I completed this assignment entirely on my own.

Finite-state Machine → simple vending machine

1) Run examples

NNG → G
DG → G
NNNC → C
NDC → C
NNNG → N, G
DNNG → N, N, G
DC → D
NG → N

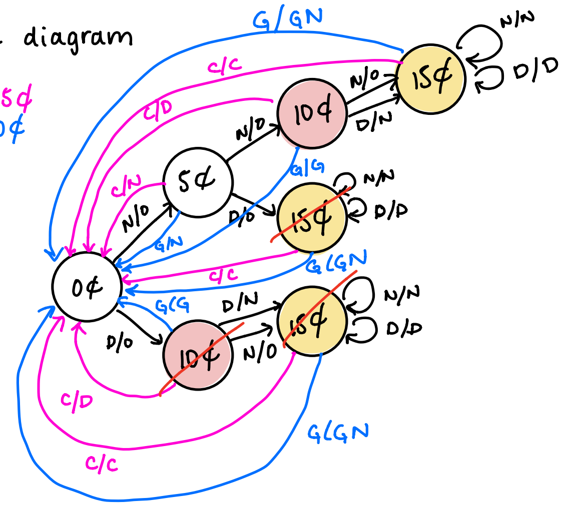
2) states: how much $ is in the vending machine (up to 15¢)

inputs: N = nickel
D = dime
G = request gum
C = request candy

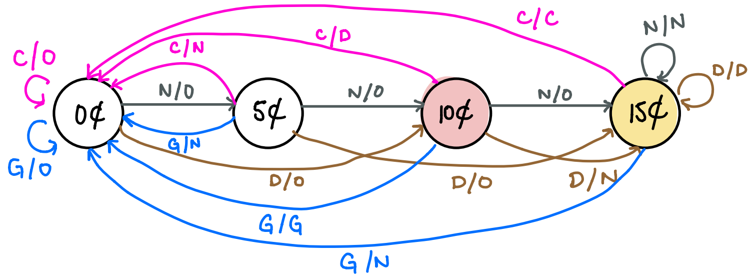
outputs: O_N = nickel
O_D = dime
O_G = gum
O_C = candy

3) state diagram

candy = 15¢
gum = 10¢



4) state reduction - don't need to keep track of different inputs to get to same state!



to reduce states:

- 2 nickels = 1 dime
- 3 nickels = 1 dime + 1 nickel
- max $ = 15 ¢ (don't care about values >)
- reduce 4 input to 2bit input

5) state assignment:

FF A = Dime

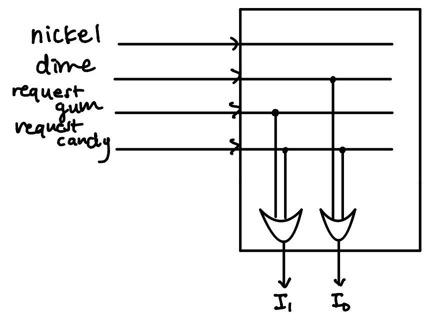
FF B = Nickel

state	FF A	FF B
0¢	0	0
5¢	0	1
10¢	1	0
15¢	1	1

Inputs:
- coin input:
  - nickel = 00
  - dime = 01
  - gum = 10
  - candy bar = 11
- need encoder for 4 to 2 bits

input	I_1	I_0
N	0	0
D	0	1
G	1	0
C	1	1

4 to 2 encoder



6) J-K flip flops (edge-triggered)

7) full-state table + flip-flop excitation table

J-K flip flop Excitation Table

Q_t	Q_{t+1}	J	K
0	0	0	x
0	1	1	X
1	0	X	1
1	1	X	0

	current state		input		next state		(D) FF A		(N) FF B		output			
											dime	nickel	gum	candy
state	A	B	I_1	I_0	A	B	JA	KA	JB	KB	O_D	O_N	O_G	O_C
0¢	0	0	0	0	0	1	0	X	1	X	0	0	0	0
0¢	0	0	0	1	1	0	1	X	0	X	0	0	0	0
0¢	0	0	1	0	0	0	0	X	0	X	0	0	0	0
0¢	0	0	1	1	0	0	0	X	0	X	0	0	0	0
5¢	0	1	0	0	1	0	1	X	X	1	0	0	0	0
5¢	0	1	0	1	1	1	1	X	X	0	0	0	0	0
5¢	0	1	1	0	0	0	0	X	X	1	0	1	0	0
5¢	0	1	1	1	0	0	0	X	X	1	0	1	0	0
10¢	1	0	0	0	1	1	X	0	1	X	0	0	0	0
10¢	1	0	0	1	1	1	X	0	1	X	0	1	0	0
10¢	1	0	1	0	0	0	X	1	0	X	0	0	1	0
10¢	1	0	1	1	0	0	X	1	0	X	1	0	0	0
15¢	1	1	0	0	1	1	X	0	X	0	0	1	0	0
15¢	1	1	0	1	1	1	X	0	X	0	1	0	0	0
15¢	1	1	1	0	0	0	X	1	X	1	0	1	1	0
15¢	1	1	1	1	0	0	X	1	X	1	0	0	0	1

