CS220 : Computer Organization

Assignment - III

Team Members:

- Ujwal Jyot Panda (201060)
- Rishav Bikarwar (200792)
- Adi Pratap Singh (200036)

Question 1

For the solution to this problem, we have written a Verilog code for detecting the sequence “1010” in the given input. In other words, our code replaces “1010” with “0001” and the other four-bit values are replaced with “0000”.

The implementation consists of two modules “fsm_sub” and “fsm_driver”. The first one is the state machine and the other one is its driver code. At last bitwise or is taken of all the produced output in order to generate final output.

Excitation Table

Present State	Input	Next State	Output
000	1010	001	00010000
000	Not 1010	001	00000000
001	1010	010	00001000
001	Not 1010	010	00000000
010	1010	011	00000100
010	Not 1010	011	00000000
011	1010	100	00000010
011	Not 1010	100	00000000
100	1010	end	00000001
100	Not 1010	end	00000000

State Table

Present State	(Next State, Output)	
	Input=1010	Input=not 1010
000	(001,00010000)	(001,00000000)
001	(010,00001000)	(010,00000000)
010	(011,00000100)	(011,00000000)
011	(100,00000010)	(100,00000000)
100	(end,00000001)	(end,00000000)

K-Map

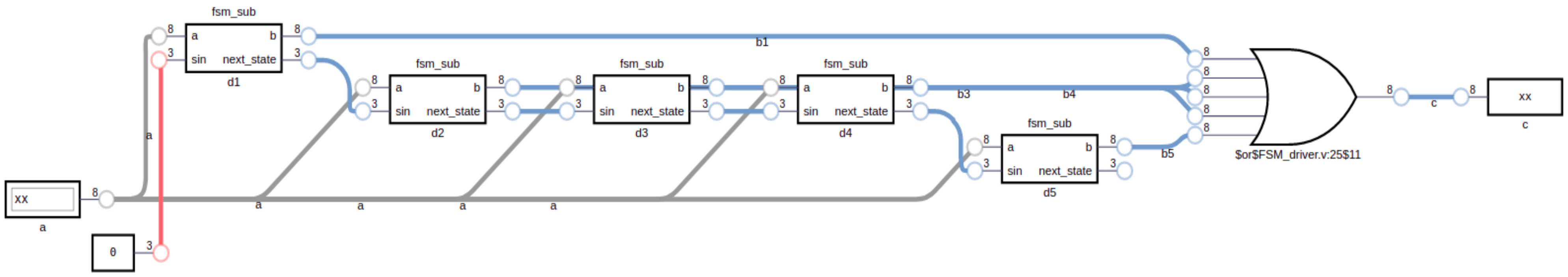
$\diagdown$ I / Q	1010	Not 1010
000	001	001
001	010	010
010	011	011
011	100	100
100	end	end

