State Machine Controller

Instructions

Case 2: opcode = 110
op = 10

How to Execute each instruction

① Put Sximm8 into Rn

How:

- Set vsel to 0100
- Set write to 1
- set nsel to 100

} S1a

② Shift Rm based on the [2:0] Shift and put into RD

How:

- Set nsel to 001
- set load a to 0
- Set load b to 1

} S2a

- set bsel to 0
- Set asel to 1
- Set load c to 1
- set load s to 1

} S2b

- Set write to 1
- Set vsel to 0001

} S2c

③ Add shifted Rm with Rn and put in Rd

How:

- Set nsel to 001
- set load a to 0
- set load b to 1

} S3a

- Set nsel to 100
- Set load a to 1
- set load b to 0

} S3b

- set asel to 0
- set bsel to 0
- Set load c to 1
- set load s to 1

} S3c

- Set write to 1
- Set vsel to 0001
- Set nsel to 010

} S3d

Top level:

```
`define S1 110

① `define S1a 10
  //Execute R[Rn] = sx(im8)

② `define S1b 00
  // Execute R[RD] = sh_Rm

`define S2 101

③ `define S2a 00
  //Execute R[Rd] = R[Rn] + sh_Rm

④ `define S2b 01
  //Execute Status = f(R[Rn] - sh_Rm)

⑤ `define S2c 10
  //Execute R[Rd] = R[Rn] & sh_Rm

⑥ `define S2d 11
  //Execute R[Rd] = ~sh_Rm
```

⑤ AND shifted Rm and Rn and put in Rd

How:

- Set nsel to 001
- Set load a to 0
- set load b to 1

} S5a

- Set nsel to 100
- Set load a to 1
- set load b to 0

} S5b

- Set asel to 0
- Set bsel to 0
- Set load c to 1
- set load s to 1

} S5c

- Set nsel to 010
- Set vsel to 0001
- Set write to 1

} S5d

⑥ NOT shifted Rm and put in Rd

How:

- set nsel to 001
- set load a to 0
- set load b to 1

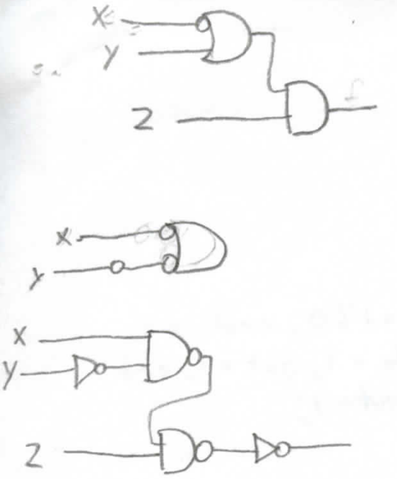
} S6a

- Set asel to 1
- set bsel to 0
- Set load c to 1
- set load s to 1

} S6b

- Set nsel to 010
- set vsel to 0001
- set write to 1

} S6C



n1	00	01	11	10
00	0 (0)	0 (1)	1 (3)	0 (2)
01	0 (4)	0 (5)	0 (7)	0 (6)
11	1 (12)	0 (13)	1 (15)	0 (14)
10	1 (8)	1 (9)	0 (11)	1 (10)

0011

1X00

1111

100X

10X0

f	00	01	11	10
00	0 (0)	0 (1)	0 (3)	0 (2)
01	0 (4)	0 (5)	0 (7)	0 (6)
11	0 (12)	0 (13)	0 (15)	0 (14)
10	1 (8)	1 (9)	1 (11)	1 (10)

10XX

state encoding

A = 00

B = 01

C = 11

D = 10

p2 , p1 = present state

n2, n1 = next_state

y, W = in

f = out

n2	00	01	11	10
00	(0)	(1)	1 (3)	(2)
01	(4)	(5)	1 (7)	1 (6)
11	(12)	1 (13)	1 (15)	(14)
10	(8)	(9)	1 (11)	(10)

