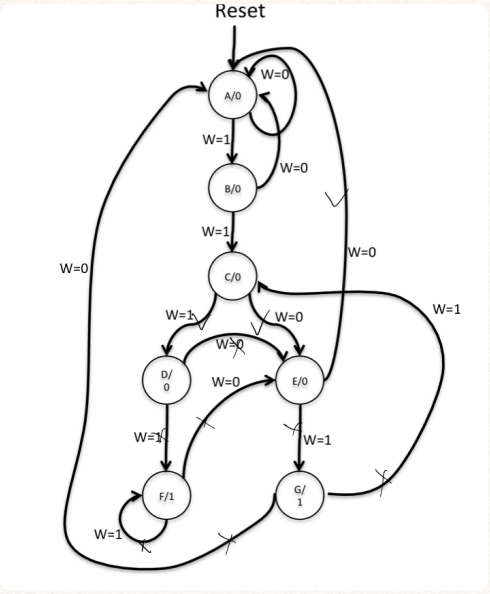
1.

- Given the starter circuit, is the `Reset` signal a synchronous or asynchronous reset?
- Is it active high, or active low signal?
- How should the `Reset` signal feature in the tests that you run on your FSM?

- Reset is asynchronous reset since it doesn't depend on clock signal.
- It is active high signal
- When need to use reset signal to clear out the status in FSM.



2.

State	Flip Flop Values		
A/0	0	0	0
B/0	0	0	1
C/0	0	1	0
D/0	0	1	1
E/0	1	0	0
F/1	1	0	1
G/1	1	1	0

	F_2	F_1	F_0	W	F_2	F_1	F_0	
A	0	0	0	0	0	0	0	A
	0	0	0	1	0	0	1	B
B	0	0	1	0	0	0	0	A
	0	0	1	1	0	1	0	C
C	0	1	0	0	1	0	0	E
	0	1	0	1	0	1	1	D
D	0	1	1	0	1	0	0	G
	0	1	1	1	1	0	1	F/1
E	1	0	0	0	0	0	0	A
	1	0	0	1	1	1	0	G/1
F	1	0	1	0	1	0	0	E
	1	0	1	1	1	0	1	F/1
G	1	1	0	0	0	0	0	A
	1	1	0	1	0	1	0	C

K-MAP: F_2

	$\bar{F}_0 \cdot \bar{W}$	$\bar{F}_0 \cdot W$	$F_0 \cdot W$	$F_0 \cdot \bar{W}$
$\bar{F}_2 \cdot \bar{F}_1$	0	0	0	0
$\bar{F}_2 \cdot F_1$	1	0	1	1
$F_2 \cdot F_1$	0	0	X	X
$F_2 \cdot \bar{F}_1$	0	1	1	1

$$F_2 = \bar{F}_2 F_1 \bar{W} + F_1 F_0 + F_2 F_0 + F_2 \bar{F}_1 W$$

K-MAP: F_1

	$\bar{F}_0 \cdot \bar{W}$	$\bar{F}_0 \cdot W$	$F_0 \cdot W$	$F_0 \cdot \bar{W}$
$\bar{F}_2 \cdot \bar{F}_1$	0	0	1	0
$\bar{F}_2 \cdot F_1$	0	1	0	0
$F_2 \cdot F_1$	0	1	X	X
$F_2 \cdot \bar{F}_1$	0	1	0	0

