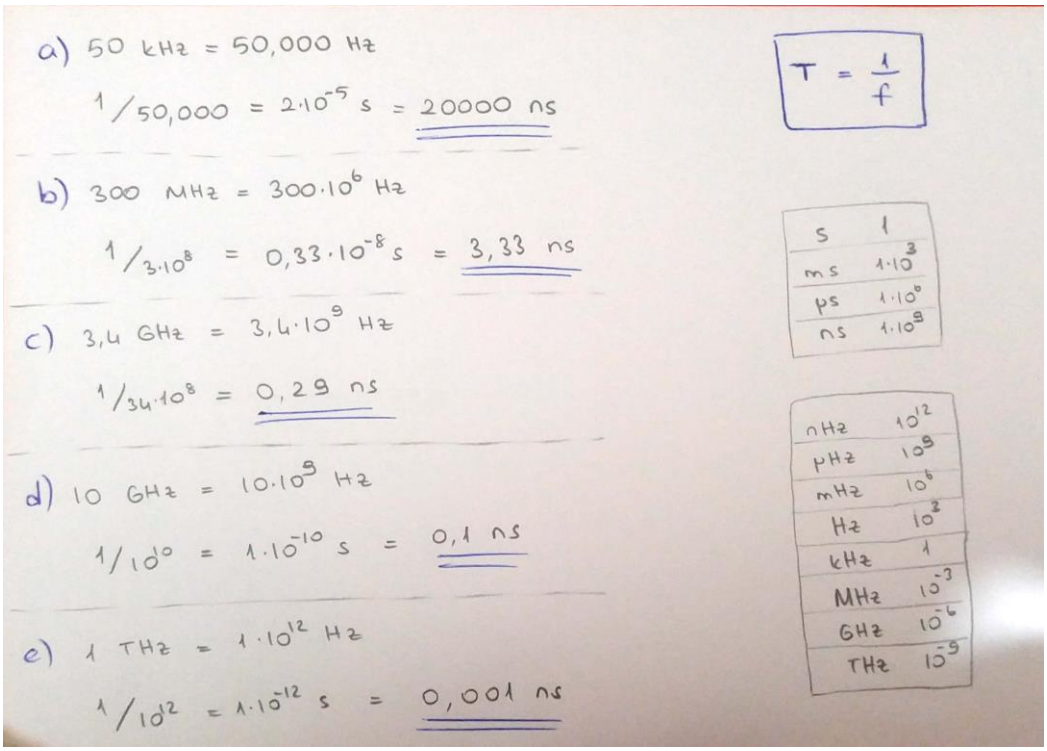
ESRA ERYILMAZ
171044046

CSE 232 SPRING 2020

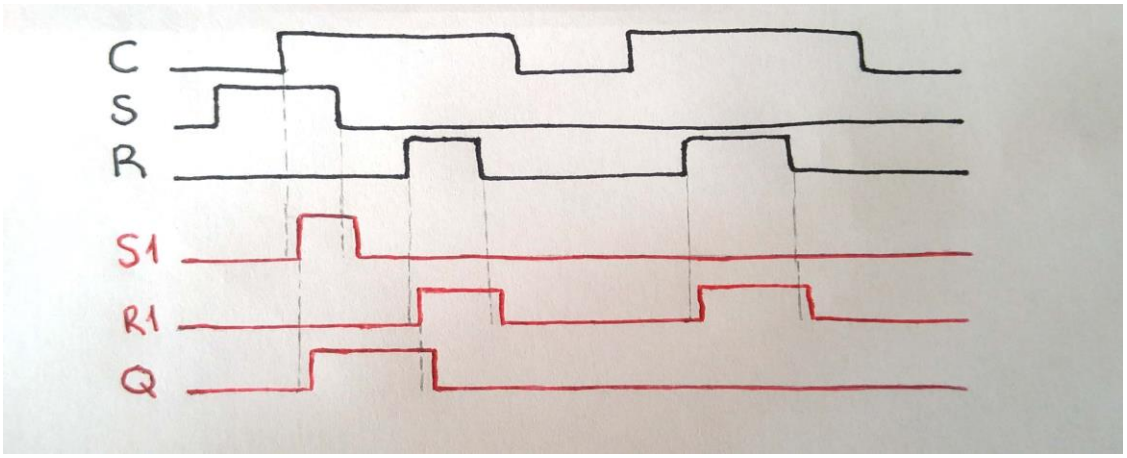
HOMEWORK 3

Due Date May 8, Friday

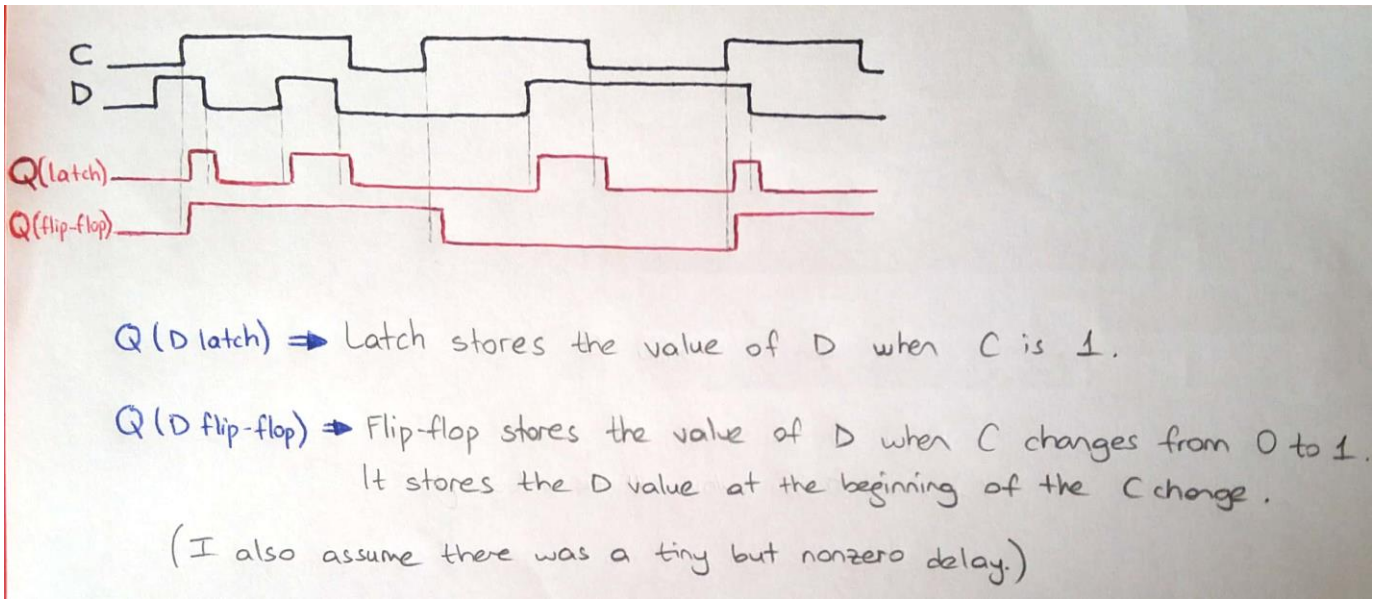
1. Compute the clock period for the following clock frequencies.
a. 50 kHz (early computers)
b. 300 MHz (Sony Playstation 2 processor)
c. 3.4 GHz (Intel Pentium 4 processor)
d. 10 GHz (PCs of the early 2010s)
e. 1 THz (1 terahertz) (PCs of the future?)



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



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.



Q (D latch) ⇒ Latch stores the value of D when C is 1.

Q (D flip-flop) ⇒ Flip-flop stores the value of D when C changes from 0 to 1. It stores the D value at the beginning of the C change.

(I also assume there was a tiny but nonzero delay.)

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

a. 4 b. 8 c. 9 d. 23 e. 900

1 bit can have two states: 0 or 1.

So;

Number of bits $\geq \boxed{\log_2 n}$

a) $\log_2 4 =$ 2 bits.

b) $\log_2 8 =$ 3 bits.

c) $\log_2 9 = 3{,}17$ rounded up is 4 bits.

d) $\log_2 23 = 4{,}52$ rounded up is 5 bits.

e) $\log_2 900 = 9{,}81$ rounded up is 10 bits.

