Today

State Tables

*Shows same info as state diagram but in different form.

Curr. State	next state x=0	next state x=1	output (z) x=0	output (z) x=1
a	a	b	0	0
b	c	b	0	0
c	a	d	0	0
d	c	b	0	1

Sequential Circuit Design

* Procedure to implement a state diagram/table in/as a digit circuit.

Steps: 1) Given a verbal description, decide on states, draw diagram/table

2) Try to reduce the number of states (state reduction)

---

3) If states are "english names", then assign binary numbers to each state => diagram/table now only has "1"s + "0"s. (state assignment)

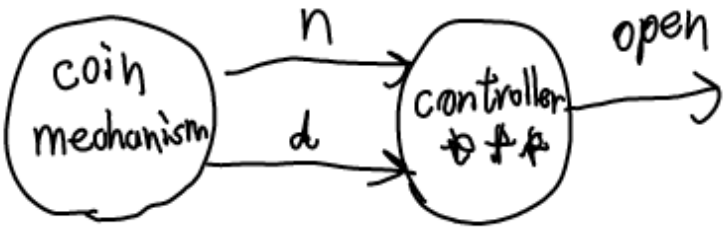
4) Pick a FF type (DFF, TFF, JK FF) to hold the state. The # of FFs depends on how many bits are needed.

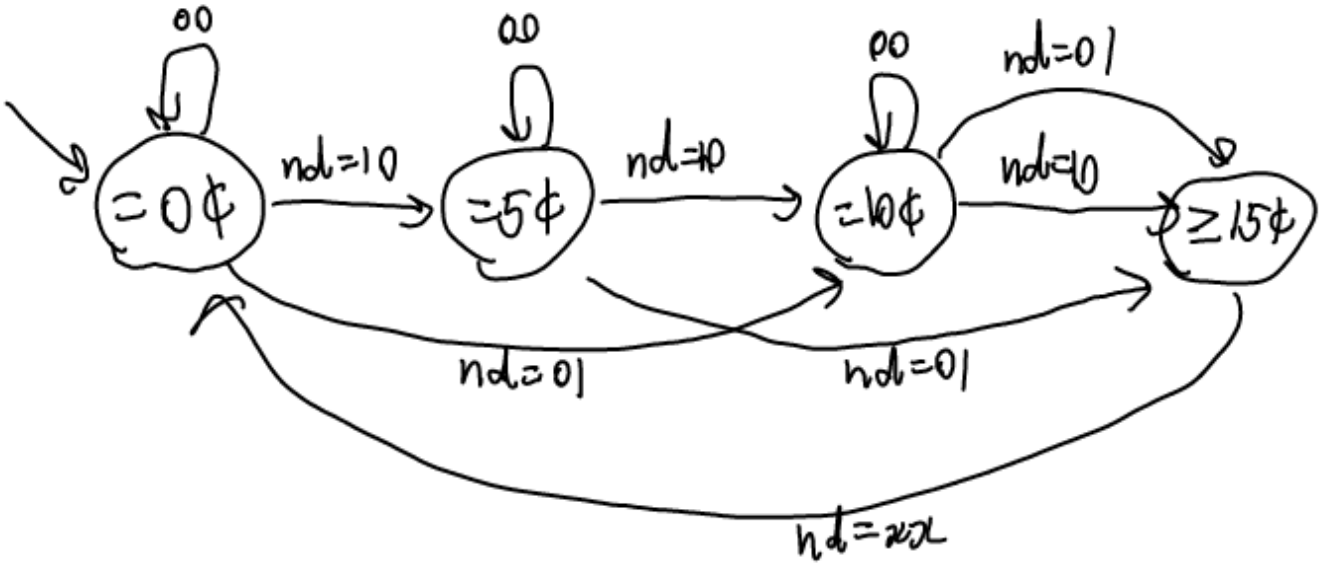
5) Derive FF input eqns. (next state eqns)

6) Derive output eqns. 7) Draw Circuit.

Eg. Design a controller for a vending machine that accepts nickels (n) and dimes (d) and oppens a door once 15¢ is deposited.

