

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Sciences

Introductory Digital Systems Lab (6.111)

Quiz - Spring 2004

Prof. Anantha Chandrakasan

Student Name: _____

Problem 1 (16 Points): _____

Problem 2 (24 Points): _____

Problem 3 (24 Points): _____

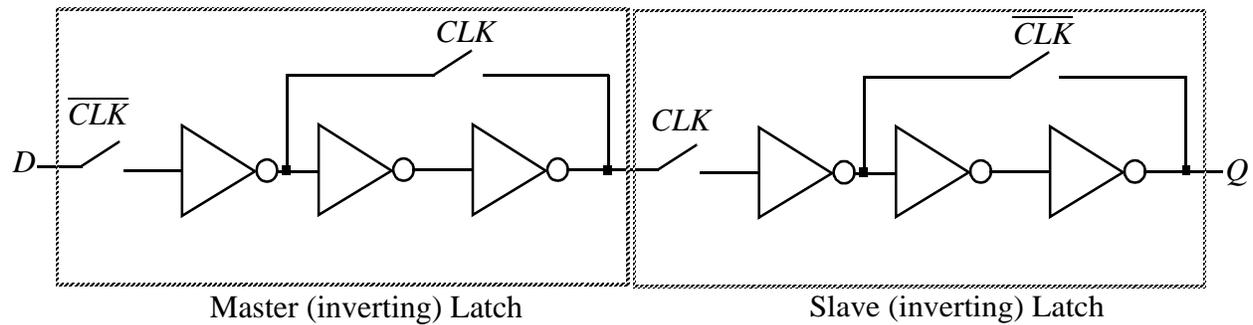
Problem 4 (16 Points): _____

Problem 5 (20 Points): _____

Total (100 Points): _____

Problem 1: Sequential Building Block Characterization:

Consider the following implementation of a register. Assume that an inverter has a delay of 1 and that the switch is ideal. The switch is closed when the control input is high.



(a) What type of register is shown above (circle one)?(4 points)

positive edge-triggered

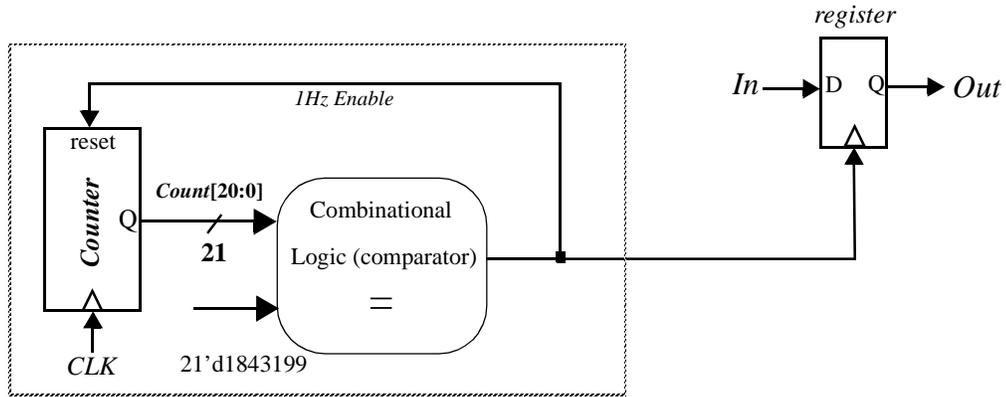
negative edge-triggered

(b) What is the setup time (t_{su}), hold time (t_{hold}), and propagation (t_{cq}) delay. Assume that CLK and \overline{CLK} are ideal (i.e., the delay to derive \overline{CLK} from CLK is zero). (12 points)

