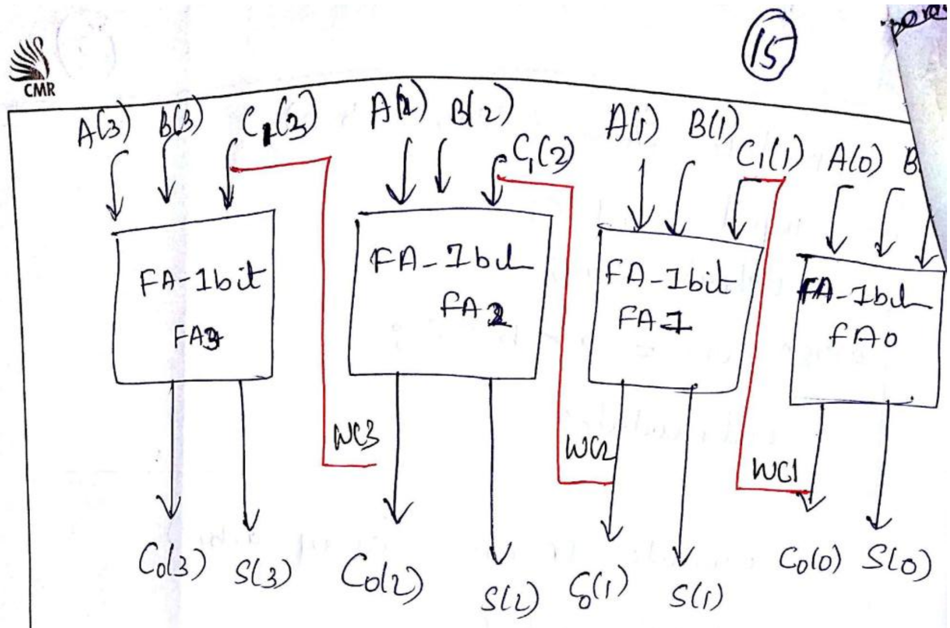
```
module sti_ex;
reg clk, reset;
wire [3:0]Q;
ripple_c c1(q, clk, reset);
initial
    clk = 1'b0;

always
#5 clk = ~clk;
initial
    begin
    reset = 1'b1;
#10 reset = 1'b0;
#180 reset = 1'b1;
    $ finish
  end
initial
 $monitor ($ time, " o/p q = %d ", q);
 endmodule
```

- **test bench/stimulus block for the top level module with necessary comments -5**

•

2. Design and write a 4-bit ripple carry adder HDL top level module and test bench /stimulus block for the same top level module.

- **Program For 4-bit ripple carry adder top level with necessary comments -5M**




```
module  4-bit FA ( S, Co, a, b, Ci)
Input [3:0] a, b ;
Output [3:0] S ;
wire  output  Co3;
      WC1 , WC2 ; WC3;

FA-1   FA0 (S(0), WC1, a(0), b(0), 0);
FA-2   FA1(S(1), WC2  a(1), b(1), WC1);
FA-3   FA2 ( S(2), WC3, a(2), b(2), W(2));
FA-4   FA3 ( S(3), Co3, a(3), b(3), W(3));
```

NOTE:
Cin of
LSB = 0
∴ Add

- **Test bench/stimulus block for the top level module with necessary comments -5M**

Example 5-9 Stimulus for 4-bit Ripple Carry Full Adder

```
// Define the stimulus (top level module)
module stimulus;

// Set up variables
reg [3:0] A, B;
reg C_IN;
wire [3:0] SUM;
wire C_OUT;

// Instantiate the 4-bit full adder. call it FA1_4
fulladd4 FA1_4(SUM, C_OUT, A, B, C_IN);


// Set up the monitoring for the signal values
initial
begin
  $monitor($time," A= %b, B=%b, C_IN= %b, --- C_OUT= %b, SUM= %b\n",
                          A, B, C_IN, C_OUT, SUM);
end

// Stimulate inputs
initial
begin
  A = 4'd0; B = 4'd0; C_IN = 1'b0;

  #5 A = 4'd3; B = 4'd4;

  #5 A = 4'd2; B = 4'd5;

  #5 A = 4'd9; B = 4'd9;

  #5 A = 4'd10; B = 4'd15;

  #5 A = 4'd10; B = 4'd5; C_IN = 1'b1;
end

endmodule
```