XX11

011X

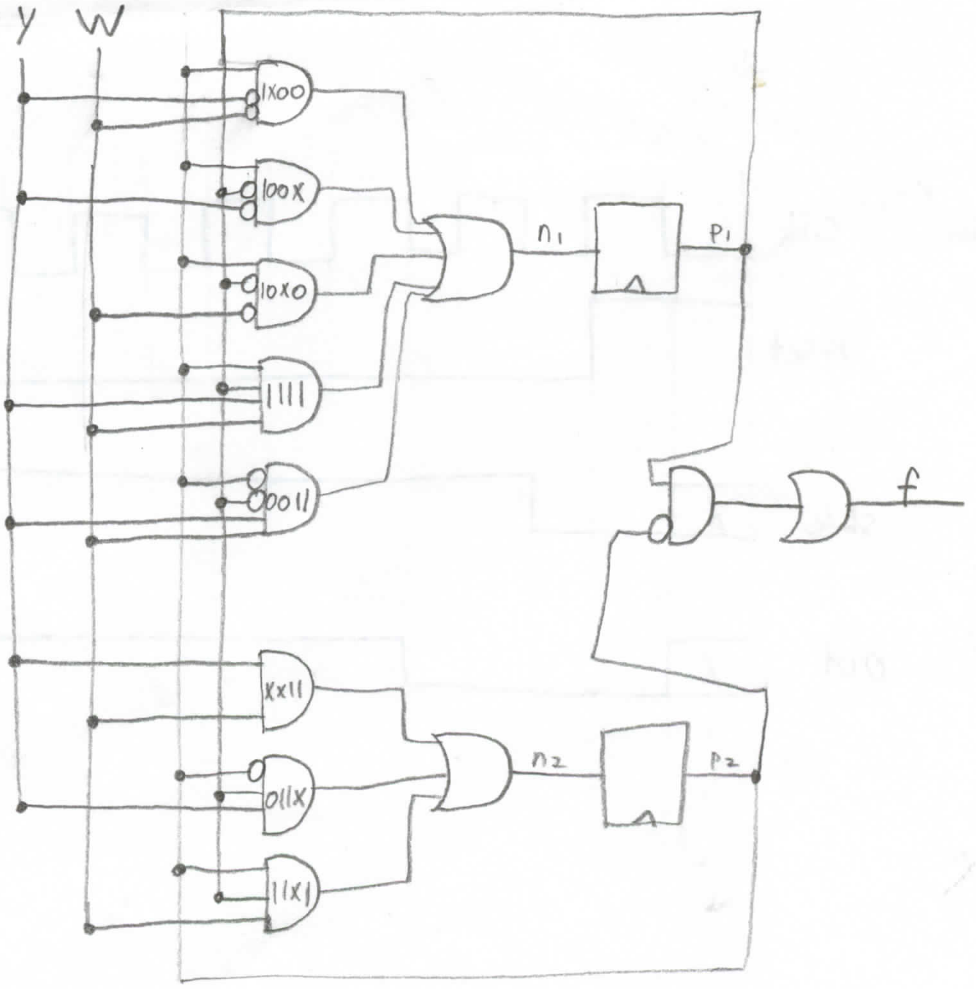
11X1

f	0	1
0	0	0
1	1	0

10

$f = p_2 \wedge \neg p_1$

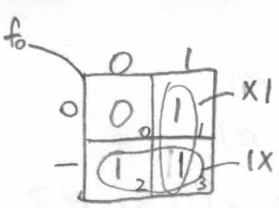
	p2	p1	y	w	n2	n1	f	
	0	0	0	0	0	0	0	(0)
A	0	0	0	1	0	0	0	(1)
	0	0	1	0	0	0	0	(2)
	0	0	1	1	1	1	0	(3)
	0	1	0	0	0	0	0	(4)
B	0	1	0	1	0	0	0	(5)
	0	1	1	0	1	0	0	(6)
	0	1	1	1	1	0	0	(7)
	1	0	0	0	0	1	1	(8)
D	1	0	0	1	0	1	1	(9)
	1	0	1	0	0	1	1	(10)
	1	0	1	1	1	0	1	(11)
	1	1	0	0	0	1	0	(12)
C	1	1	0	1	1	0	0	(13)
	1	1	1	0	0	0	0	(14)
	1	1	1	1	1	1	0	(15)



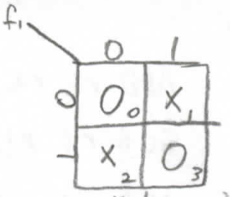
State encoding

A = 00
B = 01
C = 11
D = 10

P1	P0	f1	f0	
0	0	0	0	(0)
0	1	1	X	(1)
1	0	1	X	(2)
1	1	1	0	(3)



f0	0	1
0	0_0	1_1
–	1_2	1_3

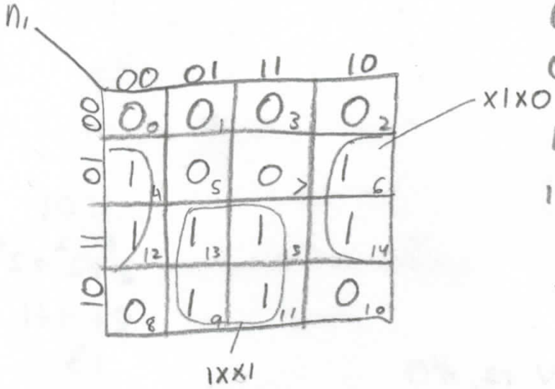


f1	0	1
0	0_0	X_1
–	X_2	0_3

"doesn't depent on present"

let :

p_1, p_0 = present state
n_1, n_0 = next state
X_2, X_1 = inputs
f_1, f_0 = output

