The University of Michigan
Department of Electrical Engineering and Computer Science

EECS 270 Fall 2003

Exam #2 Solutions

Name:_____ UM ID:_____

For all questions, show all work that lead to your answer.

Problem #	Possible Points	Points Earned
1	10	
2	12	
3	15	
4	16	
5	22	
6	25	
Total	100	

I have neither given nor received any unauthorized aid on this exam.

Signed: _____

1. *Short Answer: 10 Points Total*

(a: 2 pts) A ring counter that counts from 57 to 0 will have __58__ D flip-flops,
but a binary counter that counts from 57 to 0 will have _6__ D flip-flops.

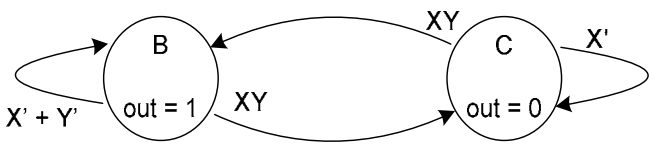
(b: 2 pts) In general, which of the following types of finite state machine has a smaller circuit? (circle one)

Mealy Moore?
either answer is acceptable

(c: 2 pts) Which sequential gates are faster? (circle one)

latches flip-flops

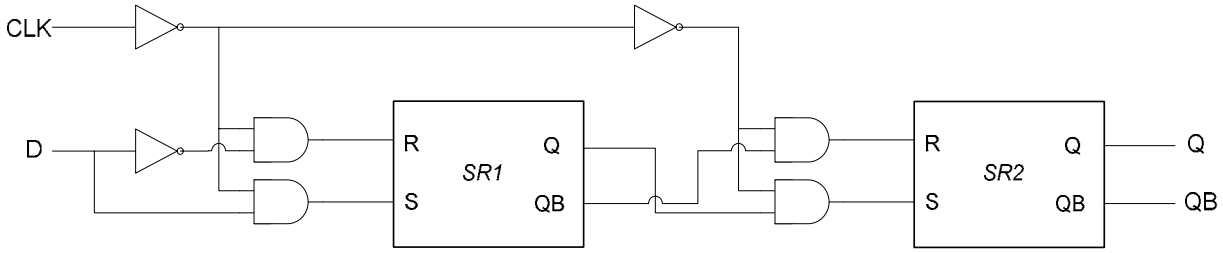
(d: 4 pts) The following state diagram for a FSM with two inputs X and Y is ambiguous:



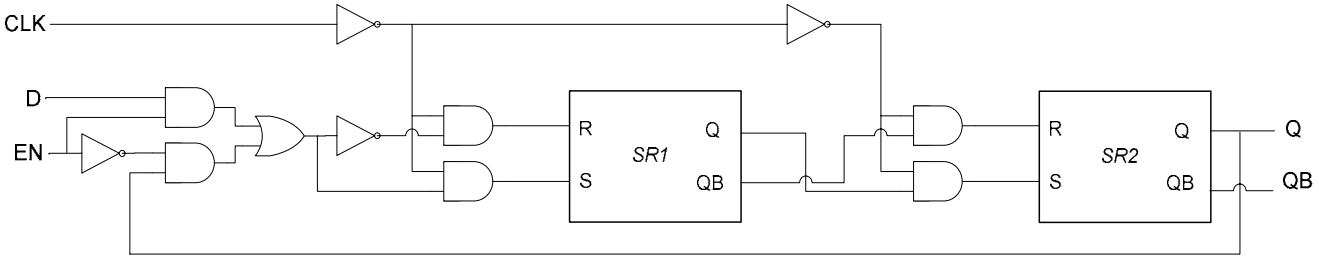
True **False**

2. *Flip-flop Design: 12 Total Points*

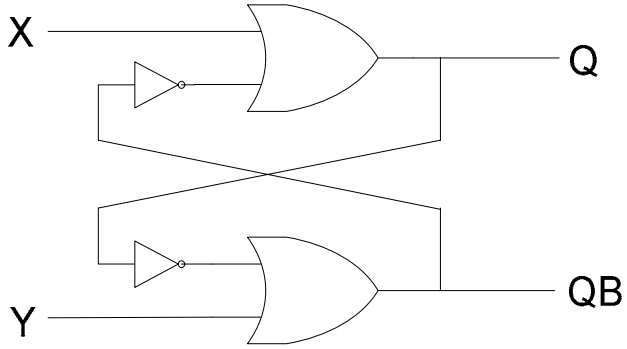
(a: 6 pts) Construct a positive-edge-triggered D flip-flop from the two SR latches below and any number of inverter/nand/nor/and/or gates.



(b: 6 pts) Copy your design from 2(a) below and add to it an ENABLE signal that is active high. You may need to add additionally gates to your design.



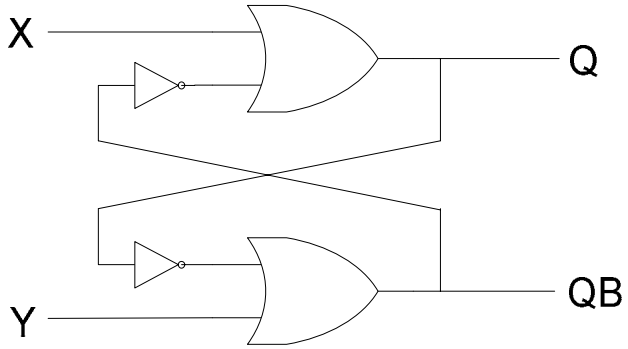
3. *Latch Design: 15 Total Points*

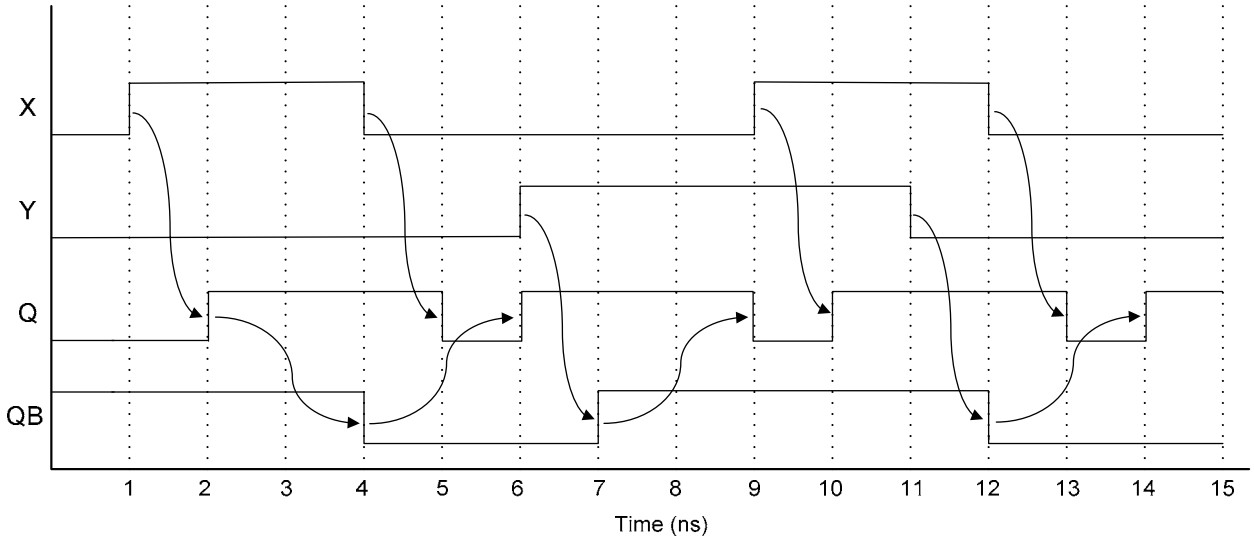


(a: 5 pts) Given the above SR latch design above, complete the following truth table.