5. If an FSM has N states, what is the maximum 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.

Assume that : if there are — Possible transitions

Two states A and B → A→B, B→A — 2 · 1

Three states A, B and C → A→B, A→C, B→A, B→C, C→A, C→B — 3 · 2

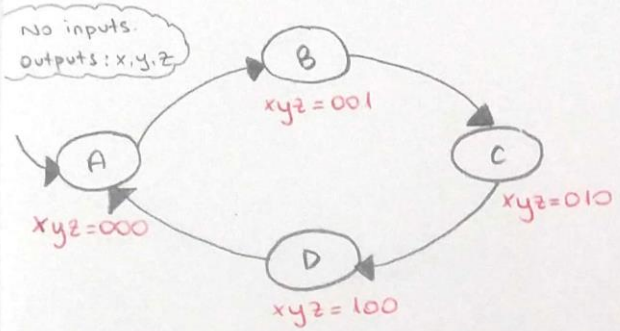
Four states A, B, C and D → A→B, A→C, A→D, B→A, B→C, B→D, C→A, C→B, C→D, D→A, D→B, D→C — 4 · 3

⇒ For N states, the number of possible transitions can generalize ; $N \cdot (N-1)$

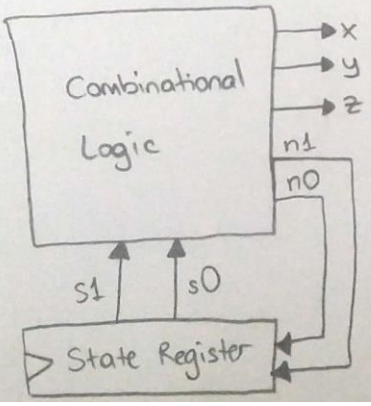
6. Draw a state diagram for an FSM with no inputs and three outputs x, y, and z. xyz should always exhibit the following sequence: 000, 001, 010, 100, repeat. The output should change only on a rising clock edge. Make 000 the initial state. Using the process for designing a controller, convert the FSM to a controller, stopping once you have created the truth table and derive the Boolean expressions.

Controller Design Process

Step 1 : Capture the FSM



Step 2A : Set up architecture



Step 2B : Encode the states

A = 00 C = 10

B = 01 D = 11

Step 2C : Fill in the truth table

inputs		outputs				
s1	s0	n1	n0	x	y	z
0	0	0	1	0	0	0
0	1	1	0	0	0	1
1	0	1	1	0	1	0
1	1	0	0	1	0	0