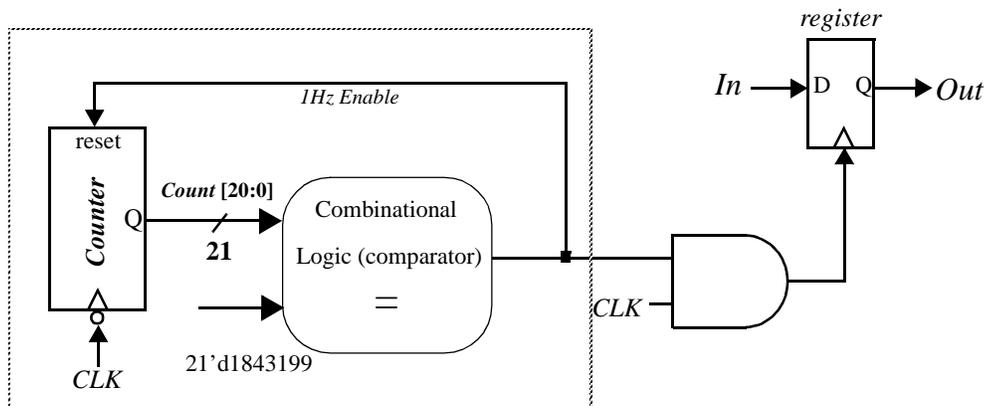
Problem 2: Clock Gating Circuit

Assume for the entire problem that the counter reset is synchronous.

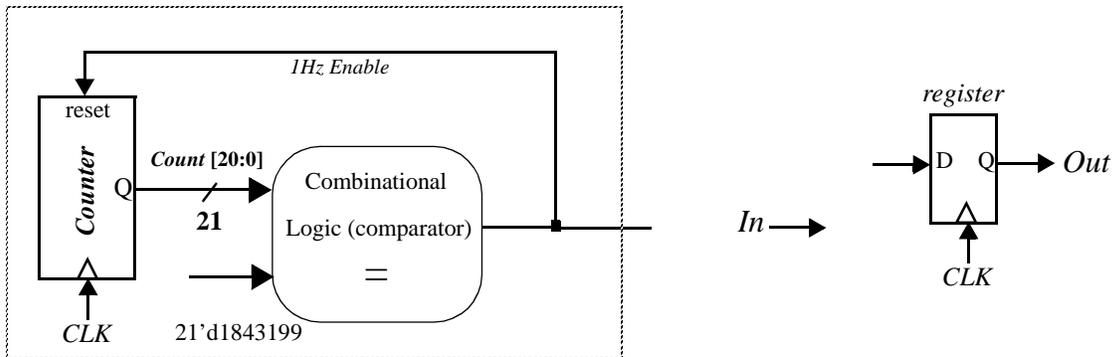
(a) Consider the one second clock from lab2. The 1Hz enable is derived from the circuit below (the clock is the 1.8432Mhz clock). Assume that a global reset signal puts the count to 0 (the circuit is not shown here). Suppose the 1Hz enable is used to clock other circuits that need to be updated once per second (e.g., the register shown below). Identify the main problem with this circuit. (4 points)



(b) The following is a simple modification that is proposed to solve the problem of part (a). The counter is negative edge-triggered and the AND gate is introduced before the register clock. Assume that the AND gate is ideal (i.e., has zero delay). Does this circuit fix the problem of part (a)? If so, under what conditions does this circuit function properly? If not, explain why. (6 points)



(c) The solution of (b) is not a synchronous one (i.e., a single global clock to every register). Propose a fix to the problem in part (a) using a single clock (CLK). That is, both the counter and register are clocked from the same clock (CLK). The signal In should be loaded into the register once per second. You may add additional components in the diagram below. (6 points)



(d) For each of the following signals, specify if glitches are allowed. (circle one) (8 points)

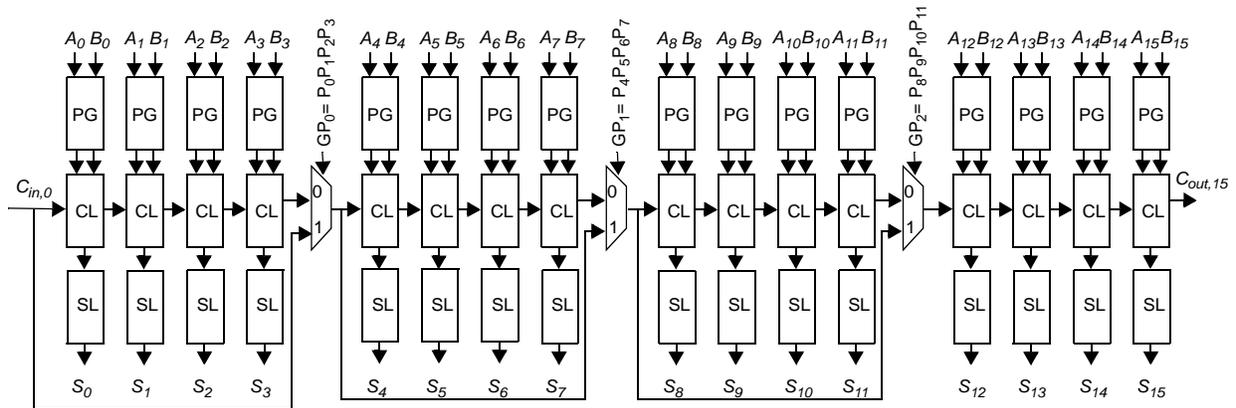
- | | | |
|--|----------------------|----------------------|
| (i) Data lines to the M6264 when writing (when \overline{W} is low, \overline{G} is high, chip is enabled) as long as it settles well before \overline{W} goes high | Glitches Not Allowed | Glitches are Allowed |
| (ii) Address lines to the M6264 when reading (when \overline{W} is high, \overline{G} is low, chip is enabled) as long as it settles well before the output data is required | Glitches Not Allowed | Glitches are Allowed |
| (iii) R/\overline{W} for the AD670 (ADC) when \overline{CE} and \overline{CS} are low | Glitches Not Allowed | Glitches are Allowed |
| (iv) Data inputs for the AD558 (DAC) when \overline{CE} and \overline{CS} are low | Glitches Not Allowed | Glitches are Allowed |

