CMR

INSTITUTE OF

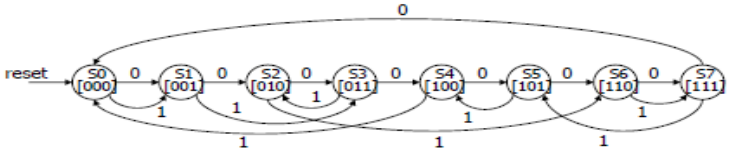
TECHNOLOGY

USN | | | | | | | | | | |

Internal Assesment Test - I

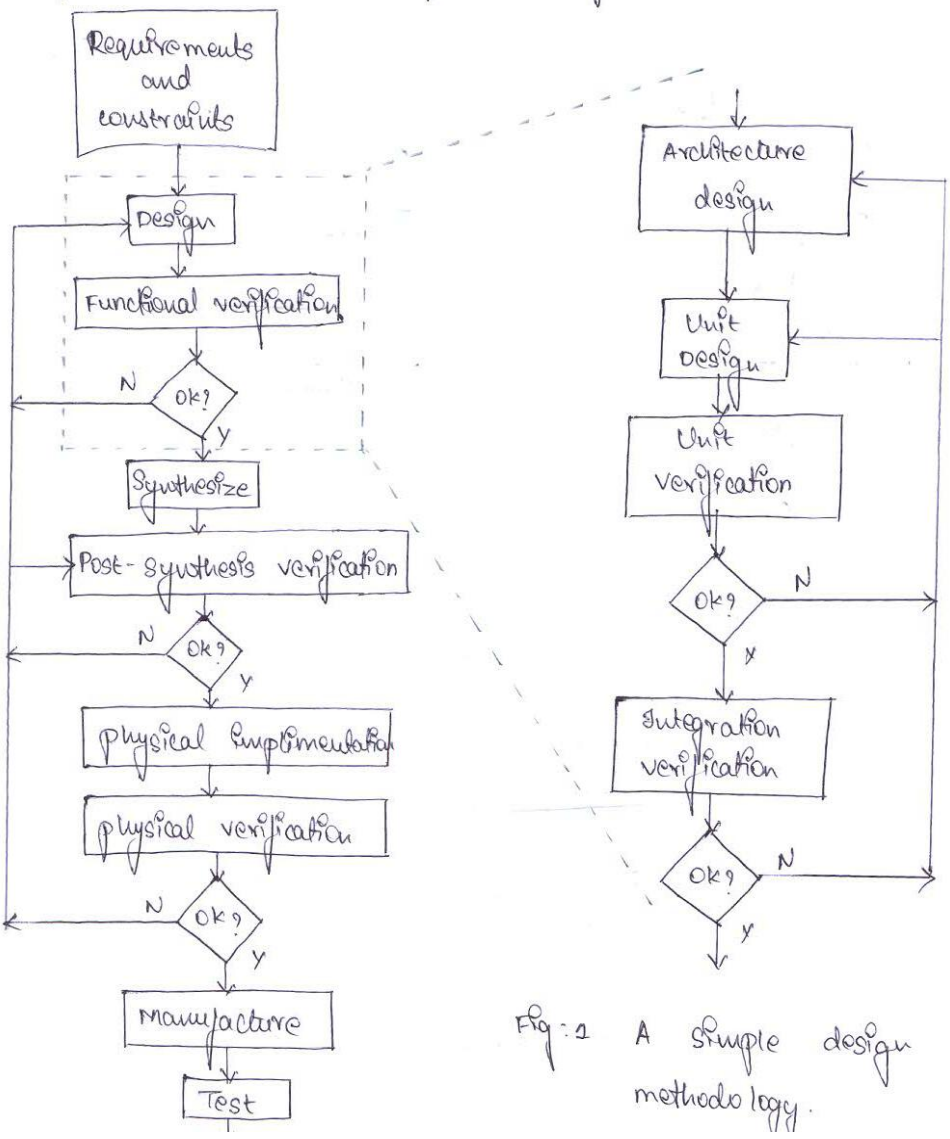
Sub:	DSDV :- DIGITAL SYSTEM DESIGN USING VERILOG							Code:	14EC666
Date:	30 / 03 / 2017	Duration:	90 mins	Max Marks:	50	Sem:	VI	Branch:	ECE/TCE

Answer Any FIVE FULL Questions

		Marks	OBE CO	OBE RBT
1	Explain the systematic process of digital system from design to Embedded system design.	[10]	CO1	L1
2	Develop a sequential circuit that has a single data input signal 'S' and produces an output 'Y'. The output is '1' whenever 'S' has the same value over three successive clock cycles, and '0' otherwise. Assume the value of 'S' for a given clock cycle is.	[10]	CO2	L2
3	Develop a digital design which down counts from '99' – '10' and repeats till input ready signal is off.	[10]	CO4	L2
4	Develop a verilog code to find a given year is a leap year or not and generate a leap year flag.	[10]	CO2	L2
5	Develop a verification module for testing either 3rd or 4th question.	[10]	CO3	L2
6	For FSM shown, develop a verilog code and plot the timing diagram. 	[10]	CO4	L3

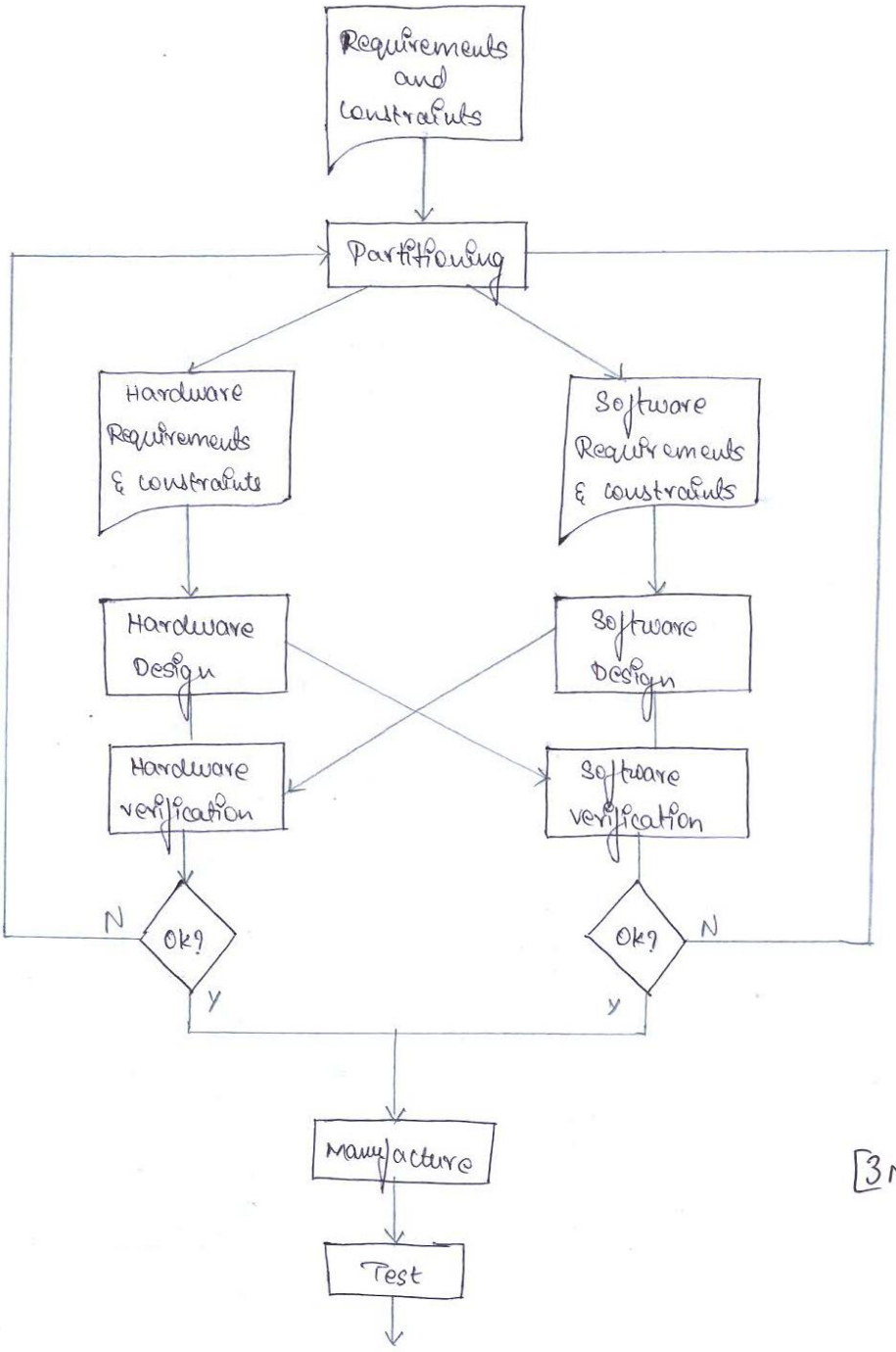
Solution

1) Explain the systematic process of digital system from design to embedded system design. [10M]



[3M]

Fig: 1 A simple design methodology.



[3M]

Fig: 2 Hardware and software Codesign

Fig 1 Illustrates a simple design methodology.
Requirements and constraints are generated externally by marketing group of a company or by customer. They include functional requirements, performance requirements, constraints on power consumption, cost and packaging.

3 Tasks: 1) Design
2) Synthesis
3) physical packing