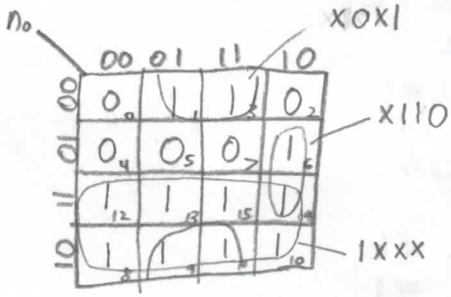


n1	00	01	11	10
00	0_0	0_1	0_3	0_2
01	1_4	0_5	0_7	1_6
11	1_{12}	1_{13}	1_{15}	1_{14}
10	0_8	1_9	1_{11}	0_{10}

P1 P0	f0
0 0	0
0 1	1
1 0	1
1 1	1

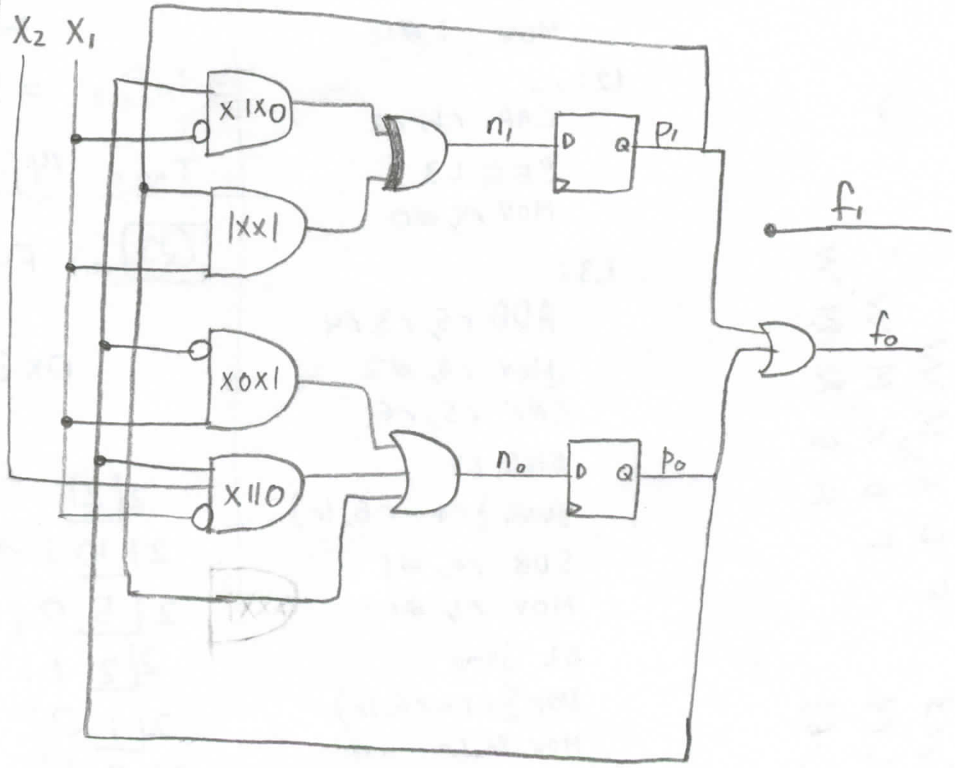
∴ this is an OR Gate

	P1	P0	X2	X1	n1	n0	
A	0	0	0	0	0	0	(0)
	0	0	0	1	0	1	(1)
	0	0	1	0	0	0	(2)
	0	0	1	1	0	1	(3)
B	0	1	0	0	1	0	(4)
	0	1	0	1	0	0	(5)
	0	1	1	0	1	1	(6)
	0	1	1	1	0	0	(7)
D	1	0	0	0	0	1	(8)
	1	0	0	1	1	1	(9)
	1	0	1	0	0	1	(10)
	1	0	1	1	1	1	(11)
C	1	1	0	0	1	1	(12)
	1	1	0	1	1	1	(13)
	1	1	1	0	1	1	(14)
	1	1	1	1	1	1	(15)



n0	00	01	11	10
00	0_0	1_1	1_3	0_2
01	0_4	0_5	0_7	1_6
11	1_{12}	1_{13}	1_{15}	1_{14}
10	1_8	1_9	1_{11}	1_{10}

0001
0011
1001
1011



X2
X1
X1X0
1XX1
X0X1
X110
n1
n0
D
Q
P1
P0
f1
f0