Problem 3: Arithmetic

(a) Consider the following 16-bit Carry Bypass Adder. Assume that the Sum Logic (SL) block for stage i has as inputs P_i , G_i , and $C_{in,i}$. Assume the following delays:

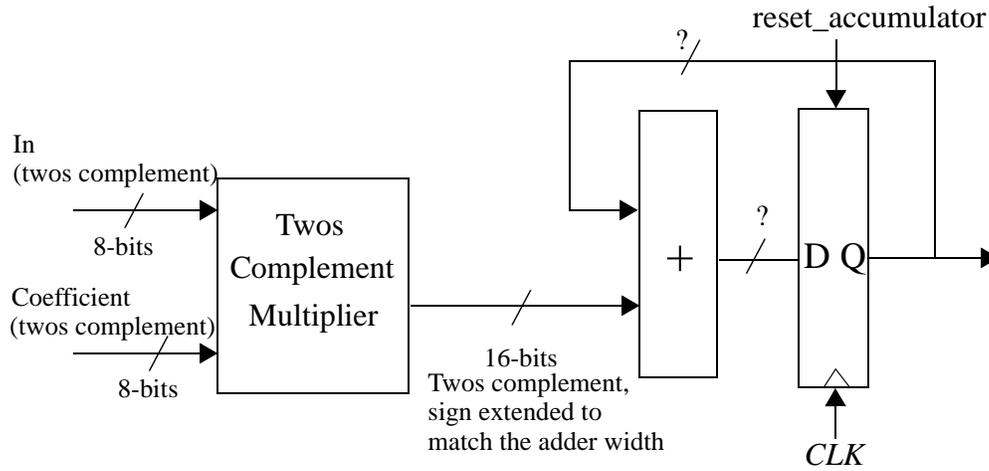
- Delay to produce the P_i , G_i signals from the A_i , B_i inputs is 1 (i.e., the delay of the PG logic block)
- Delay to compute $C_{out,i}$ from P_i , G_i and $C_{in,i}$ inputs is 1 (i.e., the delay of the Carry Logic (CL) block)
- Delay to compute S_i from $C_{in,i}$, P_i , G_i being valid is 2 (i.e., the delay of the Sum Logic (SL) block)
- Delay for the 2:1 multiplexor is 2
- Delay to compute the group propagate, GP_i , is 1

Highlight the critical path for this 16-bit adder directly on figure below (6 points)

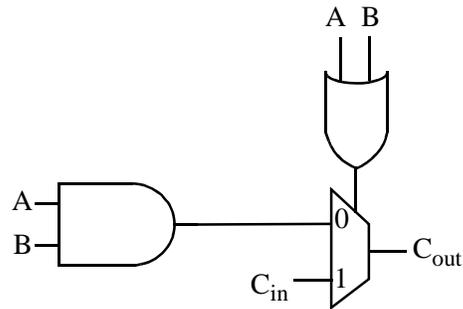


(b) Based on the structure in (a), assuming a block size equal to 4, write down an equation for the delay of an N input adder, where N is a multiple of 4. (6 points)

(c) Consider a 64-tap filter implemented using an accumulator architecture as shown below. How many bits will be required for the adder? (6 points)