Step 2D : Implement combinational logic

$n1 = s1 \text{ XOR } s0$

$n0 = s0'$

$x = s1\,s0$

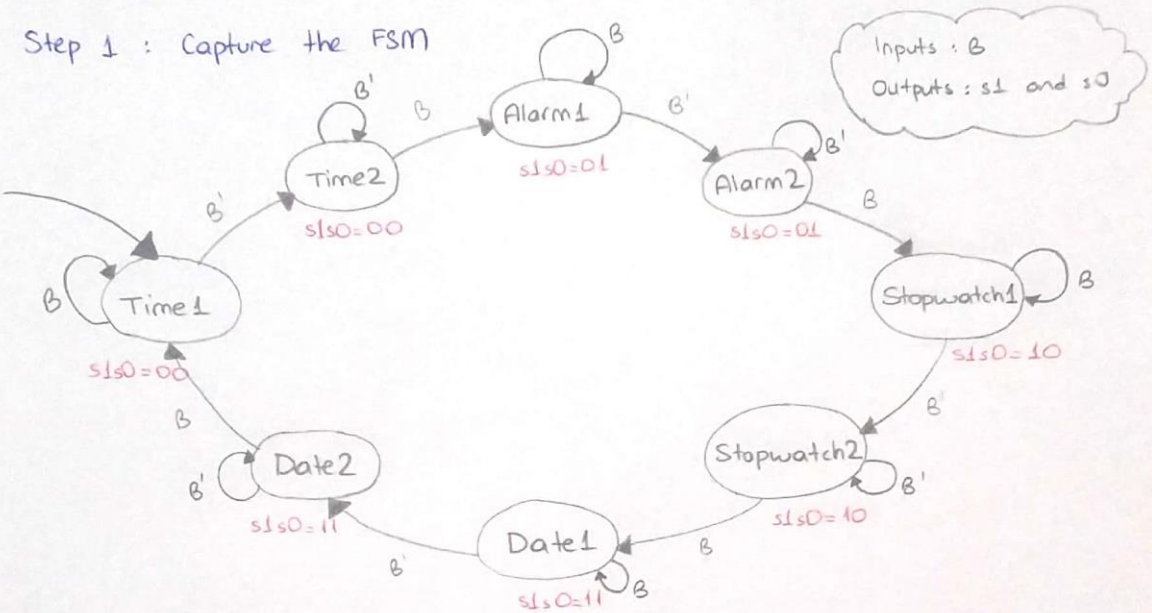
$y = s1\,s0'$

$z = s1'\,s0$

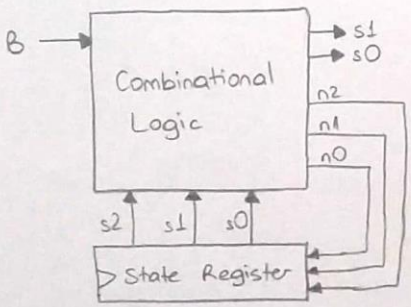
7. A wristwatch display can show one of four items: the time, the alarm, the stopwatch, or the date, controlled by two signals s1 and s0 (00 displays the time, 01 the alarm, 10 the stopwatch, and 11 the date—assume s1s0 control an N-bit mux that passes through the appropriate register). Pressing a button 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. Using the process for designing a controller, convert the FSM to a controller, stopping once you have created the truth table and derive the Boolean expressions.

Controller Design Process

Step 1 : Capture the FSM



Step 2A : Set up architecture



Step 2B : Encode the states

Time1 = 000
Time2 = 001
Alarm1 = 010
Alarm2 = 011
Stopwatch1 = 100
Stopwatch2 = 101
Date1 = 110
Date2 = 111

Step 2C : Fill in the truth table

	Inputs				Outputs				
	s2	s1	s0	B	n2	n1	n0	s1	s0
Time1	0	0	0	0	0	0	1	0	0
	0	0	0	1	0	0	0	0	0
Time2	0	0	1	0	0	0	1	0	0
	0	0	1	1	0	1	0	0	0
Alarm1	0	1	0	0	0	1	1	0	1
	0	1	0	1	0	1	0	0	1
Alarm2	0	1	1	0	0	1	1	0	1
	0	1	1	1	1	0	0	0	1
Stopwatch1	1	0	0	0	1	0	1	1	0
	1	0	0	1	1	0	0	1	0
Stopwatch2	1	0	1	0	1	0	1	1	0
	1	0	1	1	1	1	0	1	0
Date1	1	1	0	0	1	1	1	1	1
	1	1	0	1	1	1	0	1	1
Date2	1	1	1	0	1	1	1	1	1
	1	1	1	1	0	0	0	1	1

Step 2D : Implement combinational logic

$n2 = s2's1s0B + s2s1's0'B' + s2s1's0'B + s2s1's0B' + s2s1's0B + s2s1s0'B' + s2s1s0'B + s2s1s0B'$

$n2 = s2's1s0B + s2s1's0' + s2s1's0 + s2s1s0' + s2s1s0B'$

$n2 = s2's1s0B + s2s1' + s2s1s0' + s2s1s0B'$

$n1 = s2's1's0B + s2's1s0'B' + s2's1s0'B + s2's1s0B' + s2s1's0B + s2s1s0'B' + s2s1s0'B + s2s1s0B'$

$n1 = s1's0B + s2's1s0' + s1s0B' + s2s1s0'$

$n1 = s1's0B + s1s0B' + s1s0'$

$n0 = B'$

$s1 = s2$

$s0 = s1$