State Diagram

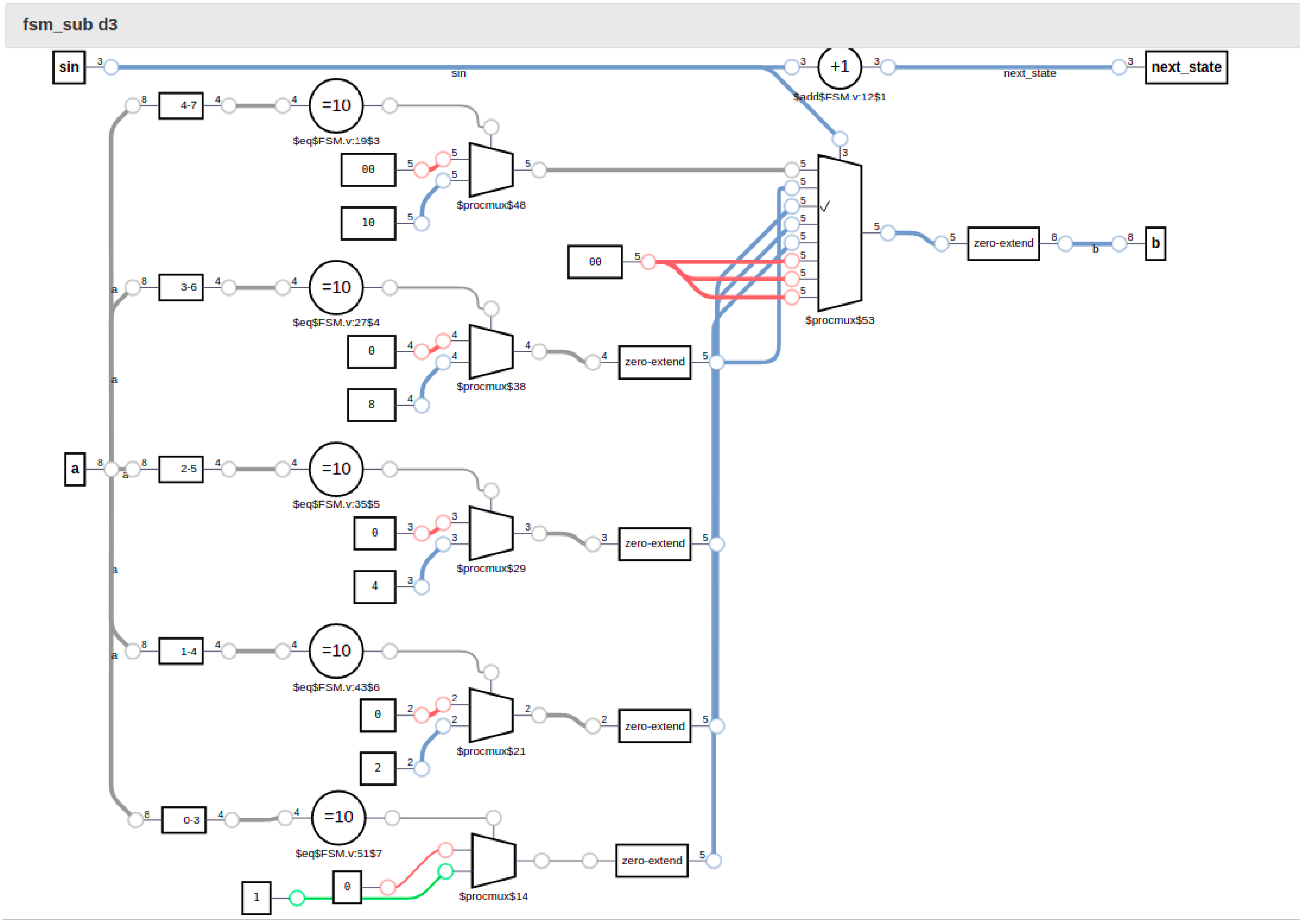
$$\begin{tikzcd}
000 \arrow[r] & 001 \arrow[r] & 010 \arrow[d] \\
\text{end} & 100 \arrow[l] & 011 \arrow[l]
\end{tikzcd}$$

Logic Digram

Following is the circuit of the verilog code we have written.



Here we have shown the “fsm_sub” as a module. On the next page we have shown its internal circuit.
This is the internal circuit of “fsm_sub” module

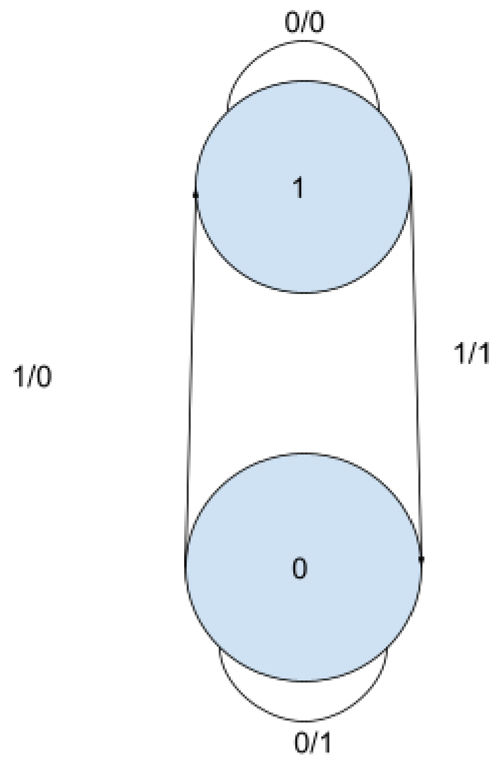


Question 2

For the solution to this problem, we have made an FSM that changes its states according to the number of ‘1’s it has received till now. Then, after the 3-bit input is passed, if the state is odd, then it gives output as 0, else it gives output as 1. This output is the parity bit. Hence, we have created a 3-bit odd parity generator.

The verilog code consists of 2 modules, namely “fsm”, which contains the FSM, and “fsm_driver” which is the driver code.