X	Y	Q	QB
0	0	Last Q	Last QB
0	1	0	1
1	0	1	0
1	1	1	1

(b: 5 pts) Assuming that all gates have a 1ns rising and falling delay, complete the timing diagram below showing all causality arrows. The latch figure has been copied for convenience.



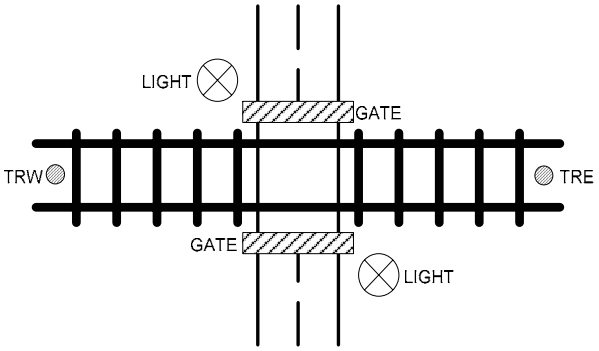


(c: 5 pts) Assign new names for inputs X and Y of this latch such that they are more semantically meaningful. Select from the following set of names: {S, S_L, R, R_L}.

A better name for input X is __S___.

A better name for input Y is __R___.

4. *State Machine Design: 16 Total Points*

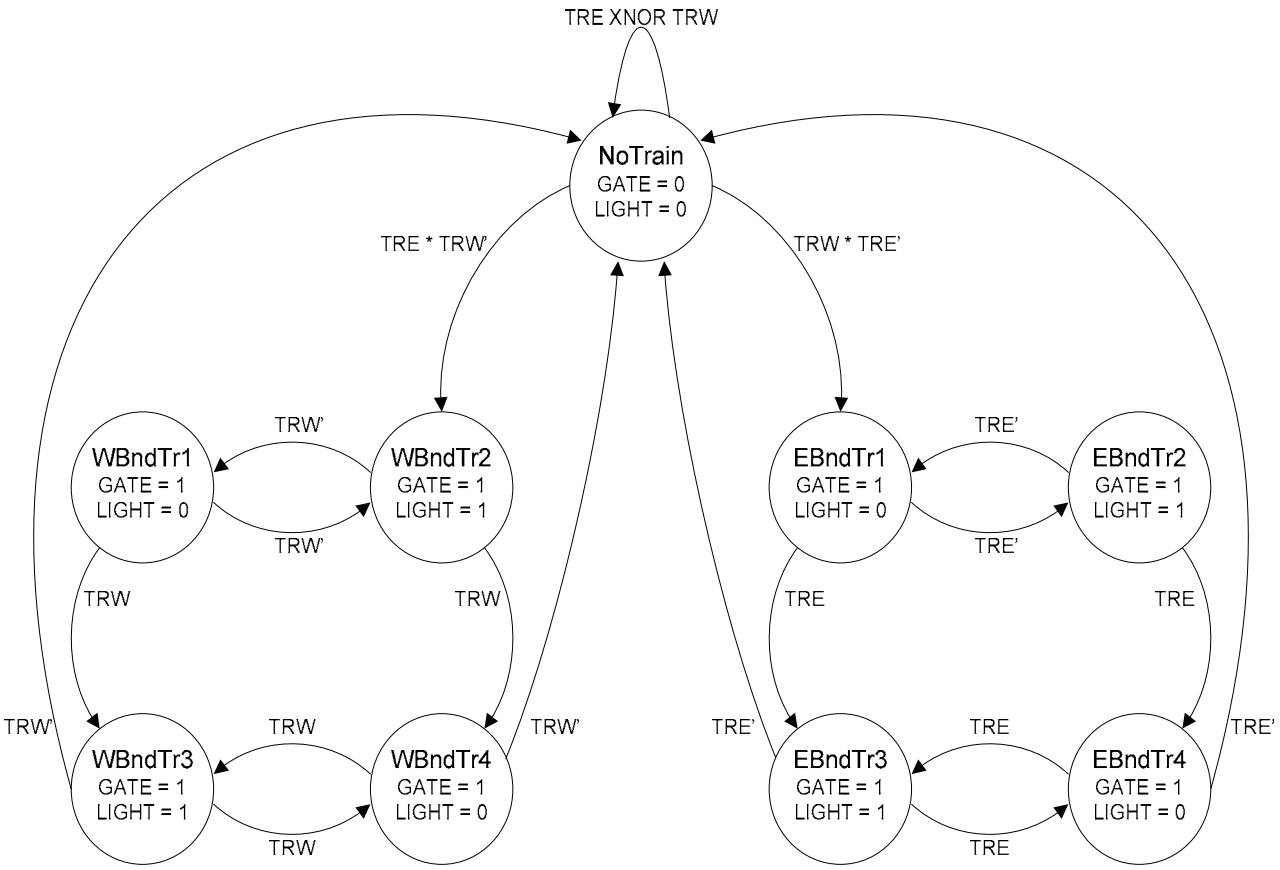


Consider the street/train track intersection shown above, where trains may be traveling eastbound or westbound. A Moore-type finite state machine is needed to control the crossing gate as well as the warning lights, based on input from two train sensors, TRE and TRW. The sensor values will be asserted (1) if and only if a train is immediately passing over them. The gates will be closed if and only if GATE is asserted (1), and the warning lights will be on if and only if LIGHT is asserted (1).

Safety considerations dictate that your finite state machine should behave as follows: When no trains are detected, the gates should remain open with the warning lights off. Whenever a train is detected on either sensor, the gates should close and the warning lights should be *flashing* until the opposite sensor is first asserted and then de-asserted. Only then can the gates open and the warning lights turn off. For simplicity, you may assume that the two sensors will never be asserted simultaneously, and therefore your design may handle the input combination TRE*TRW arbitrarily.

(see next page)

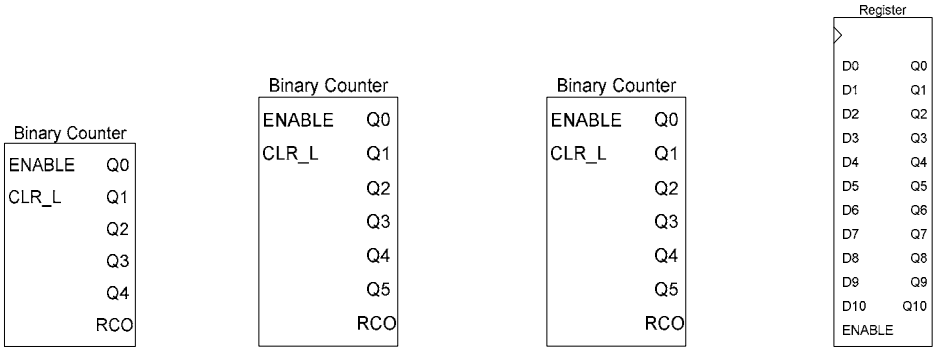
Draw the state diagram for this state machine.



Note that it is possible to combine the functionality of WBndTr3 and EBndTr3, as well as WBndTr4 and EBndTr4. This is not shown here for simplicity.

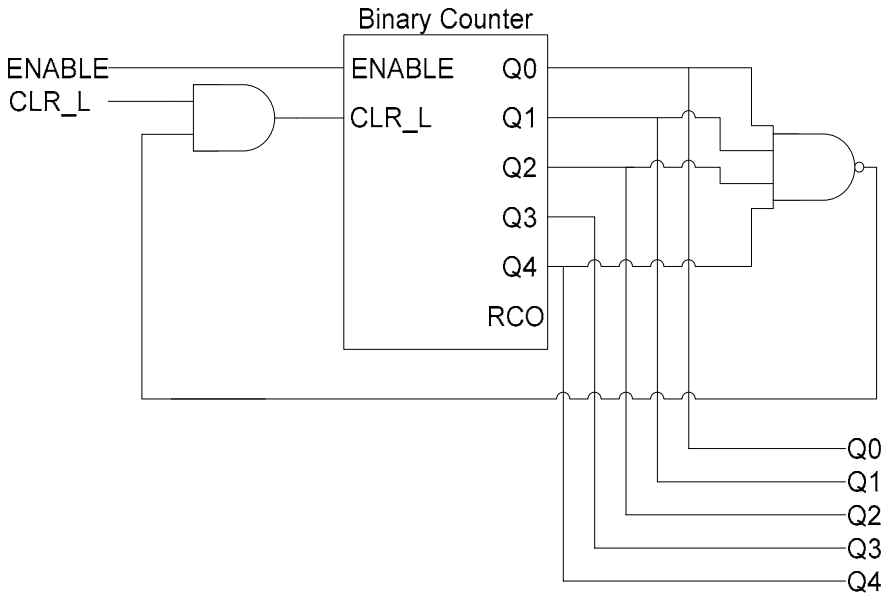
5. *Sequential Block Design: 22 Total Points*

You are given three binary counters with CLR_L and ENABLE inputs, and RCO, one of which is a 5 bit-counter, and two of which are 6-bit counters. You are also given one 11-bit register with ENABLE input. Each device is shown below and can be used once, in addition with as many combinational gates as required in the following design:

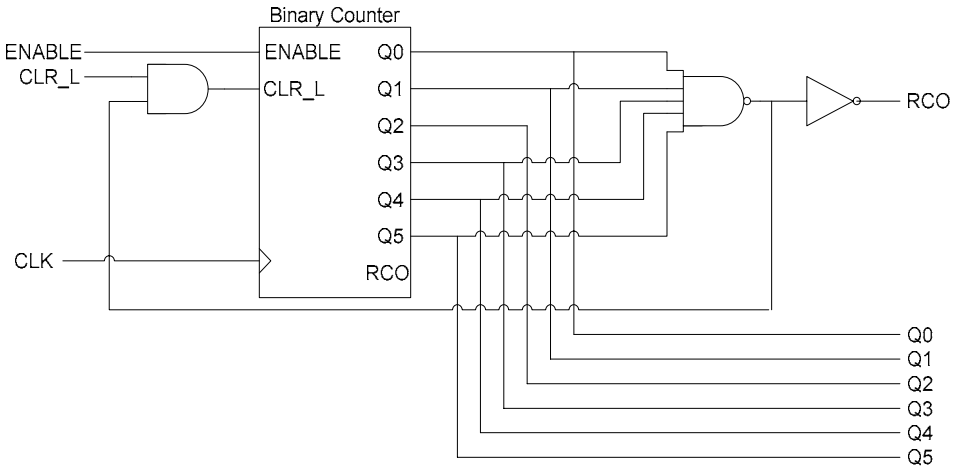


One extra point for given for effort on this problem.

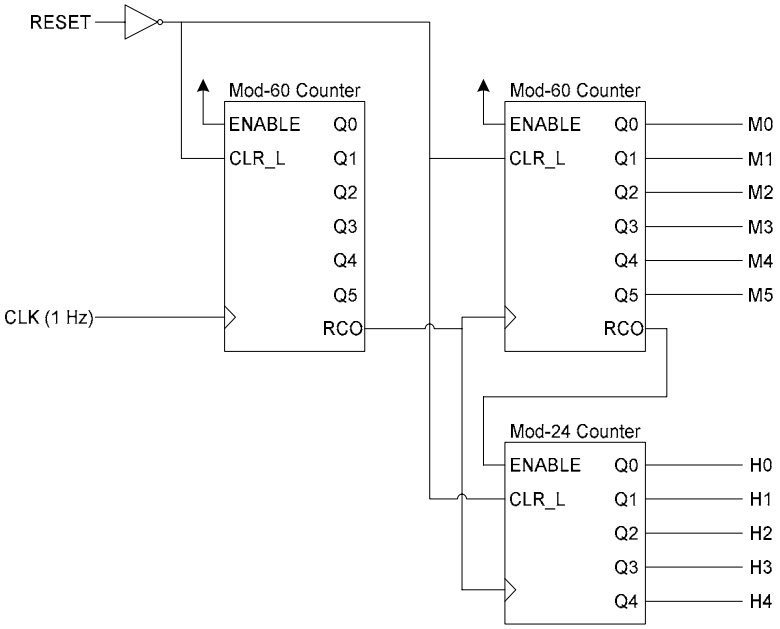
(a: 5 pts) Design a 5-bit mod-24 counter with a counting sequence of 0-23 with ENABLE and CLR_L inputs.



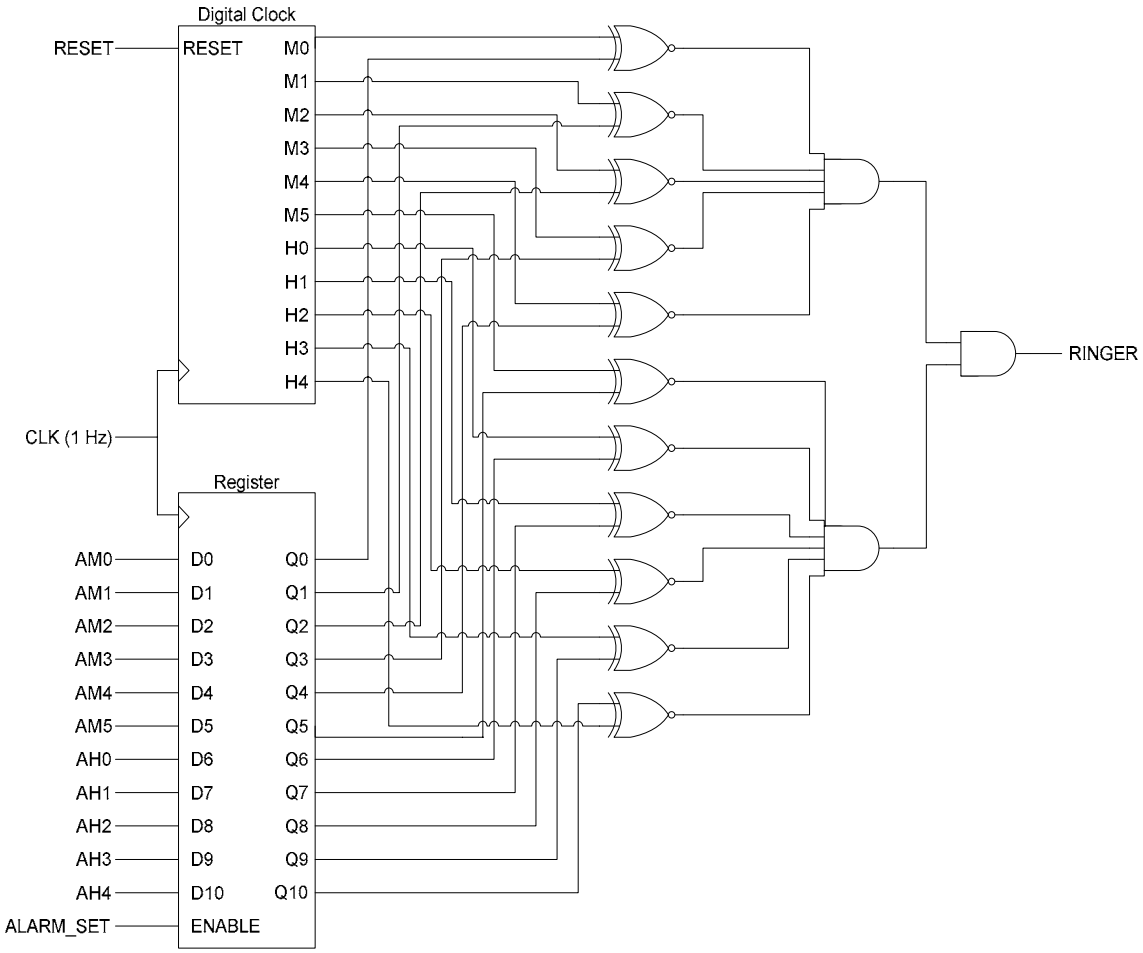
(b: 5 pts) Design a 6-bit mod-60 counter with a counting sequence of 0-59 with an ENABLE and CLR_L inputs and RCO output.



(c: 5 pts) Assuming that the global clock generator ticks once every second, use your above designs (in high-level block diagram form) to implement a 24-hour (i.e. military time) digital clock with RESET input and 11 outputs that represent the current time (in binary), with five bits (H4-H0) representing the hour and 6 bits (M5-M0) representing the minute.

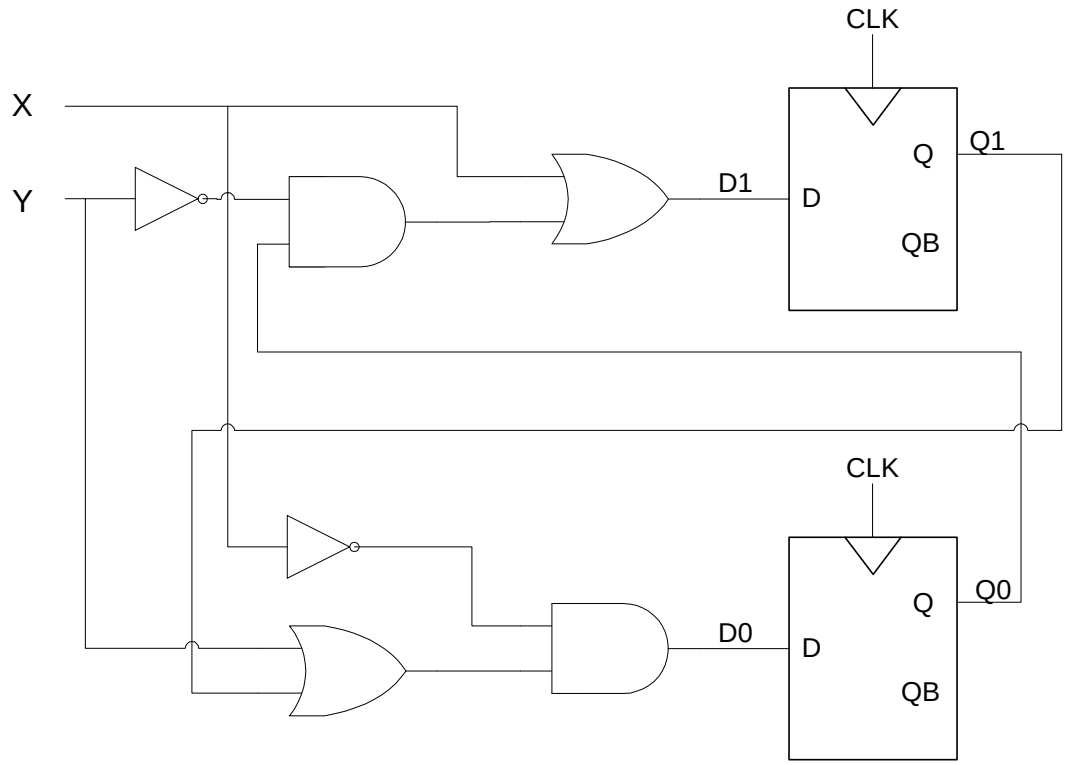


(d: 6 pts) Starting with your digital clock (again, use a block diagram), add an alarm. Your new design should have 11 inputs which are used to set the alarm time: AM5-AM0 for the minute and AH4-AH0 for the hour (both in binary), as well as an ALARM_SET input, which, when asserted, will cause the circuit to store the values on alarm time inputs. Your design should also have a RINGER output that should stay high for one minute when the current time is equal to the alarm time.



6. *Finite State Machine Analysis: 27 Total Points*

Consider the FSM circuit below:



(a: 5 pts) Derive the excitation equations for D1 and D0.

D1 = X + (Y'·Q0)
D0 = X'· (Y + Q1) = X'·Y + X'· Q1

(b: 8 pts) Given the following state assignments, complete the transition table and state table shown below.

State Assignments:

S	Q1	Q0
A	0	1
B	1	0
U1	0	0
U2	1	1

Transition Table:

		XY			
Q1	Q0	00	01	11	10
0	1	10	01	10	10
1	0	01	01	10	10
0	0	00	01	10	10
1	1	11	01	10	10
			Q1* Q0*		

State Table:

	XY			
S	00	01	11	10
A	B	A	B	B
B	A	A	B	B
U1	U1	A	B	B
U2	U2	A	B	B
		S*		

(c: 7 pts) A state machine is safe if from all unused states, a used state will be reached in one clock cycle under all possible input conditions. For unused states U1 = 00 and U2 = 11, determine the input combinations for which this state machine is unsafe (if any).

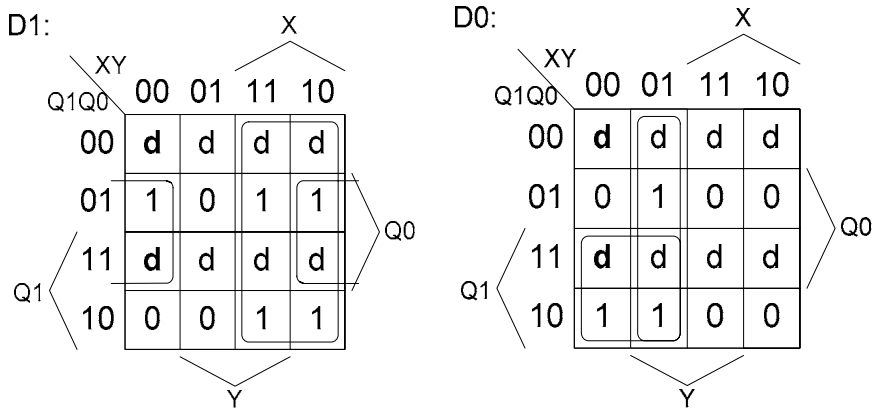
Add U1 and U2 to transition and state tables (see above).

U1 is unsafe under input condition XY = 00, because the next state under this input combination is U1, an unused state.

U2 is unsafe under input condition XY = 00, because the next state under this input combination is U2, an unused state.

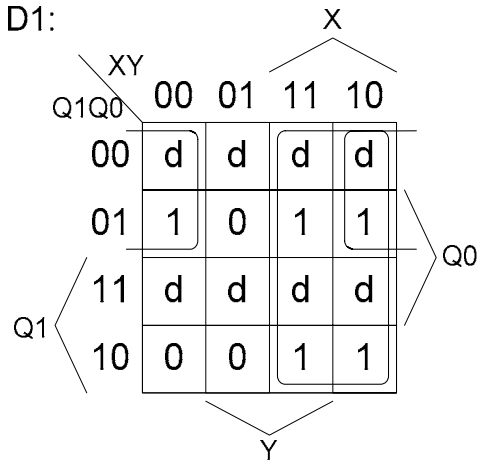
(d: 7 pts) Change the circuit so that it will be safe. Use as few additional gates as possible. *(Hint: look at the K-map for D1 and D0)* Do not redraw the circuit – instead show your new expressions for D1 and D0.

Looking at the K-maps for both D1 and D0:



The unsafe cells are marked in bold for D1 and D0.

The simplest way to modify the K-maps so that U1 and U2 will transition to used next states is by moving the Y'Q0 prime implicant of D1 to Y'Q1', as shown below:



Now, the next state for U1 under input combination XY = 00 will be 10 (state B). The next state for U2 under input combination XY = 00 will be 10 (state A). No other next states will change, so our state machine is now safe.

The new safe expressions for D1 and D0 are:

D1 = X + Y'Q1'
D0 = X'Y + X'Q1 (unchanged).

The changes to D1 require no additional gates to implement.