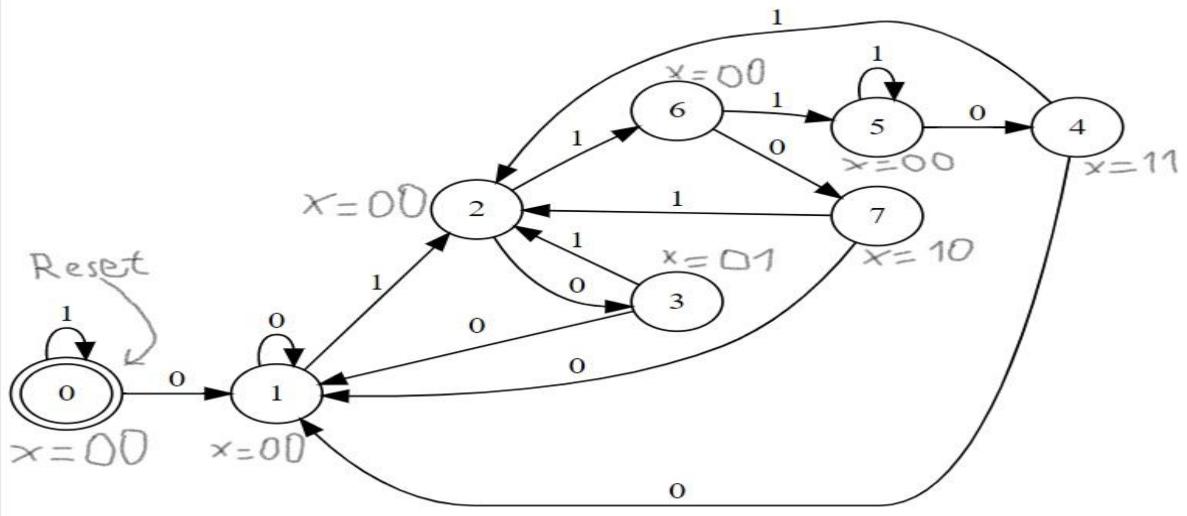
Student Name: Alp Tuna
Student ID: 2019400288

CMPE 240 2021 Experiment 3 Preliminary Work

(For illustrations you can use any drawing tool that you want including Microsoft Word Shapes. Do not use scanned images of hand drawn state machines and architecture diagrams.)
(For tables please use insert table feature of Microsoft Word)

Step 1: Capture the FSM: Create and draw the finite state machine that describes the desired behavior of the controller.

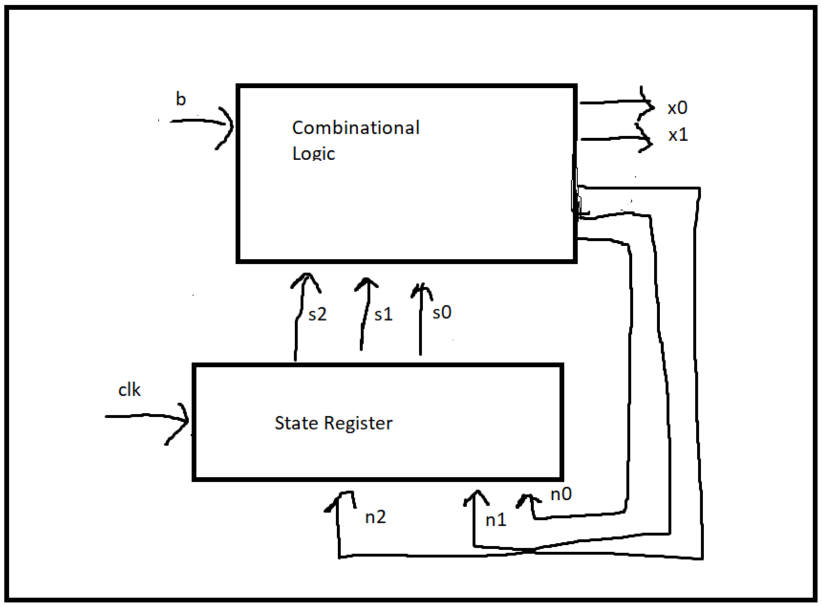
I really couldn't find proper names for the states. They were either getting too long or becoming irrelevant. Thus, I just enumerated them.



