Tünker Tercan
171044032

CSE 232 SPRING 2020

HOMEWORK 3

1. Compute the clock period for the following clock frequencies.

a. 50 kHz (early computers)

1/50,000 = 0.00002 = 20 μs (microseconds)

b. 300 MHz (Sony Playstation 2 processor)

$1/300,000,000 = 3.33 \cdot 10^{-9}$ s = 3.33 ns (nanoseconds)

c. 3.4 GHz (Intel Pentium 4 processor)

$1/3,400,000,000 = 2.94 \cdot 10^{-10}$ = 294 ps (picoseconds)

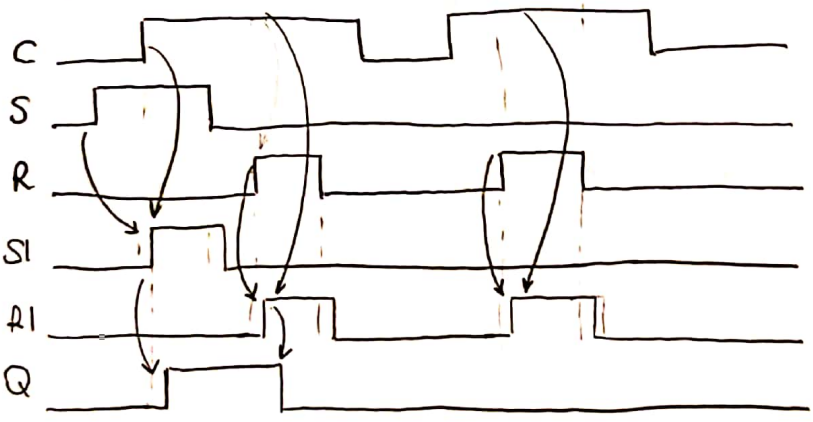
d. 10 GHz (PCs of the early 2010s)

$1/10,000,000,000 = 1.10^{-10}$ = 100 ps (picoseconds)

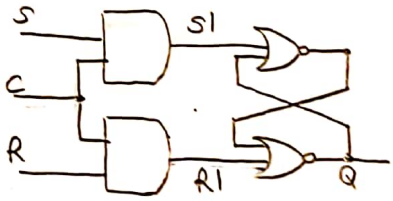
e. 1 THz (PCs of the future ?)

$1/1,000,000,000,000 = 1.10^{-12}$ = 1 ps (picoseconds)

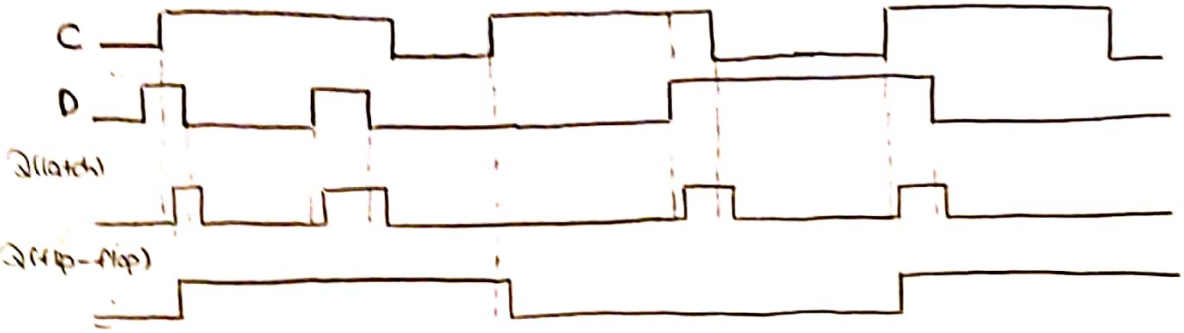
2. Trace the behavior of a level-sensitive SR latch for the input pattern in below figure. Assume S1, R1 and Q initially 0. Complete the timing diagram for S1, R1, Q. Assuming logic gates have a tiny but non-zero delay.

