Sequential Logic Timing (20 Points)

1. Use the following specifications for the pipeline of digital logic below.

\[\begin{align*}
\text{Flip-Flops} & \quad \text{Combo Logic} \\
T_{ccq} &= 30 \text{ ps} & T_{cd} &= 5 \text{ ps} \\
T_{pcq} &= 75 \text{ ps} & T_{pd} &= 50 \text{ ps} \\
T_{setup} &= 90 \text{ ps} & \\
T_{hold} &= 40 \text{ ps} &
\end{align*}\]

a. What does \(T_{ccq}\) stand for?

b. What does \(T_{pcq}\) stand for?

c. What does \(T_{setup}\) stand for?

d. What does \(T_{hold}\) stand for?

e. If the clock period is 200 ps, is there a setup time violation, why or why not?

f. If the clock period is 400 ps, is there a hold time violation, why or why not?
Verilog (15 Points)

2. Draw the logic described by this Verilog module. Include all information possible. What does this mystery block do?

```verilog
module MysteryBlock(
  input [7:0] A, B,
  input C,
  output [8:0] Z);

  wire [7:0] Bout;

  assign Bout = C ? ~B : B;
  assign Z = A + Bout + C;

endmodule
```
Verilog (15 Points)

i. Given the Verilog code of the Moore state machine below draw the state diagram.

ii. Is the state register a asynchronous reset or a synchronous reset?

iii. How many flip flops would be needed to build this state machine?

iv. Assume that you start in S0 and the following stream of bits comes in for i_in:

```
I_in 1 0 1 0 1 0 0 1 0 1
```

Label which state the machine would be in under each bit in the stream listed above.

```verilog
module moore_machine ( 
    i_in, 
    i_clk, 
    i_reset, 
    o_out 
); 

//constants 
parameter S0 = 2'00; 
parameter S1 = 2'01; 
parameter S2 = 2'10; 
parameter S3 = 2'11; 

input i_in; 
input i_clk; 
input i_reset; 
output o_out; 

reg [1:0] state; 
reg [1:0] state_next; 

//state register 
always @(posedge i_clk, negedge i_reset) 
begin 
    if (i_reset) 
        begin 
            state <= 0; 
        end 
    else 
        begin 
            state <= state_next; 
        end 
end 

//next state logic 
always @(*) 
begin 
    case (state) 
        S0: if (i_in) state_next = S1; 
            else state_next = S0; 
        S1: if (i_in) state_next = S1; 
            else state_next = S2; 
        S2: if (i_in) state_next = S3; 
            else state_next = S0; 
        S3: if (i_in) state_next = S1; 
            else state_next = S0; 
        default: state_next = S0; 
    endcase 
end 

//output logic 
assign o_out = (state == 33); 

endmodule //moore_machine 
```