Report of CS220 Assignment 2

Anushka Panda Harshit Raj Shubhan R

February 6, 2022

1 About

- About Assignment

  Name: Assignment 3
  Due on: Feb 17, 2022

- Team

  Name: Anushka Panda
  Email: apanda20@iitk.ac.in
  Roll: 200174

  Name: Harshit Raj
  Email: harshitr20@iitk.ac.in
  Roll: 200433

  Name: Shubhan R
  Email: shubhanr20@iitk.ac.in
  Roll: 200971

- About Course

  Course Code: CS220
  Name of course: Computer Organization
  Instructor: Dr. Urbi Chatterjee
  Semester: 2021-22-II

2 Sequence Detector using FSM

2.1 Question

Construct a finite state machine for a sequence detector that detects the sequence 1010. The FSM should permit overlapping too. For example if the input sequence is 01101010, the corresponding output sequence is 00000101.

1. Construct the state diagram, Transition and output table, excitation table.
2. Then build the K-map for the same and design the circuitry or logic diagram.
3. Write the Verilog code to implement the sequence detector.
4. Add a test bench to test the sequence detector given fifteen different inputs. Put five-time unit delay between consecutive inputs.

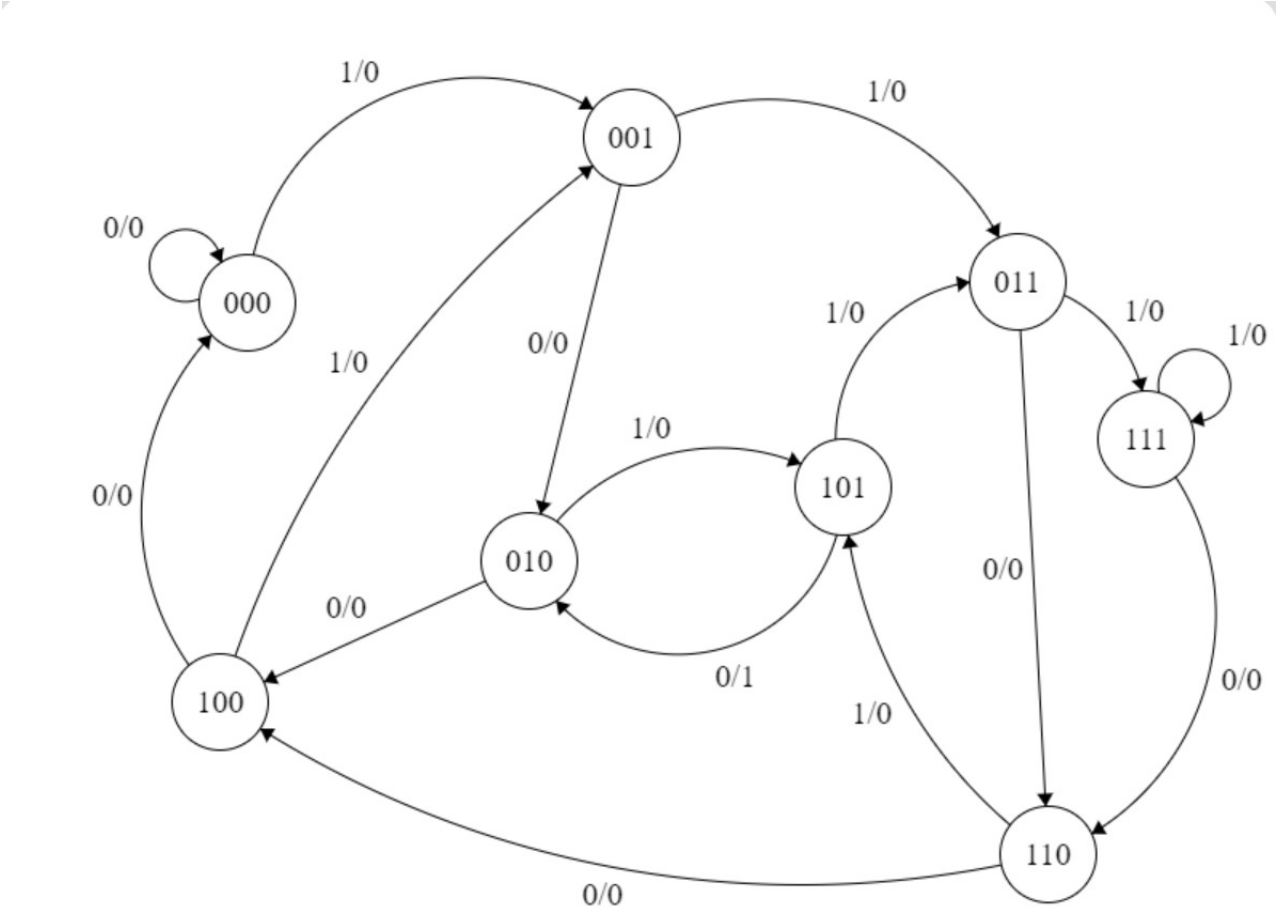
2.2 Logic

- First we take a 8 bit input from a port.
- Then we put the bits one by one in our "machine".
- The aforementioned machine has a state which is stores the value of the last three number inserted.
- Then the machine checks the input and the three stored states to match the sequence and gives an output.
- Finally the state is updated for next input.

2.3 Excitation Table

Input	Present State			Future State			Output
	q[2]	q[1]	q[0]	q[2]	q[1]	q[0]	
0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0
0	0	1	0	1	0	0	0
0	0	1	1	1	1	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	1	0	1
0	1	1	0	1	0	0	0
0	1	1	1	1	1	0	0
1	0	0	0	0	0	1	0
1	0	0	1	0	1	1	0
1	0	1	0	1	0	1	0
1	0	1	1	1	1	1	0
1	1	0	0	0	0	1	0
1	1	0	1	0	1	1	0
1	1	1	0	1	0	1	0
1	1	1	1	1	1	1	0

2.4 Sequence Diagram



2.5 Transition and Output

Present	Next State		Output op	
	Input		Input	
	0	1	0	1
000	000	001	0	0
001	010	011	0	0
010	100	101	0	0
011	110	111	0	0
100	000	001	0	0
101	010	011	1	0
110	100	101	0	0
111	110	111	0	0

2.6 Karnaugh Map for Output

x is don't care.

2.6.1 K-map for q[2]

Current State	q[1]	q[0]/inp	Next State
x	0	x/x	0
x	1	x/x	1

2.6.2 K-map for q[1]

Current State	q[0]	q[2]/inp	Next State
x	0	x/x	0
x	1	x/x	1

2.6.3 K-map for q[0]

Current State	inp	q[2]/q[1]	Next State
x	0	x/x	0
x	1	x/x	1

2.6.4 K-map for output

inp/q[0]	q[1]/q[2]			
	00	01	11	10
00	0	0	0	0
01	0	**1**	0	0
11	0	0	0	0
10	0	0	0	0

3 3-bit-odd-parity-generator

3.1 Question

Construct a finite state machine for the 3-bit odd parity bit generator. A serial parity-bit generator is a two-terminal circuit that receives coded messages and adds a parity bit to every m bits of the message so that the resulting outcome is an error detecting code. The inputs are assumed to arrive in strings of three symbols(m=3) and the string is spaced apart by single time units (i.e., the fourth place is blank). The parity bits are inserted in the appropriate spaces so that the resulting outcome is a continuous string of symbols without spaces. For even parity, a parity bit 1 is inserted, if and only if the numbers of 1s in the preceding string of three symbols are odd. For odd parity, a parity bit 1 is inserted, if and only if the numbers of 1s in the preceding string of three symbols is even. Here we will focusing on designing only odd parity generator.