8) K-maps

- Flip-Flops:

		N	N		
D	X	X	X	X	
D	X	X	X	X	I_1
	0	0	0	0	I_1
	1	1	1	0	
		I_0	I_0		

		N	N		
D	0	0	0	0	
D	1	1	1	1	I_1
	X	X	X	X	I_1
	X	X	X	X	
		I_0	I_0		

		N	N		
D	X	X	1	1	
D	X	X	0	0	I_1
	X	X	0	0	I_1
	X	X	0	1	
		I_0	I_0		

		N	N		
D	0	0	X	X	
D	1	1	X	X	I_1
	1	1	X	X	I_1
	1	0	X	X	
		I_0	I_0		

$JA = BI_1' + B'I_1'I_0$
$JA = I_1'(B + B'I_0)$

$KA = I_1$

$JB = AI_1' + A'I_1'I_0'$
$JB = I_1'(A + A'I_0')$

$KB = I_1 + A'I_1'I_0'$

- Outputs:

By inspection: $O_D = AB'I_1I_0 + ABI_1'I_0$
$O_D = AI_0(B'I_1 + BI_1')$

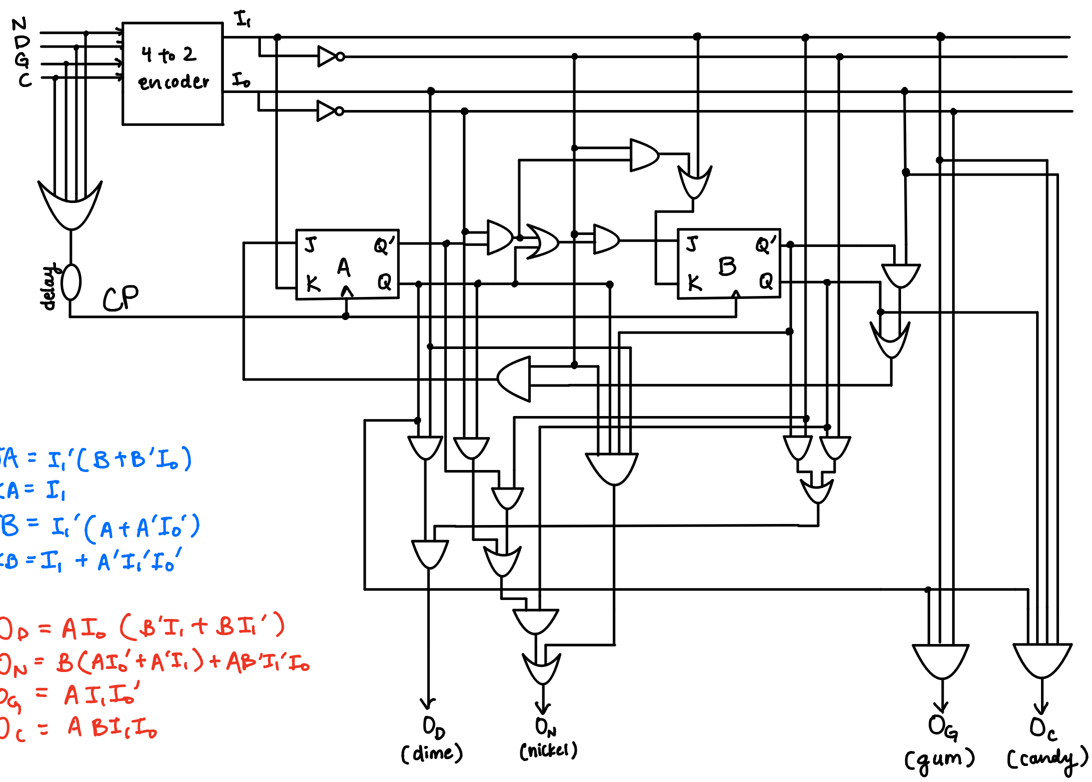
		N	N		
D	1	0	1	0	
D	1	0	0	0	I_1
	1	1	0	0	I_1
	0	0	0	0	
		I_0	I_0		

$O_N = ABI_0' + A'BI_1 + AB'I_1'I_0$
$O_N = B(AI_0' + A'I_1) + AB'I_1'I_0$

By inspection: $O_G = AB'I_1I_0' + ABI_1I_0'$
$O_G = AI_1I_0'(B' + B)$
$O_G = AI_1I_0'$

$O_C = ABI_1I_0$

9) circuit diagram



$JA = I_1'(B + B'I_0)$
$KA = I_1$
$JB = I_1'(A + A'I_0')$
$KB = I_1 + A'I_1'I_0'$

$O_D = AI_0\,(B'I_1 + BI_1')$
$O_N = B(AI_0' + A'I_1) + AB'I_1'I_0$
$O_G = A\,I_1I_0'$
$O_C = A\,BI_1I_0$

10) unused states

- there are no unused states as we cover all $2^4 = 16$ possible states.