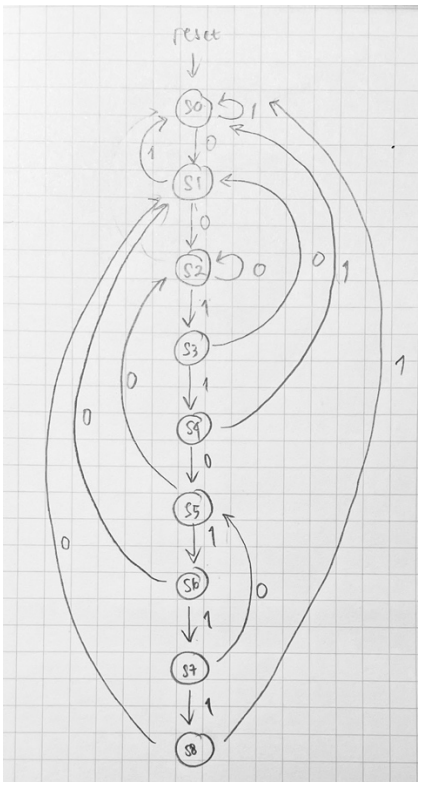
Mary Madeline Nicole 109006205

Logic Design Lab 2 ALU and Carry-Lookahead Adder - Report

1. *Design of FSM*

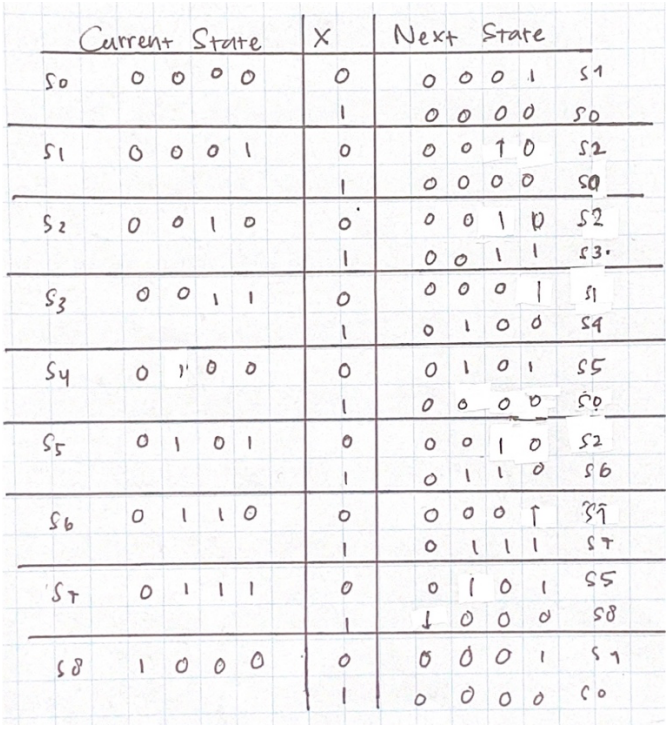
FSM or Finite State Machine consists of combinational, sequential, and output logic. Combinational logic is used to decide the next state, sequential logic is used to store current state and the output is a mixture of both combinational and sequential.

I created a State Diagram by searching for the pattern that fits according to the pattern given. When s0 data input is 1, it the arrow goes to itself, but when data is 0, it goes to the next state. Then it continues, when s1 data input is 0, it goes to the next state, s2, but when data is 1, it goes back to s0. When s2 data input is 0, it goes to itself, but when data input is 1, it goes to the next state s3. When s3 data input is 0, it goes back to s1 because there is one 0 to salvage. But when data is 1, it goes to the next stage s4. When s4 data input is 1, the state goes back to s0 because there is no pattern to salvage, but when it is 0, it goes to next state s5. When s5 data input is 1, it goes to the next state, but when data input is 0, next state goes back to s2. When s5 data input is 1, it goes to s6 but when its 0, it goes back to s1. When s6 next data is 1, it goes to s7, but if it is 0, it goes back to s1. Now comes the tricky part, when s7 data input is 0, it could be another (101) at the pattern, so the. State goes to s5. But if data input is 1, it will go to s8. And in s8, the pattern is complete. If s8 next data input is 1, state will go reset to s0, but if it is 1, it will go to s1. This is my state diagram:



I designed FSM according to pattern and my state diagram by creating a truth table, which consists of current state, the data, and next state. As the pattern we want is {001(101)$^{+}$11} so I made states from 0 to 8, and according to the pattern I assume that s0 is nothing, s1 is ‘0’, s2 is ‘0’, s3 is ‘1’, s4 is ’1’, s5 is ’0’, s6 is ‘1’, s7 is ‘1’, s8 is ‘1’. The conditions when s0, and the data is 0, means that it clops with the first pattern, which is 0, so the next state would be s1, but if the data is 1, next state will still be s0. I derived a truth table based on the pattern condition accordingly for every state.

Here is the truth table I created to find out the next state:



Current State	X	Next State
S0 0 0 0 0	0	0 0 0 1 S1
	1	0 0 0 0 S0
S1 0 0 0 1	0	0 0 1 0 S2
	1	0 0 0 0 S0
S2 0 0 1 0	0	0 0 1 0 S2
	1	0 0 1 1 S3
S3 0 0 1 1	0	0 0 0 1 S1
	1	0 1 0 0 S4
S4 0 1 0 0	0	0 1 0 1 S5
	1	0 0 0 0 S0
S5 0 1 0 1	0	0 0 1 0 S2
	1	0 1 1 0 S6
S6 0 1 1 0	0	0 0 0 1 S1
	1	0 1 1 1 S7
S7 0 1 1 1	0	0 1 0 1 S5
	1	1 0 0 0 S8
S8 1 0 0 0	0	0 0 0 1 S1
	1	0 0 0 0 S0

2. *How I implement FSM in Verilog*

My first always block introduces flip-flop s into the circuit. If the reset = 1 it will reset, change the state from s8 back to s0. But when reset = 0, state transition from the current state to the next state will occur.

My second always block describes the combinational circuit for computing the next state. According to the truth table I created.

The output is the flag, When the current state is equal to s8, flag will change into 1, else flag stays 0.

3. *Problem faced & how to deal*

- Finding out the next state needed some extra work, because I had to think of the conditions of the states, whether it matches the current pattern. And if not, what would be the next state, would it return to 0, or change to another state.

4. *Questions for TA*

-