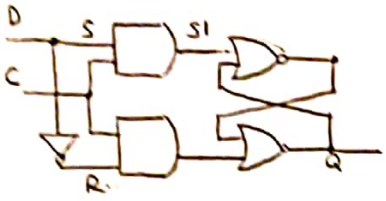


Level-Sensitive SR latch

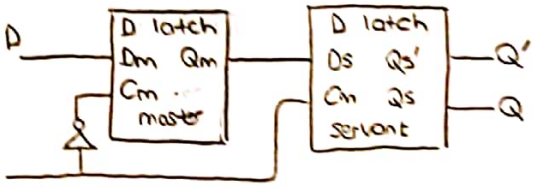


3. Compare the behavior of D latch and D flip-flop devices by completing the timing diagram adding Q(latch) and Q(flip-flop) in below figure. Provide a brief explanation of the behavior of each device. Assume each device initially stores a 0.





D latch
(bit storage that stores C=1)



D flip-flop
(bit storage that stores on clock edge)

4. FSMs with following number of states, indicate smallest number of bits for a register representing those states:

a. 4 $2^2 = 4$ 2 bits required

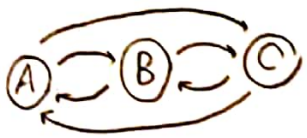
b. 8 $2^3 = 8$ 3 bits required

c. 9 $2^3 < 9 < 2^4$ 4 bits required

d. 23 $2^4 < 23 < 2^5$ 5 bits required

e. 900 $2^9 < 900 < 2^{10}$ 10 bits required

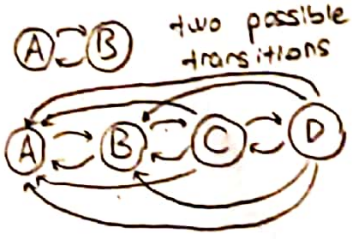
5. If an FSM has N states, what is number of possible transitions that could exist in the FSM? Assume that no pair of states has more than one transition in the same direction, and that no state has a transition point back to itself. Assuming there are a large number of inputs, meaning the number of transitions is not limited by the number of inputs? Hint: try for small n, and then generalize.



6 possible transitions

Then, for each N states, it can go to N-1 states. The maximum possible is

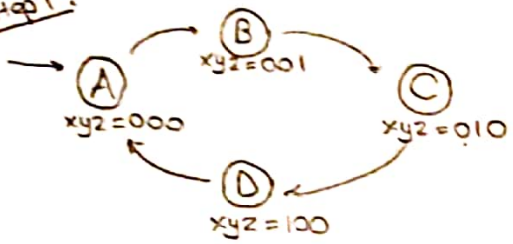
$$N*(N-1)$$



possible
12 transitions

6. 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 initial state. Using the process for designing a controller, convert FSM to a controller, stopping once you have created the truth table and derive boolean expressions.

Step 1:

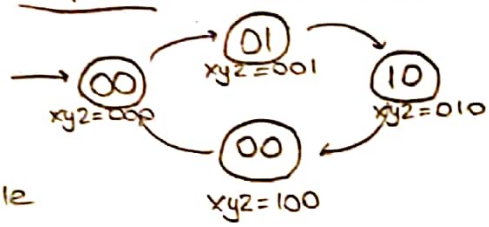


Step 2A: 2 bit state register (4 states)

No input, x, y, z outputs

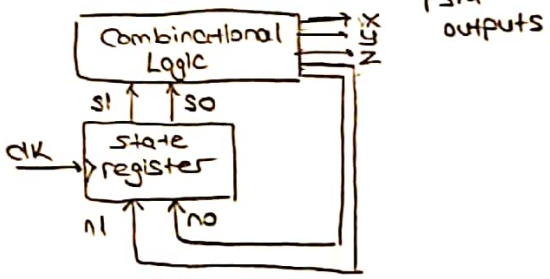
Next state signal n0, n1

Step 2B: Encode the states



Step 2C: Fill in the truth table

	Inputs		Outputs				
	S1	S0	n1	n0	x	y	z
A	0	0	0	1	0	0	0
B	0	1	1	0	0	0	1
C	1	0	1	1	0	1	0
D	1	1	0	0	1	0	0



