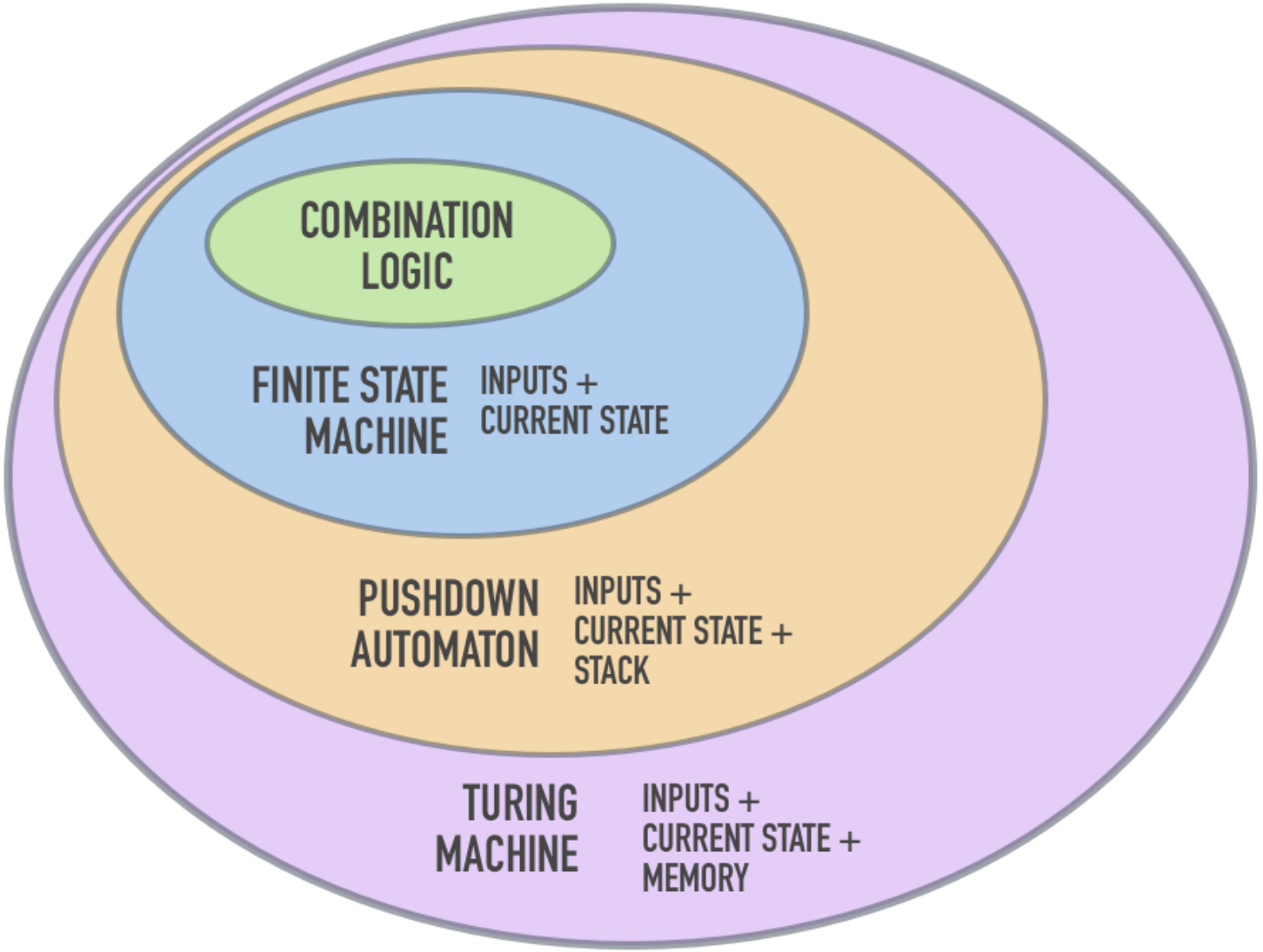
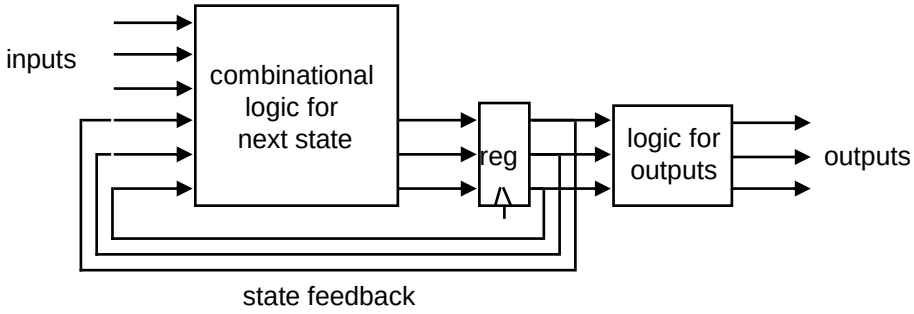
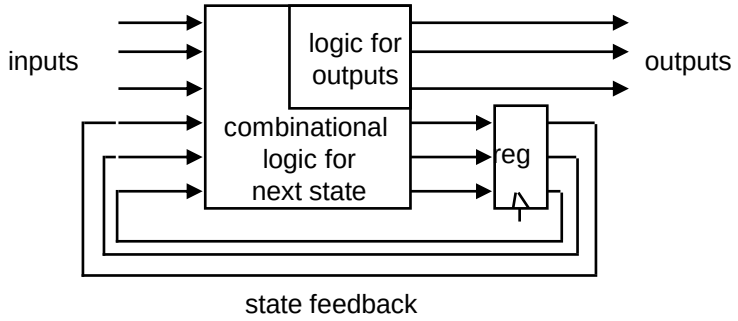
Finite-State Machines (FSMs) and Controllers