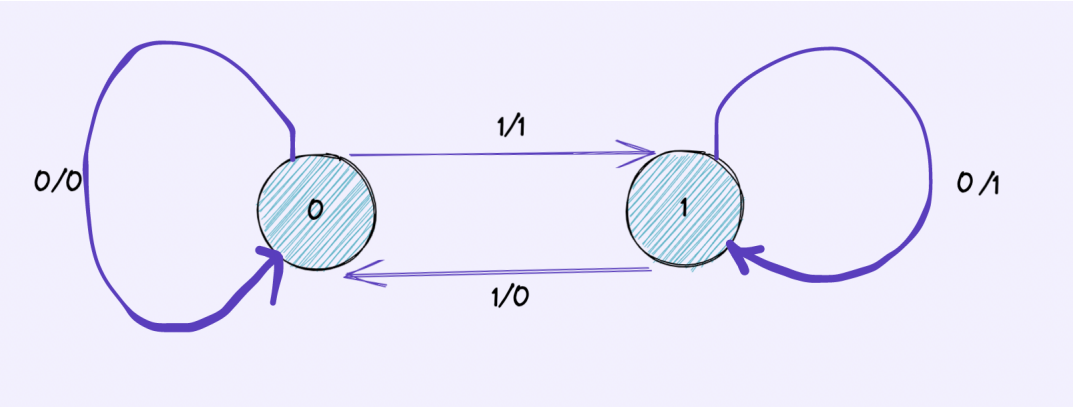
- Construct the state diagram, state table, transition and output table, excitation table.
- Then build the K-map for the same and design the circuitry or logic diagram.
- Then write the Verilog code module to implement the 3-bit odd parity bit generator.
- Now, add a test bench to test the 3-bit odd parity bit generator given 8 different inputs. Put five-time unit delay between consecutive inputs.

3.2 Basic explanation

We use a T-flip flop to carry out the parity generation as explained below.

- The bits of the input are passed into the T-flip flop, one by one.
- The default stored value of the flip-flop is 1. (So that we directly get odd parity bit)
- The output of the flip-flop, toggles every time a 1 is detected in the input. As a result of repeated toggling, even numbers of 1 will give 1 as output, and odd numbers of 1 will give 0 as output.
- The final output of the generator is the final output of the T-flip flop after reading all 3 bits of the input. As soon as the output is calculated, the flip flop is reset to read the new input.

3.3 State Diagram



3.4 Transition and Output Table

Current State	Input	Next State	Output
0	0	0	0
0	1	1	1
1	1	0	0
1	0	1	1

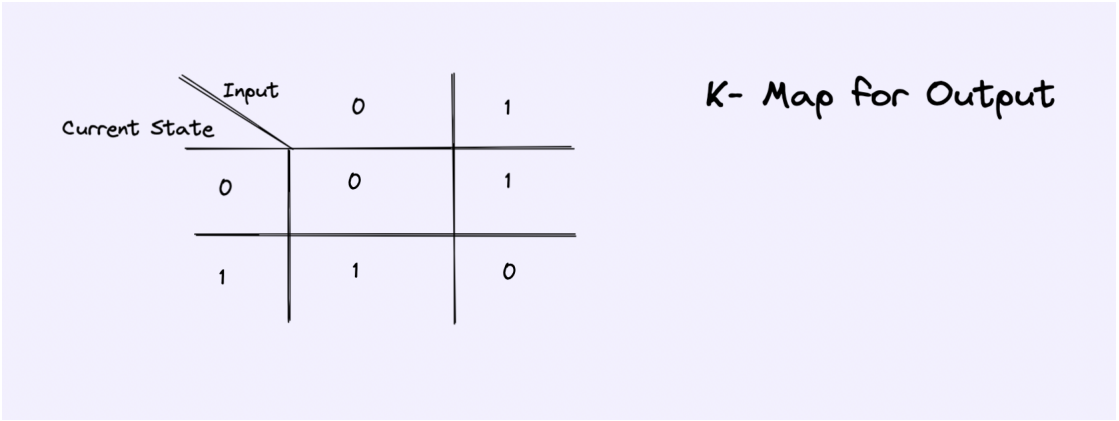
3.5 State Table

Current State	Input	(Next State, Output)
0	0	(0,0)
0	1	(1,1)
1	1	(0,0)
1	0	(1,1)

3.6 Excitation Table

Current State	Input	Next State	Output
0	0	0	0
0	1	1	1
1	1	0	0
1	0	1	1

3.7 Karnaugh Map for Output(Next State)



3.8 Circuit