State Diagram



Excitation Table

Present State	Input	Next State	Output
1	1	0	1
1	0	1	0
0	1	1	0
0	0	0	1

State Table

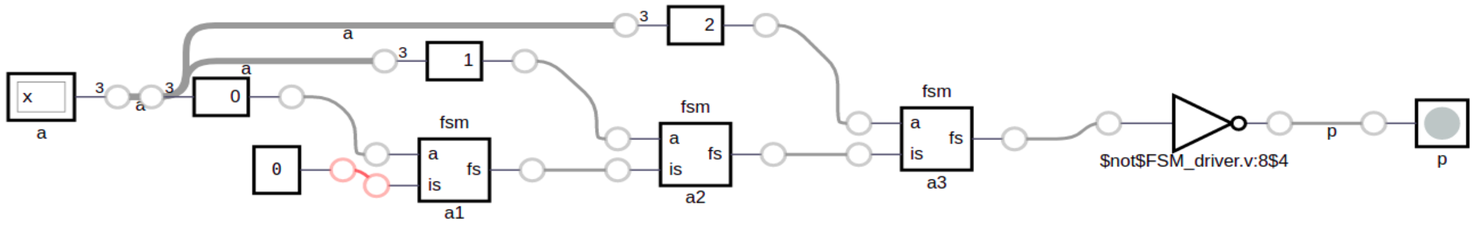
Present State	(Next State, Output)	
	Input=0	Input=1
1	(1,0)	(0,1)
0	(0,1)	(1,0)

K-Map

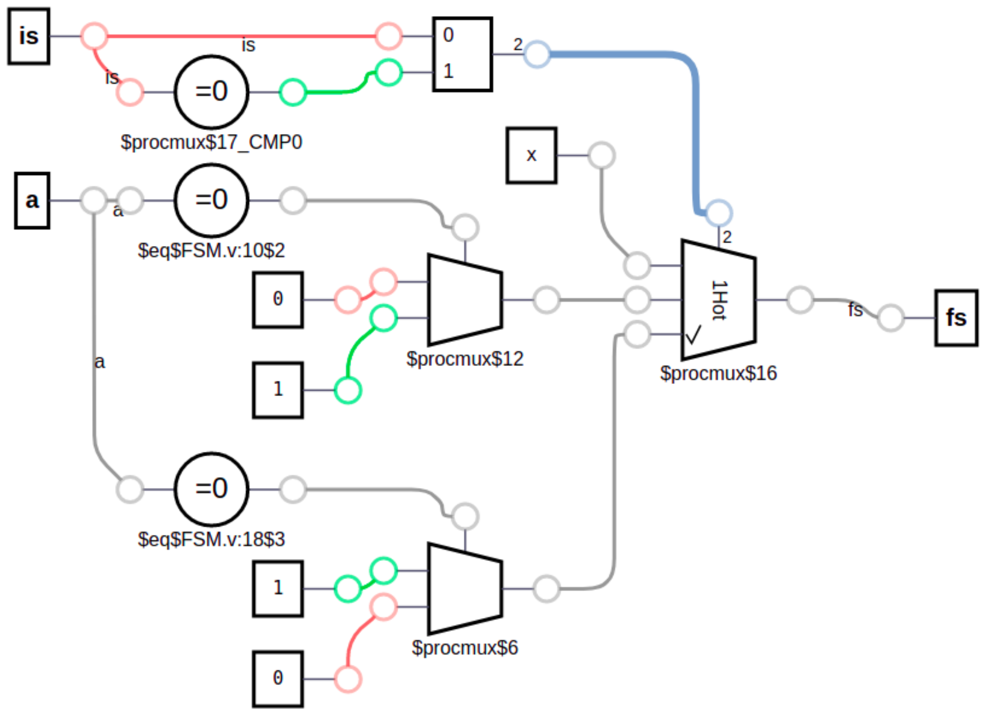
Q \ I	0	1
0	0	1
1	1	0

Logic Diagram

Following is the circuit of our implementation. We have used “fsm” as a module.



Below is the expanded internal circuit of “fsm” module which is imported in previous Circuit.


is
is
is
=0
$procmux$17_CMP0
0
1
2
a
a
=0
eqFSM.v:10$2
x
0
1
$procmux$12
2
1Hot
$procmux$16
fs
fs
a
=0
eqFSM.v:18$3
1
0
$procmux$6