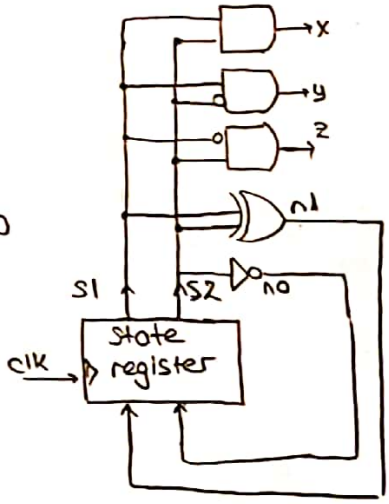
Step 2D: Implement gate logic

$n1 = S1 \text{ XOR } S0$

$n2 = S1'S0' + S1S0'$

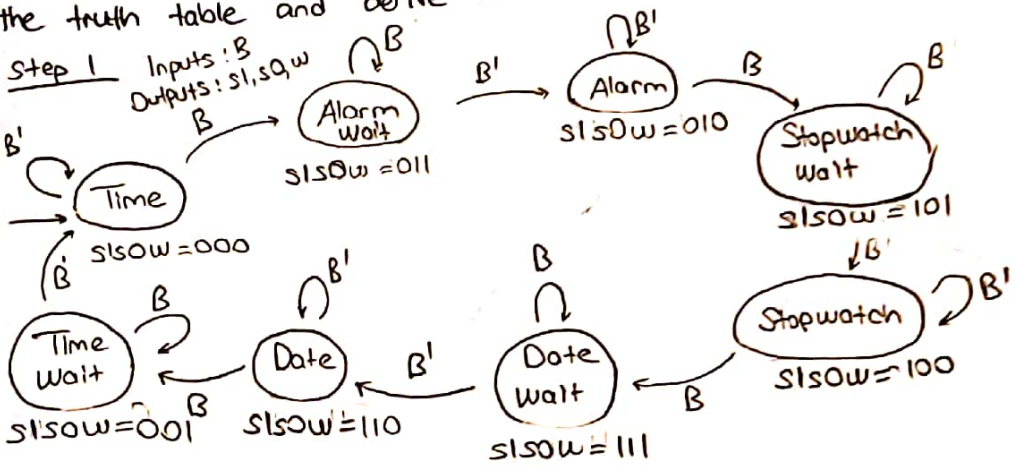
$n2 = S0'$

$x = S1S0 \quad y = S1S0' \quad z = S1'S0$



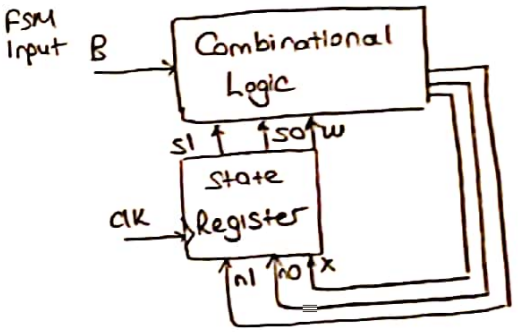
7. A wristwatch display can show one of four items; the time, the alarm, the stopwatch, or the date, controlled by two signals s1 and s0 (00 displays time, 01 the alarm, 10 the stopwatch and 11 the date - assume s1s0 control an N-bit mux that passes through appropriate register). Pressing a button B (which sets B=1) sequences the display to the next item. For example, if the presently displayed item is 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 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. Using the process for designing a controller, convert the FSM to a controller, stopping once you have created the truth table and derive boolean expressions.

Step 1 Inputs: B
Outputs: s1, s0, w



Step 2A : Set up architecture

n-bit mux for state
one bit register for wait signal
next signal n0, n1, x
input b



Step 2c : Fill the truth table

	inputs				outputs		
	s1	s0	w	b	n1	n0	x
Time wait	0	0	1	0	0	0	0
	0	0	1	1	0	0	1
Time	0	0	0	0	0	0	0
	0	0	0	1	0	1	1
Alarm Wait	0	1	1	0	0	1	0
	0	1	1	1	0	1	1
Alarm	0	1	0	0	0	1	0
	0	1	0	1	1	0	1
Stopwatch wait	1	0	1	0	1	0	0
	1	0	1	1	1	0	1
Stopwatch	1	0	0	0	1	0	0
	1	0	0	1	1	1	1
Date Wait	1	1	1	0	1	1	0
	1	1	1	1	1	1	1
Date	1	1	0	0	1	1	0
	1	1	0	1	0	0	1

$n1 = s1s0' + s1b' + s1w' + s1's2w'b$

$n0 = s0b' + s0w + s0'w'b$

$x = b$