$$F_1 = F_1 \bar{F}_0 W + F_2 \bar{F}_0 W + \bar{F}_2 \bar{F}_1 F_0 W$$

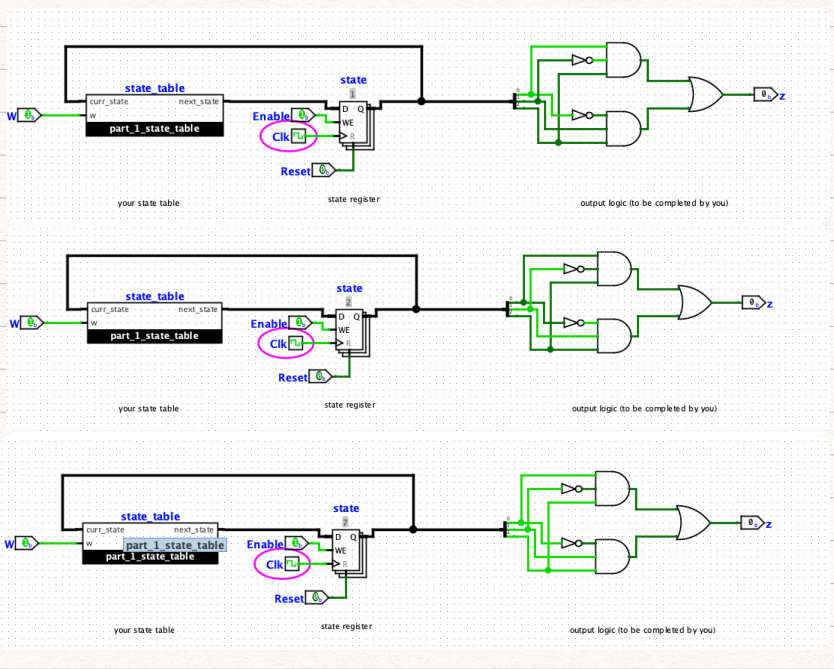
K-MAP: F_0

	$\bar{F}_0 \cdot \bar{W}$	$\bar{F}_0 \cdot W$	$F_0 \cdot W$	$F_0 \cdot \bar{W}$
$\bar{F}_2 \cdot \bar{F}_1$	0	1	0	0
$\bar{F}_2 \cdot F_1$	0	1	1	0
$F_2 \cdot F_1$	0	0	X	X
$F_2 \cdot \bar{F}_1$	0	0	1	0

$$F_0 = \bar{F}_2 \bar{F}_0 W + F_1 F_0 W + F_2 F_0 W$$

5.

1.



↓

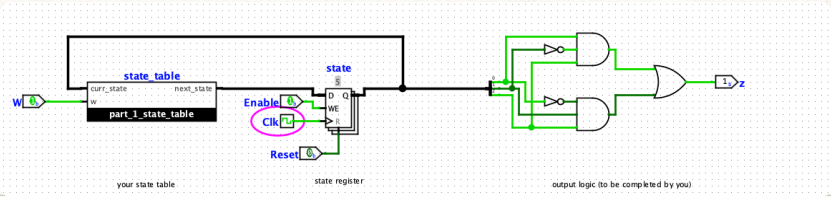
1

↓

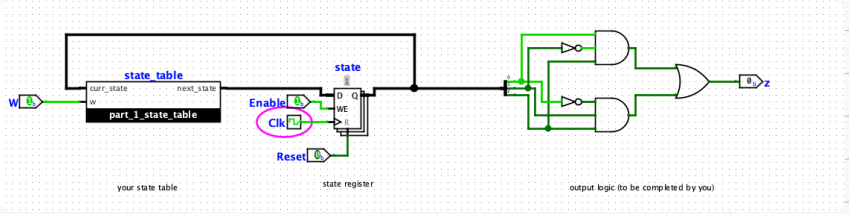
1

↓

1

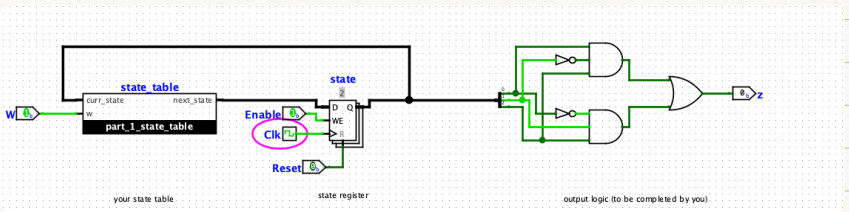

state_table
curr_state
next_state
w
part_1_state_table
W
Enable
Clk
Reset
state
D
Q
WE
R
z
your state table
state register
output logic (to be completed by you)

1


state_table
curr_state
next_state
w
part_1_state_table
W
Enable
Clk
Reset
state
D
Q
WE
R
z
your state table
state register
output logic (to be completed by you)