[3M]

Each of these are followed by a verification task.

If verification fails at any stage, we must revisit a previous task to correct the error.

Fig 2 Illustrates hardware and software codesign.

Designing the hardware and software for a system together is called hardware/software codesign. Deciding which parts to put in hardware and which in the software is called partitioning.

Once the functionality has been partioned blw hardware and software, development of the two can proceed concurrently.

For these aspects of the embedded system software that depends on hardware, the abstract behavioral models from the hardware design task can be used to verify software design. A similar approach can be used to

verify parts of hardware that interface directly with a processor core.

2) Develop a sequential ckt that has a single data i/p signal 's' and it produces an o/p 'y'. The o/p is '1' whenever i/p s has same value for three successive cycles and '0' otherwise. Assume the value of 's' for the given cycle.

To compare the value of i/p to 3 successive clock cycles, the value of i/p should be save for previous two cycles To store previous values we use a pair of D-FF

The o/p Y is 1 if and only if 3 successive values of i/p are all 1 or all 0.

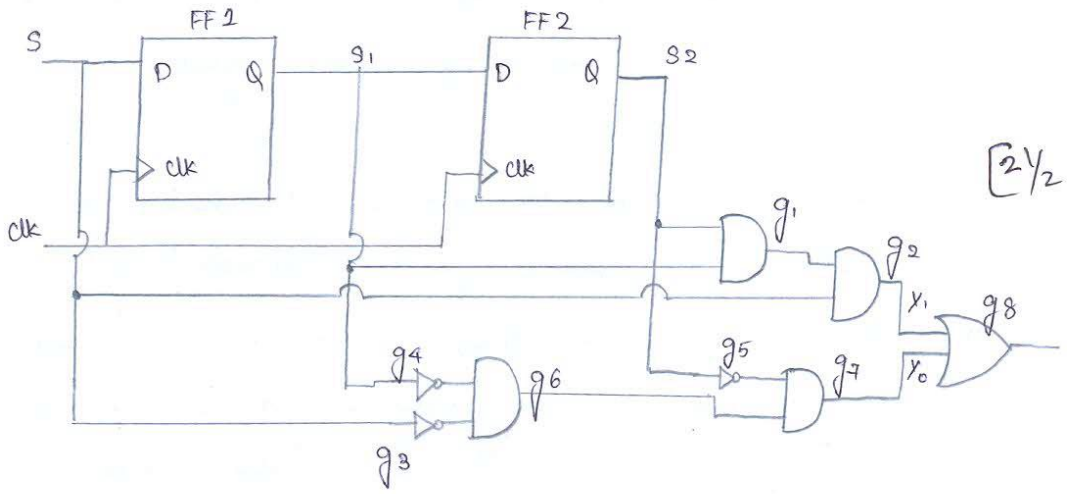
g^1, g^2 jointly determine if 3 values are 1

g^3, g^4, g^5 negate 3 values.