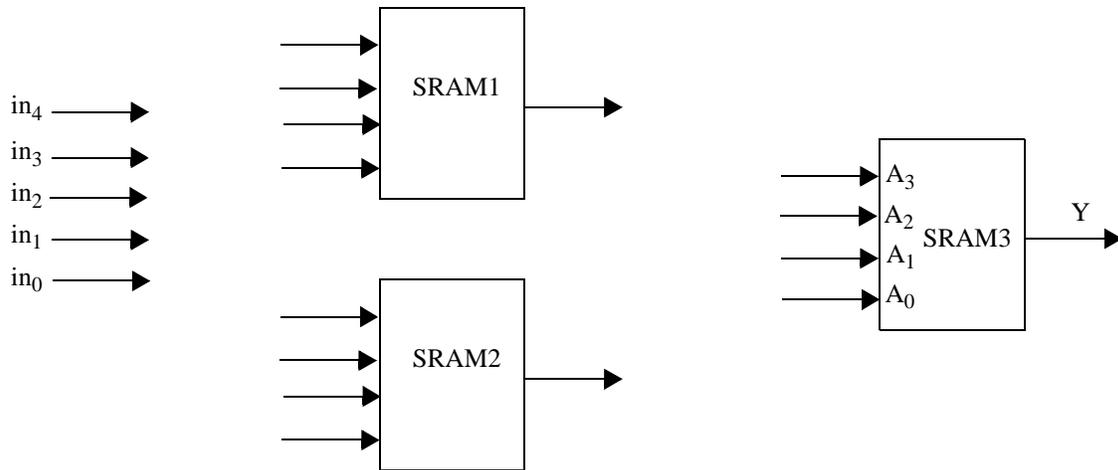


(d) There is an error in the following implementation of the carry out logic. Identify the error and propose a simple solution (it might involve changing the logic). (6 points)



Problem 4: FPGA

(a) Consider an FPGA architecture with three identical 4-input SRAM based lookup tables. For this problem do not worry about the circuits to perform the write operation into the SRAM. Each SRAM is shown to have 4 inputs (i.e., address bits) and one output bit. Draw the circuit diagram for using the three SRAMs shown below to implement an arbitrary function Y of 5-input variables $in_4, in_3, in_2, in_1, in_0$? Do not add any additional components. **(8 points)**



(b) What is the function implemented by SRAM3 corresponding to your solution in part 4(a)? Explain. **(8 points)**

PROBLEM 5: Verilog

(a) Consider the follow piece of code to implement a **synchronizer** (with two registers). Does the code implement the required functionality? (explain). If the code does not properly implement a synchronizer, modify the code as necessary. **(6 points)**

```
input in;
reg q1, q2;

always @ (posedge clk)
begin
  q1 = in;
  q2 = q1;
end
```

(b) The following Verilog code should synthesize to a purely combinational logic function (3:1 multiplexor, where we don't care about the case when sel = 11). Does it? If not, fix the code so that there are **no memory elements**. **(6 points)**

```
module mux_3to1 (a, b,c, sel, out);

input [1:0] sel;
input a,b, c;
output out;
reg out;

always @ (a or b or c or sel)
begin
  case (sel)
    2'b00: out = a;
    2'b01: out = b;
    2'b10: out = c;
  endcase
end
endmodule
```

(c) A simple digital system consists of a four state FSM which increments a 4-bit synchronous counter when the FSM is in state 10. An asynchronous reset signal is used to force both the state machine and the counter to all zeros whenever it is high. Identify the major problem with this code when synthesized and propose a fix (i.e., make the necessary modifications to the code below. You may add a control signal). **(8 points)**

```
module simple_fsm(clk, reset, count);
  input reset, clk;
  output [3:0] count;
  reg [3:0] count;
  reg [1:0] state;
  reg [1:0] next;

  always @ (posedge clk or posedge reset) begin
    if (reset)
      begin
        state <= 2'b00;
      end
    else
      begin
        state <= next;
      end
  end

  always @ (state or reset) begin
    if (reset)
      count = 4'b0000;

    case (state)
      2'b00: next = 2'b01;
      2'b01: next = 2'b10;
      2'b10: begin
        next = 2'b11;
        count = count + 1;
      end
      2'b11: next = 2'b00;
    endcase
  end //always

endmodule
```