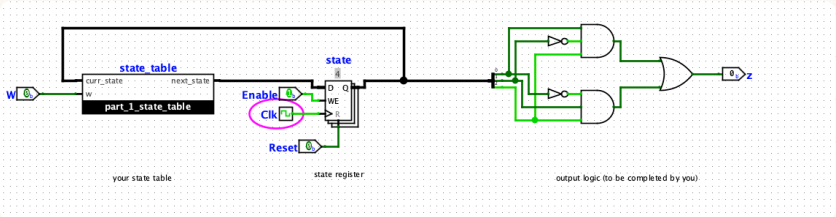
↓

1


state_table
curr_state
next_state
w
part_1_state_table
W
Enable
Clk
Reset
state
D
Q
WE
R
z
your state table
state register
output logic (to be completed by you)

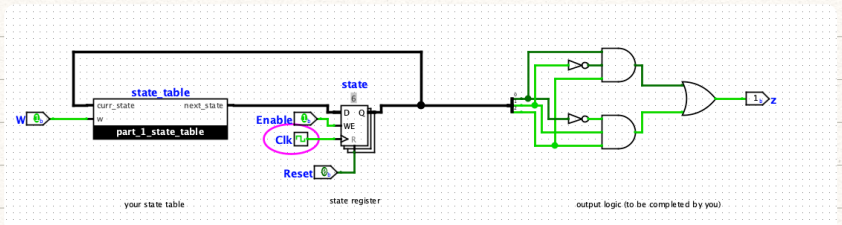
↓

0


state_table
curr_state
next_state
w
part_1_state_table
W
Enable
Clk
Reset
state
D
Q
WE
R
z
your state table
state register
output logic (to be completed by you)

↓

1


state_table
curr_state
next_state
w
part_1_state_table
W
Enable
Clk
Reset
state
D
Q
WE
R
z
your state table
state register
output logic (to be completed by you)