Define the states as "amount deposited so far"





* State Assignment

=0¢ → 0 → 00

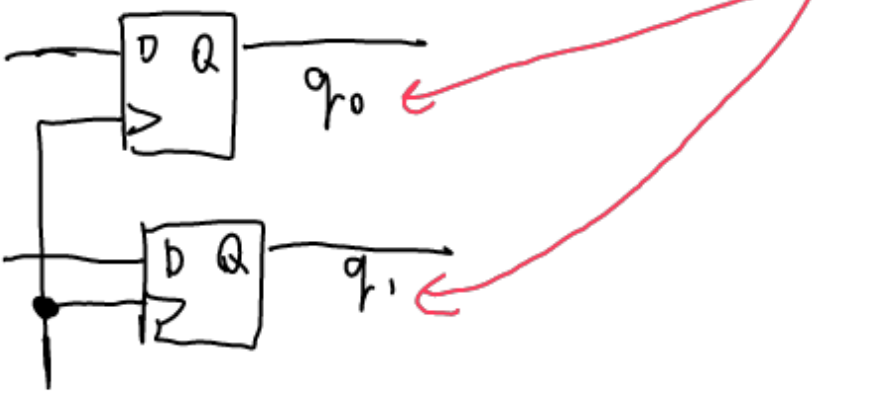
=5¢ → 1 → 01

=10¢ → 2 → 10

≥15¢ → 3 → 11

state appear in circuit at FF outputs

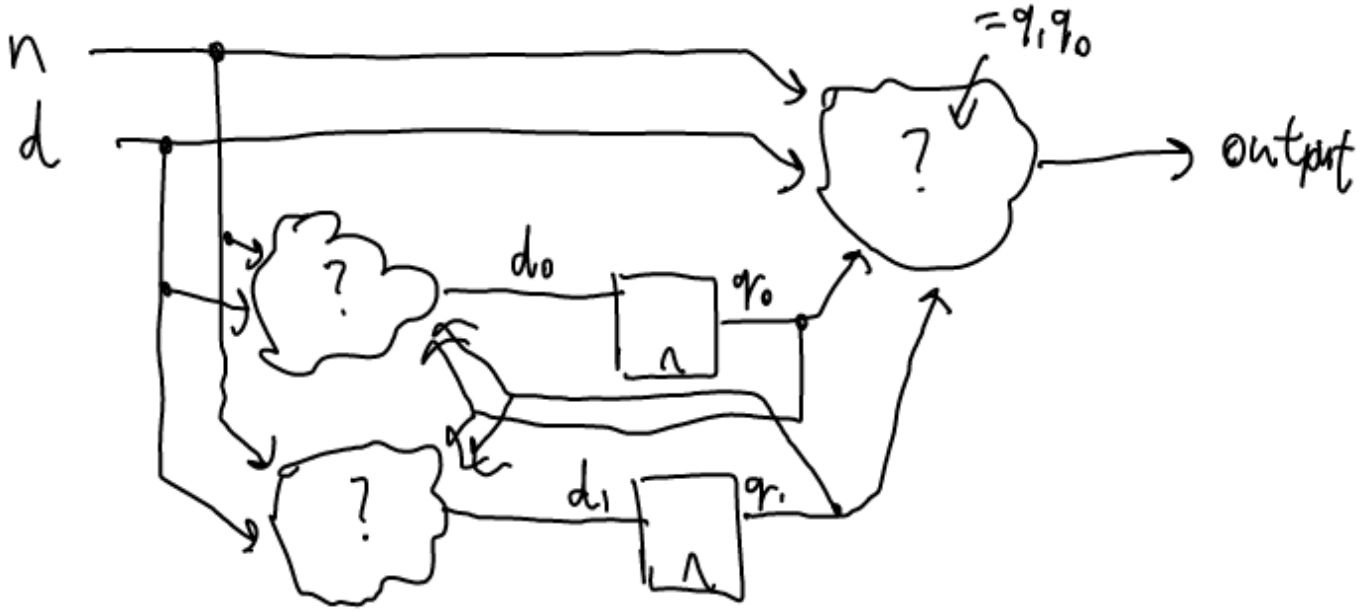
* FF selection ⇒ Need 2 of them; pick DFFs.



State table in terms of binary values

	Curr. State $q_1 q_0$	Next state (q_1, q_0) nd 00	01	10	11	output (open) nd 00	01	10	11	FF input (d_1, d_0) nd 00	01	10	11
0¢	0 0	00	10	01	XX	0	0	0	X	00	10	01	XX
5¢	0 1	01	11	10	XX	0	0	0	X	01	11	10	XX
10¢	1 0	10	11	11	XX	0	0	0	X	10	11	11	XX
15¢	1 1	11	00	00	XX	1	1	1	X	00	00	00	XX

* FF input values (d_1, d_0) depend on FF type AND next state *



* Input Eqns. (next state eqns)

d_0

$q_1 q_0$	00	01	11	10
00	0	0	X	1
01	1	1	X	0
11	0	0	X	0
10	0	1	X	1

d_1