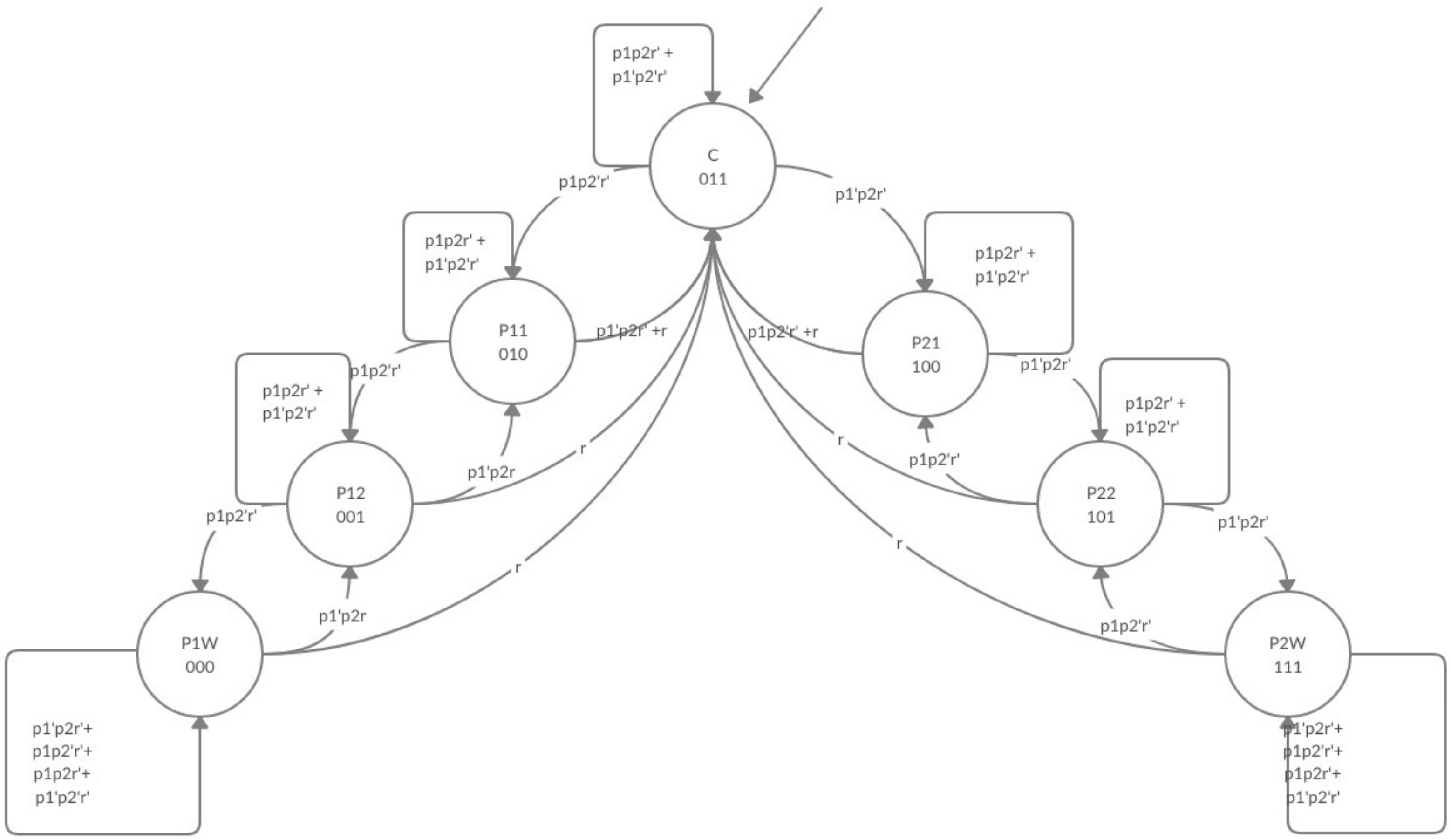
GIT Department of Computer Engineering
CSE 232 - Spring 2020
Project 1

BARAN HASAN BOZDUMAN
171044036

Finite State Machine Diagram



r values express for all given inputs for with r such as p1p2r + p1’p2’r + p1p2’r + p1’p2r for each state

For the given finite state machine I used button which are passed from button syncronizer circuit and firstly when program run initially normally the 000 led is blink which is used to show player 1’s win so I put a constant value put it in button syncronizer too and take its output and put in operations with register’s initial value which is 000 and when circuit begins it takes it to initial value for every starting of simualtion in the and I take s2s1s0 outputs to a decoder and connected the decoder outputs to leds.