Student Name: Alp Tuna
Student ID: 2019400288

Step 2: Create the architecture: Create and draw standard architecture by a using state register of the appropriate width and combinational logic. Refer to book or lecture slides. Use the same convention.

There should be an arrow symbol at the left side of the state register to indicate that it includes rising-edge flip-flops. I forgot to draw it so I state it here.



Student Name: Alp Tuna
Student ID: 2019400288

Step 3: Encode the states: Assign a unique binary number to each state. Each binary number representing a state is known as an encoding. Any encoding will do as long as each state has a unique encoding. (The content of the following table is an example. Rename the states according to the ones you specified in the first section. Also use any encoding you want.)

STATE NAME	ENCODING
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Student Name: Alp Tuna
Student ID: 2019400288

Step 4: Create the state table: Create a truth table for the combinational logic such that the logic will generate the correct FSM outputs and next state signals. Ordering the inputs with state bits first makes this truth table describe the state behavior, so the table is a state table. (Update the table according to the number of state variables that you have used.)

CURRENT STATE	**INPUTS(X)**	**NEXT STATE**	**OUTPUTS(Y1Y0)**
0000	0	0000	00
…	…	…	…
…	…	…	…
…	…	…	…

MY TABLE

CURRENT STATE	***INPUTS(b)***	***NEXT STATE***	***OUTPUTS(x0,x1)***
000	***0***	***001***	***00***
000	***1***	***000***	***00***
001	***0***	***001***	***00***
001	***1***	***010***	***00***
010	***0***	***011***	***00***
010	***1***	***110***	***00***
011	***0***	***001***	***01***
011	***1***	***010***	***01***
100	***0***	***001***	***11***
100	***1***	***010***	***11***
101	***0***	***100***	***00***
101	***1***	***101***	***00***
110	***0***	***111***	***00***
110	***1***	***101***	***00***
111	***0***	***001***	***10***
111	***1***	***010***	***10***

Student Name: Alp Tuna
Student ID: 2019400288

Step 5: Draw the combinational logic: Implement the combinatorial logic using any method (You do not need to draw the inside circuit of multiplexers or decoders if you are using any. You can show those as blocks).

