Digital Logic Design - Chapter 7-Verilog

(In addition to the code include design documentation either as comments in the code or separate document.)

1S. Write a Verilog code that implements function f(A,B,C) = A'.B.C + A.B.C.

Solution:

```
// Problem 1S.  f(A,B,C) = A'.B.C + A.B.C.
//-------------------------------------------------------------------------------------------
`timescale 1 ns / 1 ps

module prob6s (A, B, C, f);

input A, B, C;
wire A, B, C;

output f;
reg f;

always @ (A, B, C) begin
        f = (~A)&B&C | A&B&C;          //Use logical or "|", and "&", not "~"
end

endmodule
```

1U. Write Verilog code to implements function f(A,B,C) = A'.B.C' + A'.B'.C + A.B'.

Solution:

2S. Write Verilog code to implement a positive-edge-triggered D latch.

Solution:

```
`timescale 1ns/100ps     // time measurement unit is 1 nsec with 100 ps percision

// Design a D flip flop
// Author:  Instructor

module D_ff(clock, d, q);          // defines the input and output into module

input clock, d;                    // define input
wire clock, d;                     // declare input type

output q;                          // define output
reg q;                             // declare output

// Body of the design
always @ (posedge clock)           // executes following code at every clock rising edge
begin
     q <= d;                       // make an assignment
end

endmodule                          // end of module ;
```

2U. Write Verilog code to implement a positive-edge-triggered T latch.

Solution:

3S. Write Verilog code to implement a positive-edge-triggered JK flip flop.

Solution:

```
`timescale 1ns/100ps     // time measurement unit is 1 nsec with 100 ps percision

// Design a JK flip flop
// Author:  Instructor

module D_ff(clock,j, k, q);        // defines the input and output into module

input clock, j, k;                 // define input
wire clock, j,k;                   // declare input type

output q;                          // define output
reg q;                             // declare output

// Body of the design
always @ (posedge clock)           // executes following code at every clock rising edge
begin
     q <= j&(~q) | (~k)&q;                          // make an assignment
end

endmodule                          // end of module ;
```

3U. Write Verilog code to implement a positive-edge-triggered SR flip flop.

Solution:

4S. Write Verilog code to implement a 4-bit binary up counter.

Solution:

```
`timescale 1ns/100ps     // time measurement unit is 1 nsec with 100 ps percision

// Design a 4-bit up counter
// Author:  Instructor

module Counter( clk, count );

input clk ;
wire clk ;

output count ;
reg [3:0] count = 4'b000 ;


//Counting section
always@(posedge clk)
   begin
         count = count | 4'b0001;
   end

endmodule    //Counter
```

4U. Write Verilog code to implement a 4-bit binary down counter

Solution:

5S. Write Verilog code that outputs a signal that has a frequency equal to 1/2, 1/4, 1/8 & 1/16 of input clock frequency depending on the value of two input select bits.

Solution:

```
// Divide by 2, 4, 8 clock.
//-------------------------------------------------------------------------------------------------
`timescale 1 ns / 1 ps

module prob6s (clk, sel, out);

input clk, sel;
wire clk, [1:0]sel;

output out ;
reg out =0;

integer count=0;

always @ (posedge clk) begin
        count = count + 1;
        case (sel)
                0       : out <= out ^ 1;
                1       : if (count == 2) begin
                                out <= out ^ 1;
                                count = 0;
                          end
                2       : if (count == 4) begin
                                out <= out ^ 1;
                                count = 0;
                          end
                3       : if (count == 8) begin
                                out <= out ^ 1;
                                count = 0;
                          end
                default: count = 0;
        endcase
end

endmodule
```

5U. Write Verilog code to implement a 3-bit binary to 8-line decoder (active high)

Solution:

6U. Write Verilog code to implement an 8-line (active low) to 3-bit binary encoder.

Solution:

7U. Write Verilog code that implements the 3-way T-intersection with each side having red, yellow and green lights. Assume that car on the right has priority and only one side has green light at any given time and yellow light is on for 2 to 4 seconds.

Solution:

8U. Write Verilog code to implement an ALU which adds, subtracts and multiplies 8-bit numbers.

Solution: