Ve270 Introduction to Logic Design

Homework 7

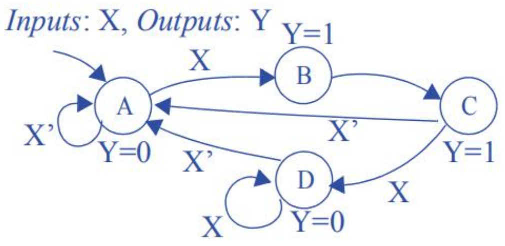
Assigned: July 7, 2020

Due: July 14, 2020, 2:00pm.

A pop quiz will be given on July 16.

1. Problem 3.24. (10 points)

3.24 Draw a state diagram for an FSM that has an input X and an output Y. Whenever X changes from 0 to 1, Y should become 1 for two clock cycles and then return to 0 -- even if X is still 1. (Assume for this problem and all other FSM problems that an implicit rising clock is ANDed with every FSM transition condition.)



2. Implement a circuit for the FSM designed in Problem 3.24. (15 points)

Encode the states:

A:	00
B:	01
C:	10
D:	11

state table.

S_1	S_0	X	n_1	n_0	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	1	1	0

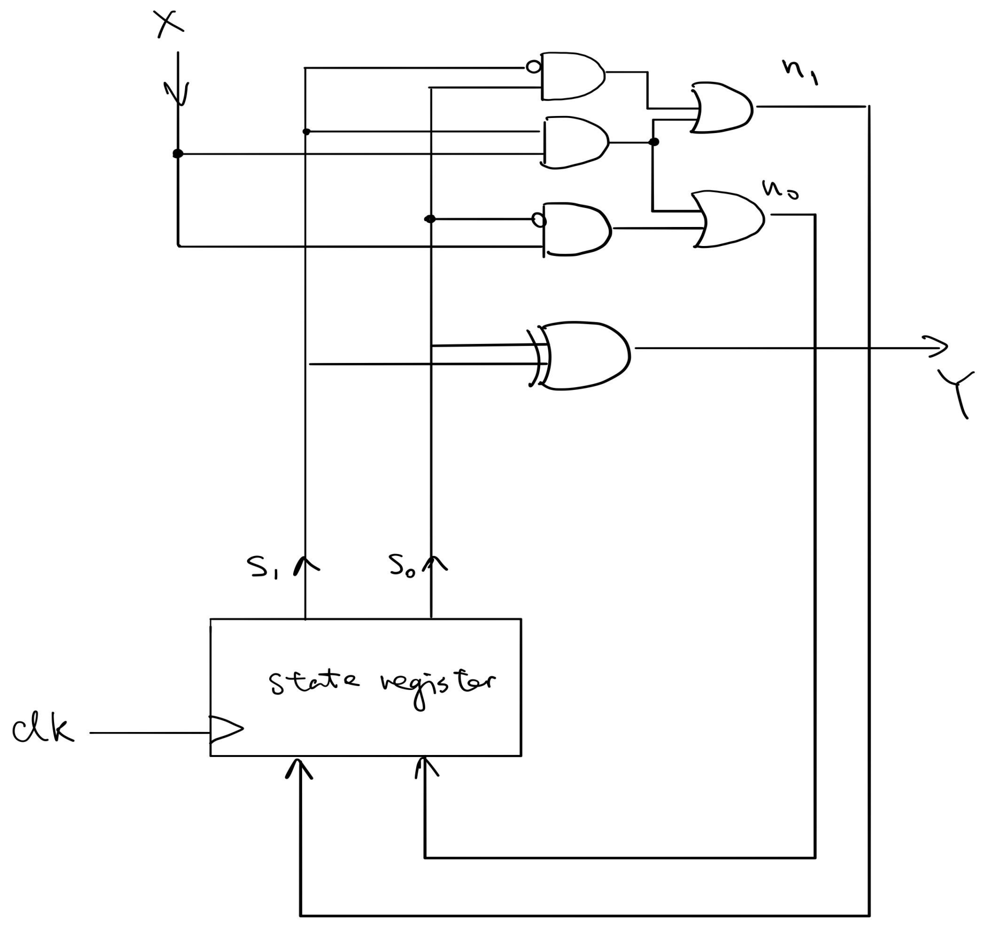
combinational logic:

$n_1 = S_1' S_0 + S_1 X$

$n_0 = S_1 X + S_0' X$

$Y = S_0 \oplus S_1$

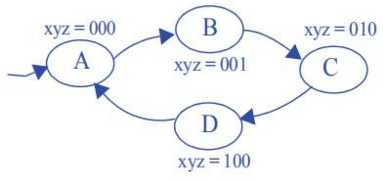
Always simplify your circuit when possible, including 'X'.



3. Problem 3.25 (10 points)

3.25 Draw a state diagram for an FSM with no inputs and three outputs x, y, and z. xyz should always exhibit the following sequence: 000, 001, 010, 100, repeat. The output should change only on a rising clock edge. Make 000 the initial state.

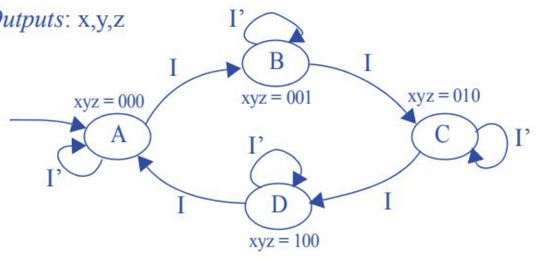
Inputs: None, *Outputs*: x,y,z



4. Problem 3.26. (10 Points)

3.26 Do Exercise 3.25, but add an input I that can stop the sequence when set to 0. When input I returns to 1, the sequence resumes from where it left off.

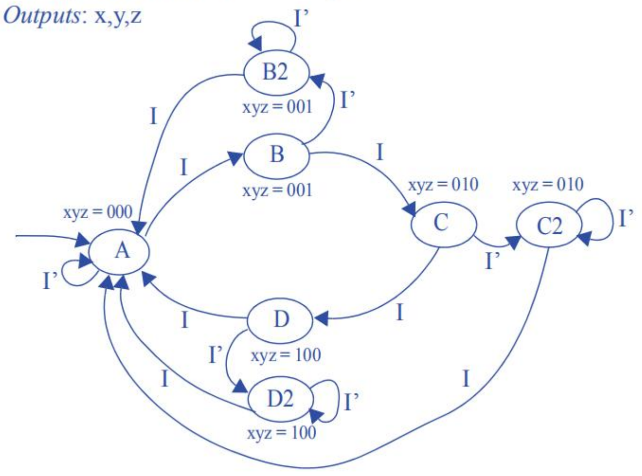
Inputs: I, *Outputs*: x,y,z



5. Problem 3.27. (10 Points)

3.27 Do Exercise 3.25, but add an input I that can stop the sequence when set to 0. When I returns to 1, the sequence starts from 000 again..

Inputs: I, *Outputs*: x,y,z



6. Implement a circuit for the FSM designed in Problem 3.27. (15 points)

A: 000 B: 010 B2: 011. C: 100 C2: 101
D: 110 D2: 101.

	S_2	S_1	S_0	I	n_2	n_1	n_0	x	y	z
A	0	0	0	0	0	0	0	0	0	0
	0	0	0	1	0	1	0	0	0	0
B	0	1	0	0	0	1	1	0	0	1
	0	1	0	1	1	0	0	0	0	1
B2	0	1	1	0	0	1	1	0	0	1
	0	1	1	1	0	0	0	0	0	1
C	1	0	0	0	0	1	0	0	1	0
	1	0	0	1	1	0	0	0	1	0
C2	1	0	1	0	1	0	1	0	1	0
	1	0	1	1	0	0	0	0	1	0
D	1	1	0	0	1	1	1	1	0	0
	1	1	0	1	0	0	0	1	0	0
D2	1	1	1	0	1	1	1	1	0	0
	1	1	1	1	0	0	0	1	0	0

$n_2 = s_2' s_1 s_0' I + s_2 s_0' I' + s_2 s_1' s_0' + s_2 s_1' I'$

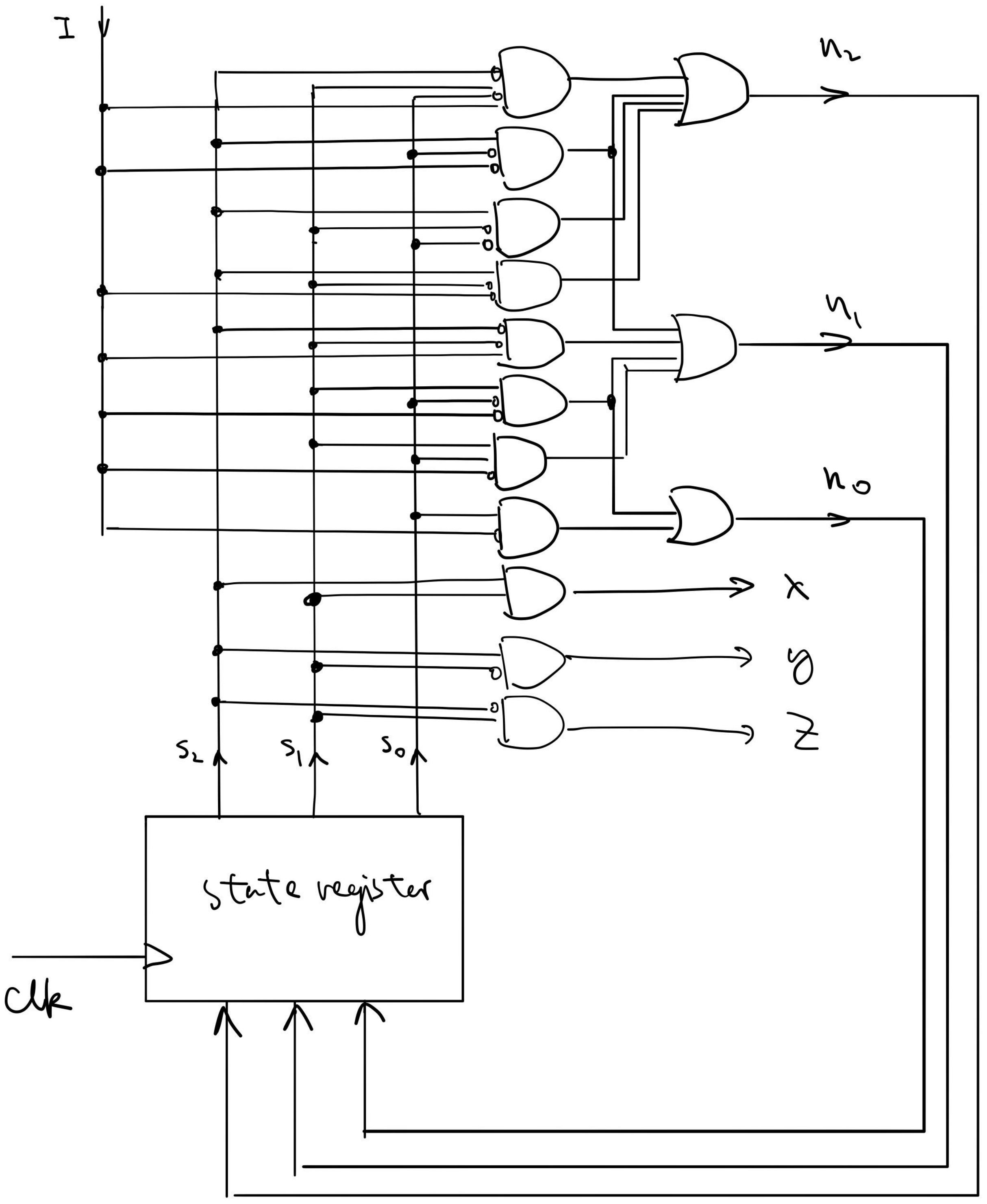
$n_1 = s_2' s_1' I + s_1 s_0' I' + s_1 s_0 I' + s_2 s_0' I'$

$n_0 = s_0 I' + s_1 s_0' I'$

$x = s_2 s_1$

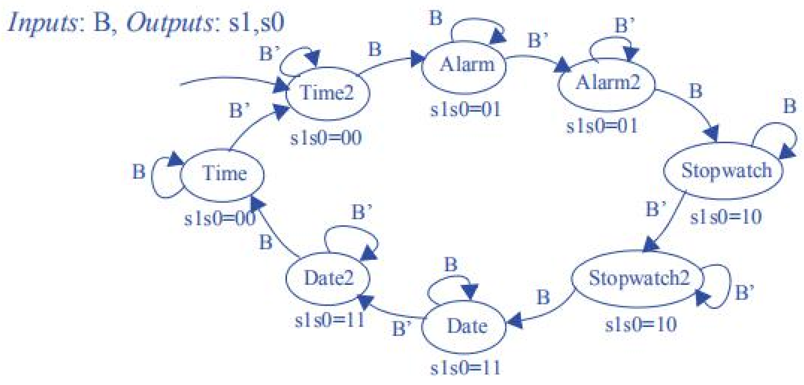
$y = s_2 s_1'$

$z = s_2' s_1$


I
n2
n1
n0
x
y
z
s2
s1
s0
state register
clk

7. Problem 3.28, show design steps, equations, and draw schematics. (20 points)

3.28 A wristwatch display can show one of four items: the time, the alarm, the stopwatch, or the date, controlled by two signals s1 and s0 (00 displays the time, 01 the alarm, 10 the stopwatch, and 11 the date—assume s1s0 control an N-bit mux that passes through the appropriate register). Pressing a button B (which sets B = 1) sequences the display to the next item. For example, if the presently displayed item is the date, the next item is the current time. Create a state diagram for an FSM describing this sequencing behavior, having an input bit B, and two output bits s1 and s0. Be sure to only sequence forward by one item each time the button is pressed, regardless of how long the button is pressed—in other words, be sure to wait for the button to be released after sequencing forward one item. Use short but descriptive names for each state. Make displaying the time be the initial state.



Time : 000	Time2 : 001
Alarm: 010	Alarm2: 011
Stopwatch : 100	Stopwatch2 : 101
Date : 110	Date2 : 111.

	P_2	P_1	P_0	B	n_2	n_1	n_0
T	0	0	0	0	0	0	1
	0	0	0	1	0	0	0
T2	0	0	1	0	0	0	1
	0	0	1	1	0	1	0
A	0	1	0	0	0	1	1
	0	1	0	1	0	1	0
A2	0	1	1	0	0	1	1
	0	1	1	1	1	0	0
S	1	0	0	0	1	0	1
	1	0	0	1	1	0	0
S2	1	0	1	0	1	0	1
	1	0	1	1	1	1	0
D	1	1	0	0	1	1	1
	1	1	0	1	1	1	0
D2	1	1	1	0	1	1	1
	1	1	1	1	0	0	0

$n_2 = P_2(P_1P_0B)' + P_2'P_1P_0B$

$n_1 = P_1P_0' + P_1B' + P_1'P_0B$

$n_0 = B'$

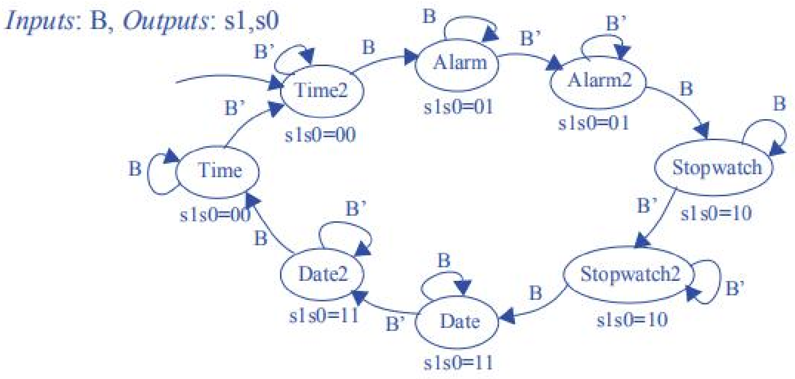
$S_1 = P_2$

$S_0 = P_1$

8. Problem 3.42 (10 points)

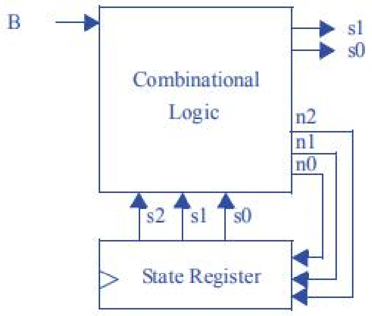
3.42 Using the process for designing a controller, convert the FSM you created for Exercise 3.28 to a controller, implementing the controller using a state register and logic gates.

Step 1 - Capture the FSM



The FSM was created during Exercise 3.28.

Step 2A - Set up the architecture



Step 2B - Encode the states

A straightforward encoding is Time2=000, Alarm=001, Alarm2=010, Stopwatch=011, Stopwatch2=100, Date=101, Date2=110, Time=111.

Step 2C - Fill in the truth table

Inputs				Outputs				
s2	s1	s0	B	n2	n1	n0	s1	s0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1
0	0	1	1	0	0	1	0	1
0	1	0	0	0	1	0	0	1
0	1	0	1	0	1	1	0	1
0	1	1	0	1	0	0	1	0
0	1	1	1	0	1	1	1	0
1	0	0	0	1	0	0	1	0
1	0	0	1	1	0	1	1	0
1	0	1	0	1	0	1	1	1
1	0	1	1	1	1	0	1	1
1	1	0	0	1	1	0	1	1
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0

Step 2D - Implement the combinational logic

n2 = s2’s1s0B’ + s2s1’ + s2s0’ + s2B
n1 = s1s0’ + s1B + s2s0B + s2’s1’s0B’
n0 = s0’B + s2’B + s1B + s2s1’s0B’
s1 = s2s0’ + s2s1’ + s2’s1s0
s0 = s1 XOR s0