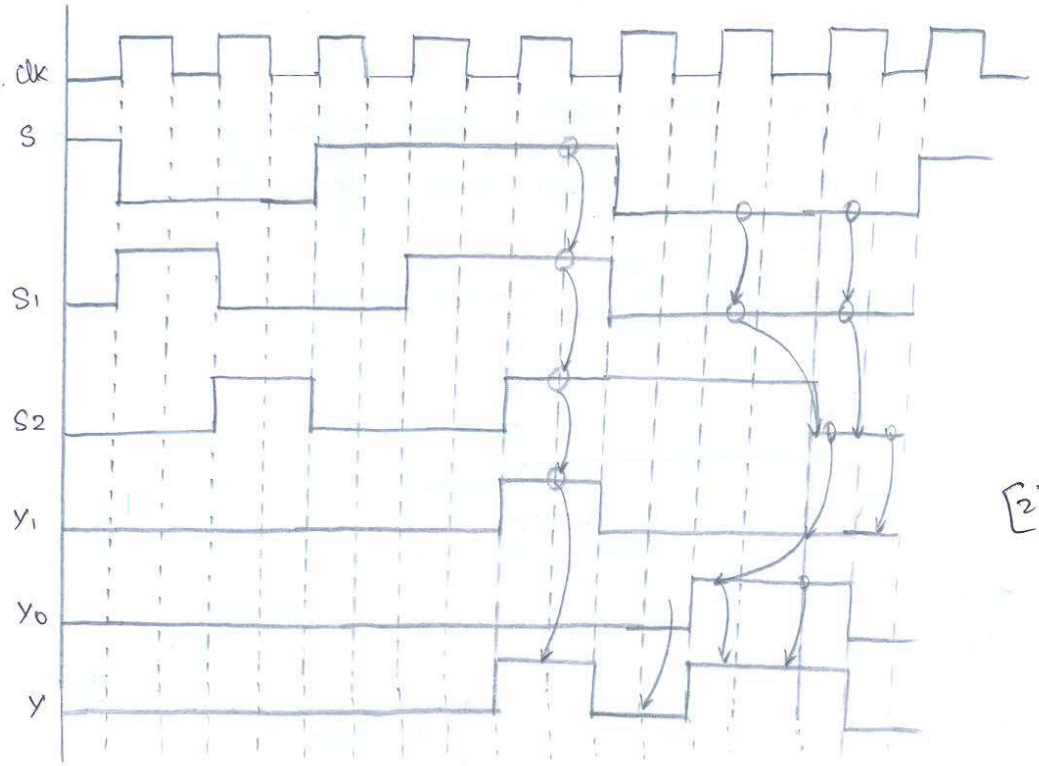
g^6, g^7 determine 3 values are 0.

g^8 combines both to yield final o/p. [5M]

The o/p of 2 FF's follow value of s over 2 cycles (delayed) ∴ When either Y_1 or Y_0 is 1, the o/p Y changes to 1.



[2½ M]



[2½ M]

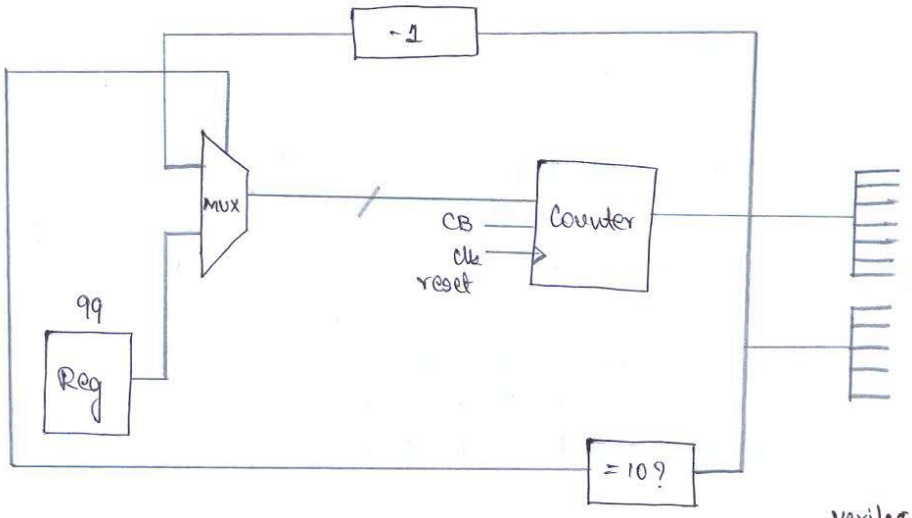
Fig: Timing diagram for the sequential comparison ckt