The output of the simulation is shown below.

```
0 A= 0000, B=0000, C_IN= 0, --- C_OUT= 0, SUM= 0000
5 A= 0011, B=0100, C_IN= 0, --- C_OUT= 0, SUM= 0111
10 A= 0010, B=0101, C_IN= 0, --- C_OUT= 0, SUM= 0111
15 A= 1001, B=1001, C_IN= 0, --- C_OUT= 1, SUM= 0010
```

3.Explain the typical VLSI IC design flow with the help of flow chart.

- VLSI IC design flow diagram **6M**
- Explanation **4M**

VLSI DESIGN FLOW.

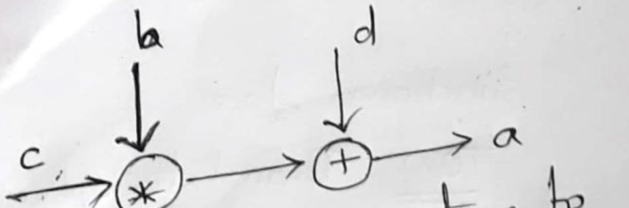
1) **Design Specification**: Requirements / demands of Customers

2) **Behavioural description**: Architects need not think about how they are implemented in the Ckts. Its either an algorithm or the plan of the requirements.

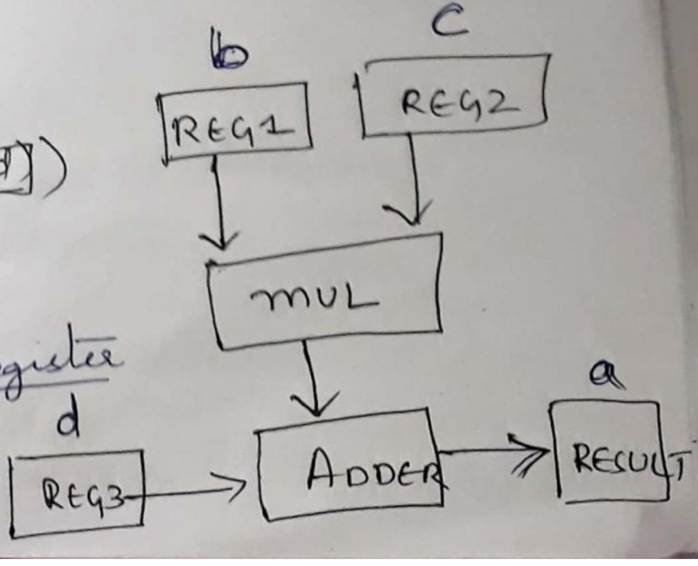
$a = (b * c) + d$

3) **RTL Description**
Register Transfer Logic

- **HDL description**
  Write the HDL description Using any description model (Behavioural level / Gate level / data flow).



(Using description to obtain $a = b*c+d$)



In terms of Register

The HDL desceiption is functionally
4> Veerified & Tested Using Tool (simulation).

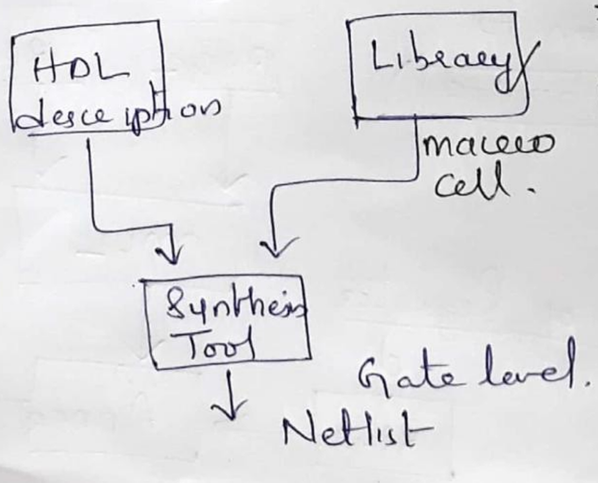
Until this step the design flow is said to be design Independent of Technology/device