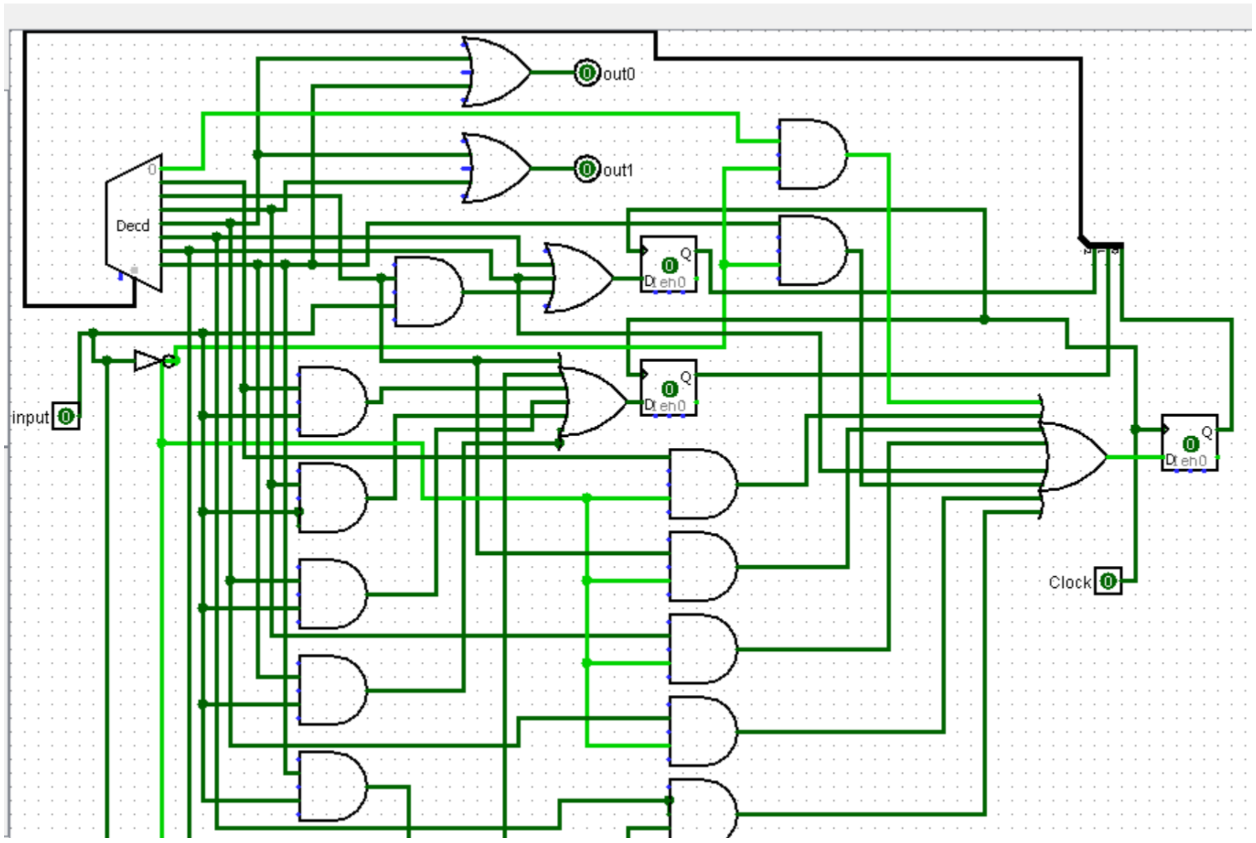
Note1: I couldn’t fit the circuit into one screen. So i divided it into two. Realize that there is an intersecting part in the pictures to avoid any ambiguity(To make it easier to understand the below picture.

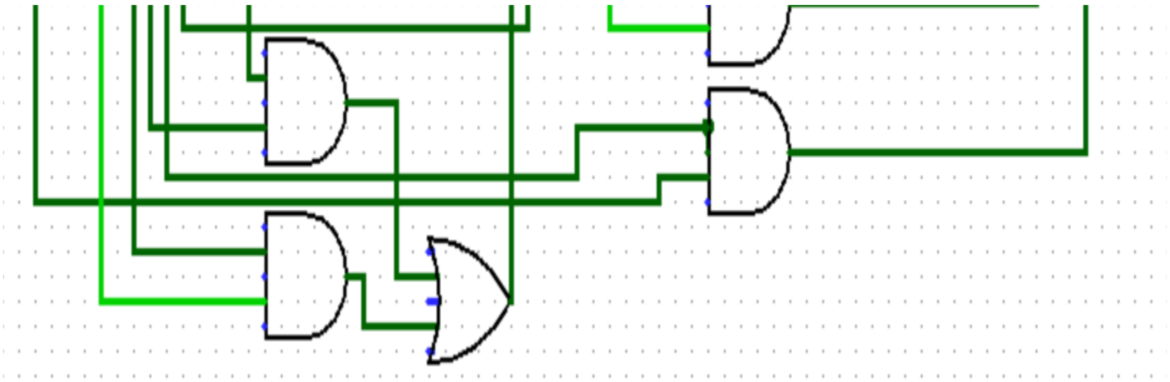
Note2: I realized that we were only supposed to implement the combinational part later. I actually implemented both combinational part and register part, I hope it is not a problem. You can see 3 D-Flip-flops in the image. They represent the register part and the inputs s0,s1,s2 are given through decoder.
You can also simulate the circuit by changing input and clock signals.

Note3: Since I ran out of space, I couldn’t show the reset input. I should’ve added this one as well.

.

Student Name: Alp Tuna
Student ID: 2019400288





Student Name: Alp Tuna
Student ID: 2019400288