Truth Table

The output of leds in order given finite machine code so for
n2n1n0~000 expreress first led of fsm and it increaases one by one just for the last led I used 111 so the 110 value has not use so the decoder output is empty for it.
For each state below I place them as groups for each state and in each group
1. row express=>p1 p2 r =>take to initial position
2. row express=>p1’ p2 r =>take to initial position
3. row express=>p1’ p2’ r =>take to initial position
4. row express=>p1 p2’ r =>take to initial position
5. row express=>p1’ p2’ r’ =>holds same position
6. row express=>p1 p2 r’ =>holds same position
7. row express=>p1 p2’ r’ =>moves p1 side which is left
8. row express=>p1’ p2 r’ =>moves p2 side which is right

INPUTS						OUTPUTS		
S2	S1	S0	P1	P2	R	N2	N1	N0
0	1	1	1	1	1	0	1	1
0	1	1	0	1	1	0	1	1
0	1	1	0	0	1	0	1	1
0	1	1	1	0	1	0	1	1
0	1	1	0	0	0	0	1	1
0	1	1	1	1	0	0	1	1
0	1	1	1	0	0	0	1	0
0	1	1	0	1	0	1	0	0
0	1	0	1	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	0	0	0	1	0	1	1
0	1	0	1	0	1	0	1	1
0	1	0	0	0	0	0	1	0
0	1	0	1	1	0	0	1	0
0	1	0	1	0	0	0	0	1
0	1	0	0	1	0	0	1	1
0	0	1	1	1	1	0	1	1
0	0	1	0	1	1	0	1	1
0	0	1	0	0	1	0	1	1
0	0	1	1	0	1	0	1	1

0	0	1	0	0	0	0	0	1
0	0	1	1	1	0	0	0	1
0	0	1	1	0	0	0	0	0
0	0	1	0	1	0	0	1	0
0	0	0	1	1	1	0	1	1
0	0	0	0	1	1	0	1	1
0	0	0	0	0	1	0	1	1
0	0	0	1	0	1	0	1	1
0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0
1	0	0	1	1	1	0	1	1
1	0	0	0	1	1	0	1	1
1	0	0	0	0	1	0	1	1
1	0	0	1	0	1	0	1	1
1	0	0	0	0	0	1	0	0
1	0	0	1	1	0	1	0	0
1	0	0	1	0	0	0	1	1
1	0	0	0	1	0	1	0	1
1	0	1	1	1	1	0	1	1
1	0	1	0	1	1	0	1	1
1	0	1	0	0	1	0	1	1
1	0	1	1	0	1	0	1	1
1	0	1	0	0	0	1	0	1
1	0	1	1	1	0	1	0	1
1	0	1	1	0	0	1	0	0
1	0	1	0	1	0	1	1	1
1	1	1	1	1	1	0	1	1

1	1	1	0	1	1	0	1	1
1	1	1	0	0	1	0	1	1
1	1	1	1	0	1	0	1	1
1	1	1	0	0	0	1	1	1
1	1	1	1	1	0	1	1	1
1	1	1	1	0	0	1	1	1
1	1	1	0	1	0	1	1	1

Boolean Equivalents

n2

r’(s2’s1s0p1’p2 + s2s1’s0’(p1p2’)’ + s2s0)

n1’

r’(s2’s1(s0p1’p2 + s0’p1p2’) + s1’s0(p1p2+ p1’p2’ + p1p2’) + s2’s1’s0’ + s2s1’s0’(p1p2+p1’p2’+ p1p2’))

n0’

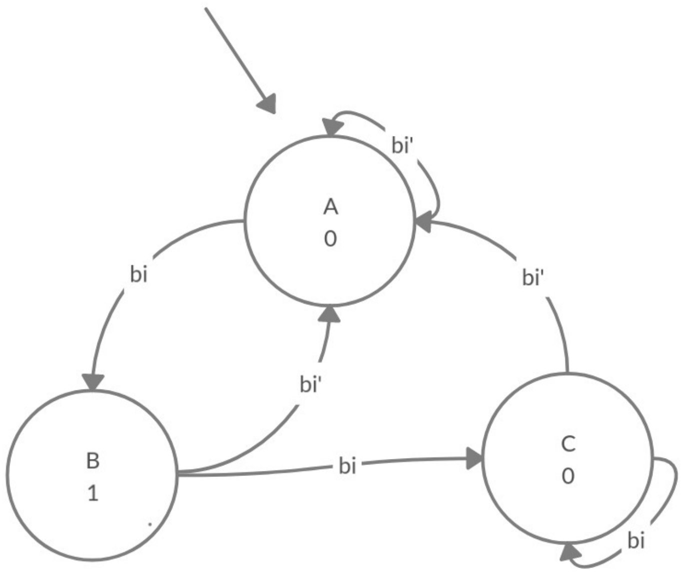
r’(s0(p1 xor p2)(s2’s1s0 + s2’s1’s0) + (p1 xor p2)’s0’(s2’s1+s2s1’)+s2’s1’s0’+s2s1’s0p1p2’)

Score

To keep score I use conter and connect it to winning states of decoder then I take output of conter and connect it to hex digit display so it shows the values hexadecimally

FSM for Button Syncronizer

Button syncronizer for the take output once and so we doesnt need extra states for our circuit
It is regardless time of press so for each it takes once press



Truth Table

INPUTS			OUTPUTS		
s1	s0	bi	n1	n0	b0
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	0	0

Boolean Equivalent

n1
s1’s0bi + s1s0’bi
n0
s1’s0’bi
b0
s1’s0
ps:

reference for the button syncronizer

Digital Design 2E : with RTL Design, VHDL and Verilog - Frank Vahid - John Wiley High