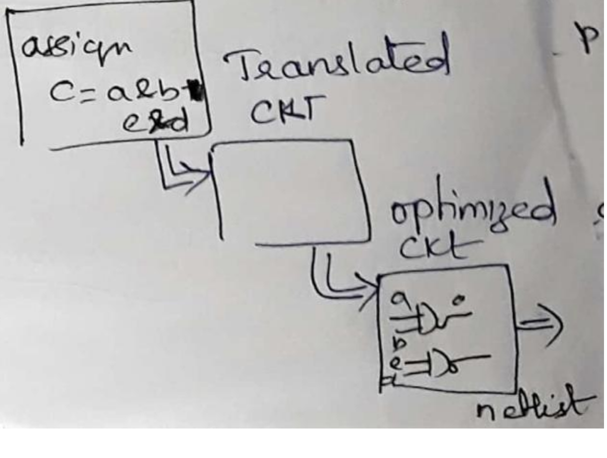
Technology - 45nm/90nm/180nm...
device - FPGA/Asic

5> Synthesis
The further on process/design flow is process dependent on Technology.

Synthesis: is a Process of Converting The HDL desceiption along with required Library/built in modules to Gate level netlist

Synthesis does:
<u>TOM</u>: Translate, optimize & map.





6) Gate level netlist (.bit file)
- is the file that is obtained from Synthesis.
- Its an Encrypted file of the design that will be Given to the fabricating Industry

7) Verification & Testing can be carried out based on The Technology/device

6

. Until this we can consider the design as front end design flow

8) The next back end design is
Planning
Placing
Routing
&
} The Gate level netlist is the input to the Automatic place, Route and Create layout

9> Layout

10> Layout verifications
- DRC (Design Rule check)
- LVS (Layout v/s Schematic check
- ERC

ERC: Electrical Rule check (power & Gnd Connection fanouts, slew Capacitive loads)

(Then the layout is Verified and fabricated on a chip)

~~Typical data~~

→ TYPICAL DESIGN Flow

1) Specifications

2) Behavioural description — front end Design Engineer

3) RTL description (Register Transfer Logic) HDL → is converted to Logic Gates / ff / Registers / mux / . . .

HDL → functional Verification & Testing — EDA Tool

→ Logical Synthesis / Timing Constraints → Translate → Mapping → Routing

Test Engineer

4) Gate level Netlist .bit file (Encrypted file of design that will be Given to the Industry)

→ Logical Verification & Testing

→ Automatic Place & Route (Floor plan)