Part II

$Cx^2 + Bx + A$

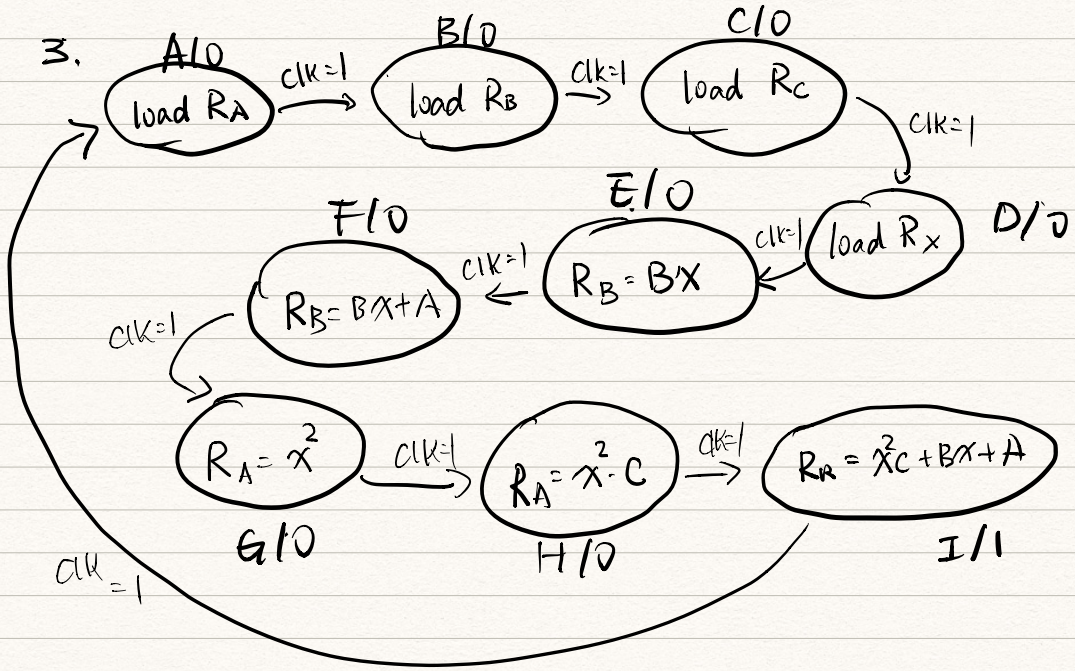
2、

- 0000 • Load data_in to R_A
  - ld_a = 1
- 0001 • Load data_in to R_B
  - ld_b = 1
- 0010 • Load data_in to R_C
  - ld_c = 1
- 0011 • Load data_in to R_X
  - ld_x = 1
- 0100 • Multiply R_B & R_X Store value in R_B
  - alu_op = 1
  - alu_select_a = 01 alu_select_b = 11
  - ld_alu_out = 1 ld_b = 1
- 0101 • Add R_A to R_B Store value in R_B
  - alu_select_a = 00 alu_select_b = 01
  - ld_alu_out = 1 ld_b = 1
- 0110 • Multiply Rx & Rx store value in R_A
  - alu_op = 1
  - alu_select_a = 11 alu_select_b = 11
  - ld_alu_out = 1 ld_a = 1
- 0111 • Multiply R_C & R_A store value in R_A
  - alu_op = 1
  - alu_select_a = 10 alu_select_b = 00
  - ld_alu_out = 1 ld_a = 1
- 1000 • Add R_a & R_B store value to R_R
  - alu_select_a = 00 alu_select_b = 01
  - ld_r = 1

$S_3S_2S_1S_0$	STATE	R_A	R_B	R_C	R_X	R_R	CLK	R_A	R_B	R_C	R_X	R_R
0000	A/0	0	0	0	0	0	1	A	0	0	0	0
0001	B/0	A	0	0	0	0	1	A	B	0	0	0
0010	C/0	A	B	0	0	0	1	A	B	C	0	0
0011	D/0	A	B	C	0	0	1	A	B	C	X	0
0100	E/0	A	B	C	X	0	1	A	BX	C	X	0
0101	F/0	A	BX	C	X	0	1	A	BX+A	C	X	0
0110	G/0	A	BX+A	C	X	0	1	X^2	BX+A	C	X	0
0111	H/0	X^2	BX+A	C	X	0	1	CX^2	BX+A	C	X	0
1000	I/1	CX^2	BX+A	C	X	0	1	CX^2	BX+A	C	X	CX^2+BX+A

	ld_a	ld_b	ld_c	ld_x	ld_alu_out	alu_select_a	alu_select_b	alu_op	ld_r
0000	1	0	0	0	0	00	00	0	0
0001	0	1	0	0	0	00	00	0	0
0010	0	0	1	0	0	00	00	0	0
0011	0	0	0	1	0	00	00	0	0
0100	0	1	0	0	1	01	11	1	0
0101	0	1	0	0	1	00	01	0	0
0110	1	0	0	0	1	11	11	1	0
0111	1	0	0	0	1	10	00	1	0
1000	0	0	0	0	0	00	01	0	1

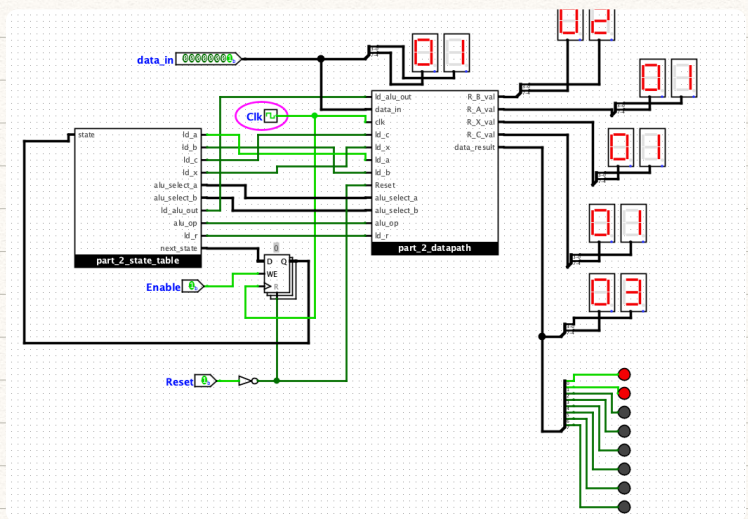
3.



5.

A	B	C	X	$Cx^2 + bx + A$
1	1	1	1	3
5	4	3	2	25

1111 → 3



5432 → 25