The circles and arrows indicate which signals are used to determine the values of other signals, leading to a 1 at the output.

When all S, S_1 and S_2 are 1, Y_1 changes to 1, indicating that S has been 1 for 3 successive cycles. Similarly when all S, S_1 and S_2 are 0, Y_0 changes to 1, indicating that S has been 0 for 3 successive cycles. When either Y_1 or Y_0 is 1, the output X changes to 1.

3) Design a digital system which down counts from 99 to 00 and suspends when input ready signal is off.

Sol:-



[5M].

Verilog PGM [5M].

4) Develop a verilog code to find a given year is a leap year or not and generate a leap year flag.

4)

```
module leapyear (output leap, input year);
begin
    if (year % 4 == 0 && year % 100 != 0 || year % 400 == 0)
        leap <= 1'b1;
    else
        leap <= 1'b0;
endmodule.
```

[5M]

5) Develop a verification for testing code:

```
module div_by4();
reg [3:0] ym, yh, yt, yo;
wire out;
);
div_by4 four (out, ym, yh, yt, yo);
initial
  begin
    # 100 ym = 4'b0000;
    # 100 yh = 4'b0000;
    # 100 yt = 4'b0000;
    # 100 yo = 4'b0000;
      .
      .
      .
  end
endmodule
```

[2M]

[4M]

```
// divisible by zero
module iszero (a,b,c,d, ym, yu, yt, yo);
input [3:0] ym, yu, yt, yo;
output A, B, C, D;
 assign #10 A = ~yo[3] & ~yo[2] & ~yo[1] & ~yo[0];
 assign #10 B = ~yt[3] & ~yt[2] & ~yt[1] & ~yt[0];
 assign #10 C = ~yu[3] & ~yu[2] & ~yu[1] & ~yu[0];
 assign #10 D = ~ym[3] & ~ym[2] & ~ym[1] & ~ym[0];
endmodule
```

-[4M]

```
module iszeroth ();
(
   reg [3:0] ym, yu, yt, yo;
   wire leave;
);
div. by zero mer (leave, ym, yu, yt, yo);
initial
    begin
      # 100  ym = 4'b0000;
      # 1000 yt = 4'b0000;
      # 1000 yu = 4'b0000;
           .
           .
           .
    end
    endmodule
```

- [6M]

) For FSM shown, develop verilog code.

code:

```
module FSM (reset, q)
( input reset;
  output [2:0] q;
  reg [2:0] q;
);
always @ (posedge reset)
begin
  if (reset == 0)
  begin                                    — [3M]
    case (q)

3'd0 : q = 3'd1;
3'd1 : q = 3'd2;
3'd2 : q = 3'd3;
3'd3 : q = 3'd4;
3'd4 : q = 3'd5;
3'd5 : q = 3'd6;                           — [3M]
3'd6 : q = 3'd7;
3'd7 : q = 3'd0;
end case.

  else
     case (q)

3'd0 : q = 3'd1;
3'd1 : q = 3'd3;
```

```
    3'd2 : q = 3'd6;
    3'd3 : q = 3'd2;
    3'd4 : q = 3'd0;
    3'd5 : q = 3'd4;
    3'd6 : q = 3'd7;
    3'd7 : q = 3'd5;
end case
end
endmodule.
```

[4M]