5) Physical layout

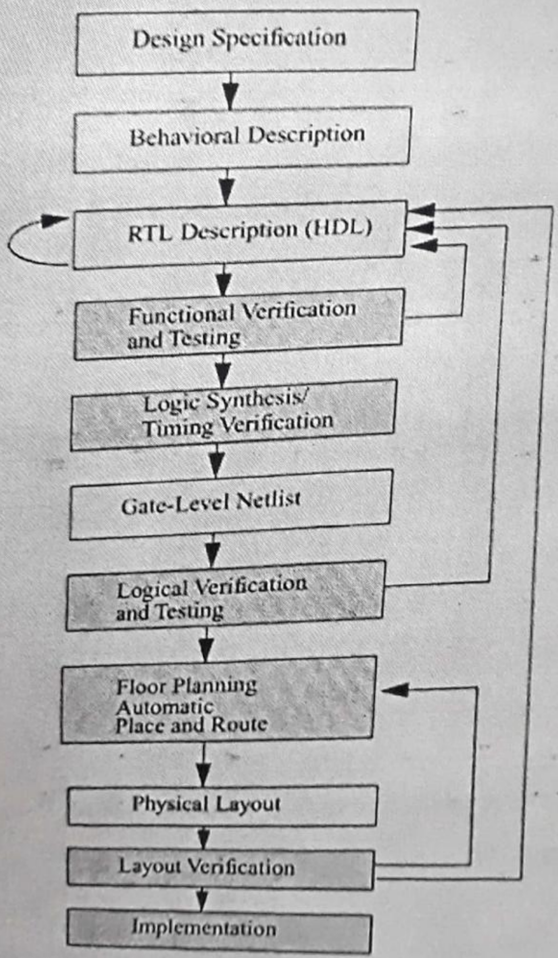
→ Layout Verification

→ Implementation

Levels of design — processes in design Flow

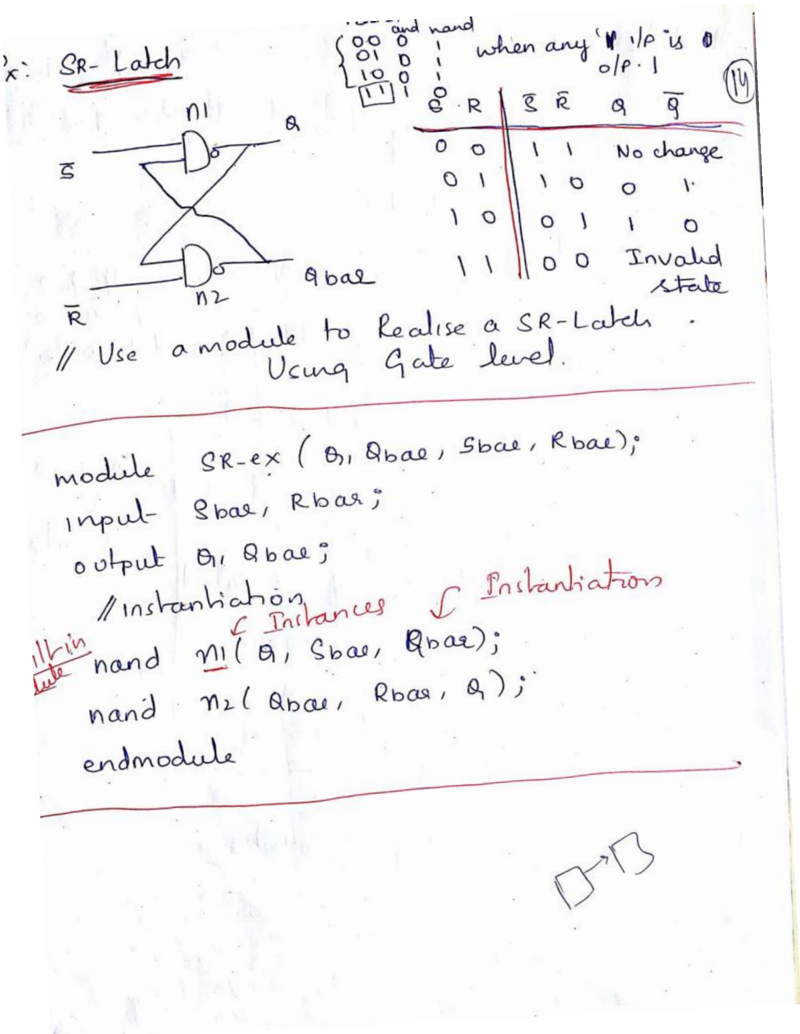
Scanned by CamScanner

A typical design flow for designing VLSI IC circuits is shown in *Figure 1-1*. *Unshaded blocks show the level of design representation; shaded blocks show processes in the design flow.*



4. What are the components of SR latch? Write Verilog HDL module of SR latch along with the test bench/stimulus block?

- components of SR latch—**2M**
- Verilog HDL module of SR latch **with necessary comments** —3M
- test bench/stimulus block **with necessary comments** ---5M



SR- Latch

	and	nand
00	0	1
01	0	1
10	0	1
11	1	0

when any i/p is 0 o/p. 1

S	R	$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$
0	0	1	1	No change	
0	1	1	0	0	1
1	0	0	1	1	0
1	1	0	0	Invalid state	

// Use a module to Realise a SR-Latch Using Gate level.

```
module SR-ex (Q, Qbar, Sbar, Rbar);
input Sbar, Rbar;
output Q, Qbar;
//instantiation                 Instances    Instantiation
nand n1(Q, Sbar, Qbar);
nand n2(Qbar, Rbar, Q);
endmodule
```

15

Module stimulus-ex // Name of stimulus module

// declare wires or reg

⇓ o/p of design

⇓ o/p of stimulus i/p to design module

```
wire q, qbar;
reg n-set, nreset;
    // Instantion of the design module

SR-ex   SR1 (q, qbar, n-set, n-reset);

    // Behavioural Block.

Initial
  begin
  $monitor( $time , "n-set =%b n-reset = %b q=%b
                    n-Set, nreset, q);
        Set =0 ; reset =0;
   #10  reset =1;
   #10  reset =0
   #10  set =1
   end
 endmodule
```

5 a. Describe why Verilog HDL has evolved as popular HDL in digital circuit design? [4]

Any 4 Popularity of Verilog HDL 1 mark each- Total 4M

* Verilog HDL is a general-purpose hardware descripl language. i.e Easy to learn and Easy to use.

* Syntax is similar to the 'C' programming, where designer's with 'C' programming Experience will find it Easy to learn Verilog HDL.

* Verilog HDL allows different levels of abstraction to be mixed in the same model. such as switch level, gate level, RTL or Behavioral code.

* Designer need to learn only one language for stimulus and hierarchial design

* Most of the logic synthesis tools support Veril HDL. This make it the language of choice for designe

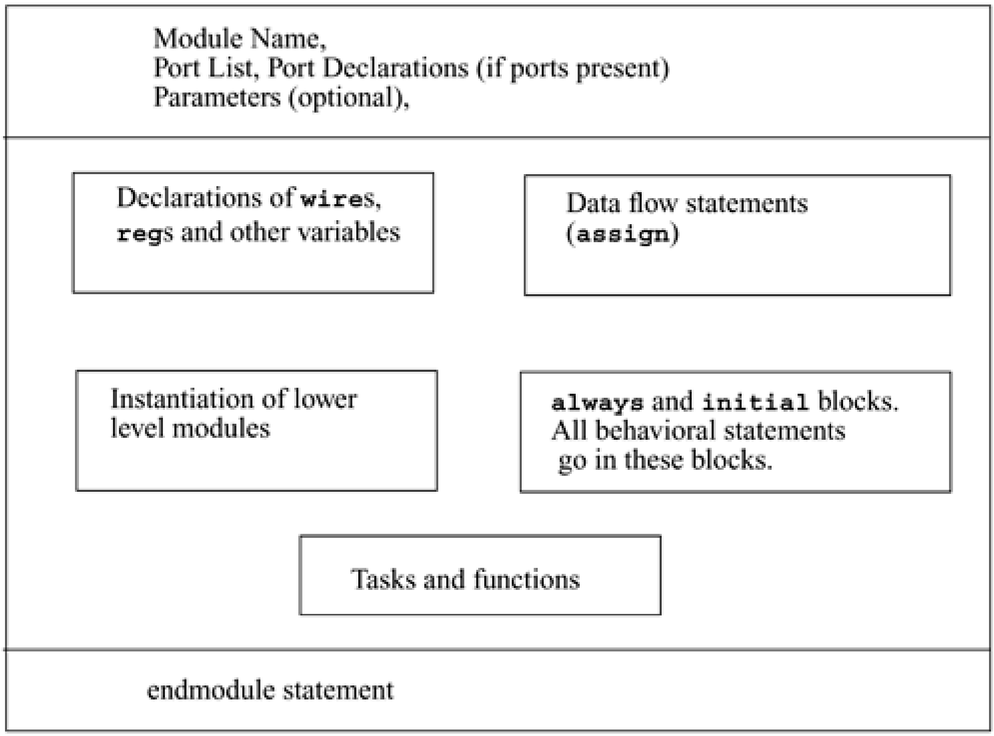
* The programming Language Interface (PLI) is a powerf feature that allows the user to write custom 'c' code to interact with the internal data structure of Verilog.

Page 9 / 29

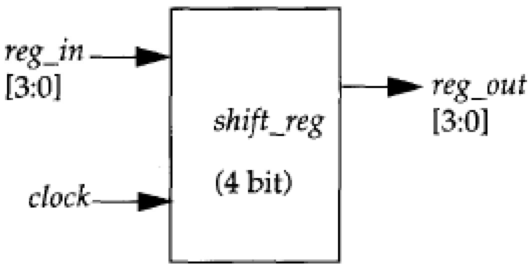
b. With a neat diagram discuss the components of Verilog HDL module. [6]

- Components of Verilog HDL Module – **3M**
- Expalantion **3M**

A module in Verilog consists of distinct parts, as shown in Figure A module definition always begins with the keyword module. The module name, port list, port declarations, and optional parameters must come first in a module definition.Port list and port declarations are present only if the module has any ports to interact with the external environment.The five components within a module are: variable declarations,dataflow statements, instantiation of lower modules, behavioral blocks, and tasks orfunctions. These components can be in any order and at any place in the module definition. The endmodule statement must always come last in a module definition. All components except module, module name, and endmodule are optional and can be mixed and matched as per design needs. Verilog allows multiple modules to be defined in a single file. The modules can be defined in any order in the file.



6 a. A 4-bit parallel shift register has I/O pins as shown in the figure below. Write the module definition for this module *shift_reg.* Include the list of ports and port declarations. You do not need to show the internals.2m



1. Declare a top-level module *stimulus.* Define *REG_IN* (4 bit) and CLK (1 bit) as reg register variables and *REG_OUT* (4 bit) as wire. Instantiate the module *shift_reg* and call it *srl. 2m*
2. Connect the ports in Step 4 by name.2m

a. module *shift_reg(reg_out, reg_in,clk);*
input [3:0] *reg_in;*

```
input clk;
output reg_out;
…………………
……………….
endmodulec
```

1 Declare a top-level module *stimulus*. Define *REG_IN* (4 bit) and CLK (1 bit) as reg register variables and *REG_OUT* (4 bit) as wire. Instantiate the module *shift_reg* and call it *srl*. ***2m***

```
module stimulus;
reg [3:0] REG_IN;
reg CLK;
wire REG_OUT;
shift_reg srl (REG_OUT, REG_IN,CLK);

……………….
endmodule
```

2.Connect the ports in Step 4 by name.**2m**

```
module stimulus;
reg [3:0] REG_IN;
reg CLK;
wire REG_OUT;
shift_reg srl (.reg_out (REG_OUT), .reg_in (REG_IN),.clk(CLK));

……………….
endmodule
```

b. Briefly Explain the trends in HDLs?

- Points on Trends and Explain **4M**

Trends in HDL's

* The most popular trend Currently is to design in HDL at an RTL level, because logic Synthesis tools Can Create gate-level netlists from RTL level design. Today, RTL design Continues to be Very popular.
* Verilog HDL is also being Constantly Enhanced to meet the needs of new Verification methodologies.
* Then formal Verification and assertion checking techniques have Emerged.
* formal Verification Applies formal mathematical techniques to Verify the Correctness of Verilog HDL descriptions and to Establish Equivalency between RTL and gate-level netlists.

* New Verification languages have also gained rapid acceptance. These languages combine the parallelism and hardware constructs from HDL's with the Object Oriented nature of C++.

* For very high-speed and timing-critical circuits like microprocessors, the gate level netlist provided by logic synthesis tools is not optimal. In such cases, designers often mix gate-level description directly into the RTL description to achieve optimum results.

* Another technique that is used for system-level design is a mixed bottom-up methodology where the designers use either existing Verilog HDL modules, basic building blocks, or vendor-supplied core blocks to quickly bring up their system simulation. This is done to reduce development costs and compress design schedules.