COMBINATION LOGIC
FINITE STATE MACHINE
INPUTS + CURRENT STATE
PUSHDOWN AUTOMATON
INPUTS + CURRENT STATE + STACK
TURING MACHINE
INPUTS + CURRENT STATE + MEMORY

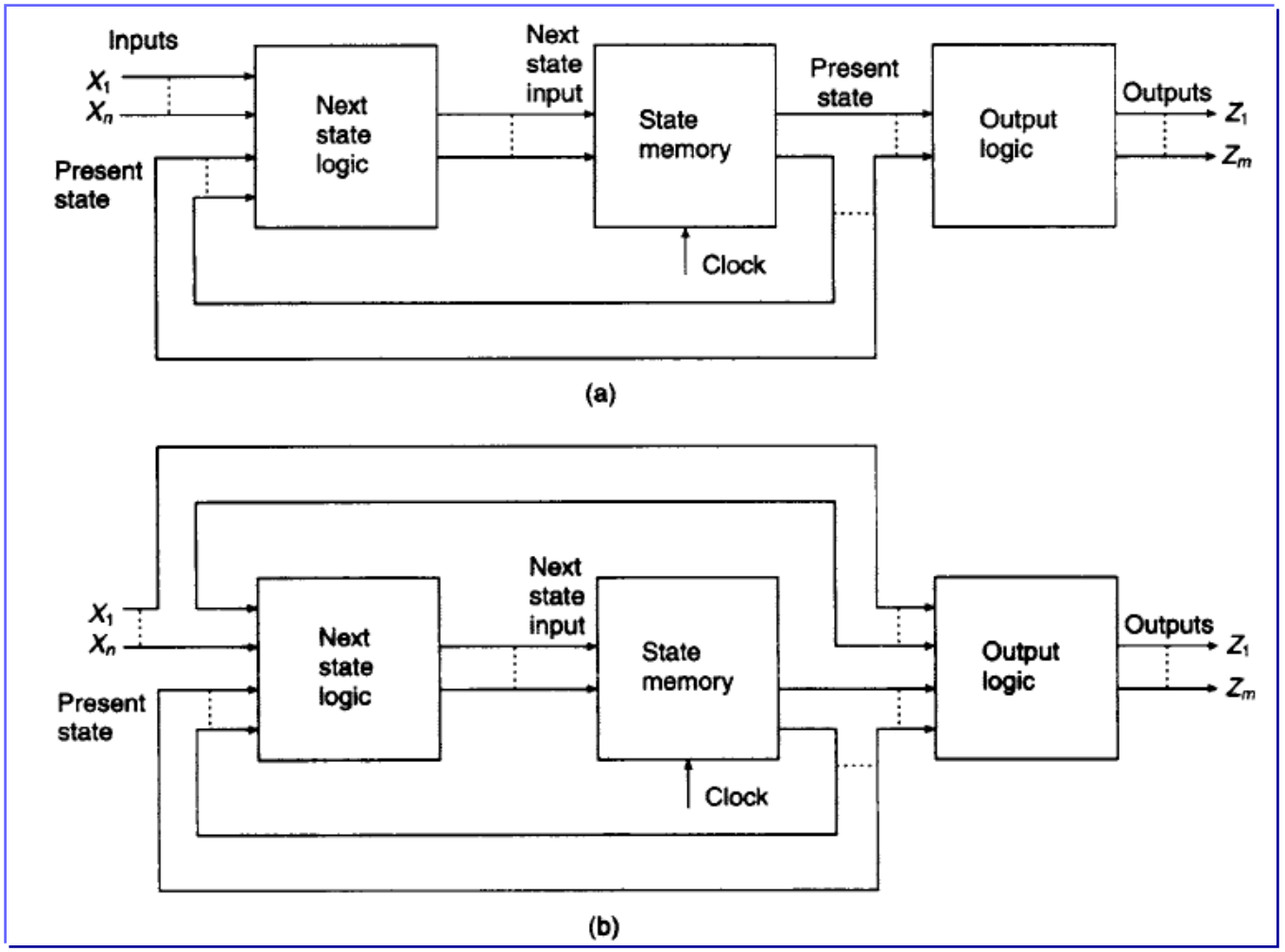
Mealy and Moore machines

- Mealy machines tend to have fewer states
- Mealy machines react faster to inputs
- Moore machines are generally safer to use
  - outputs change at clock edge (always one cycle later)
  - in Mealy machines, input change can cause output change as soon as logic is done





Mealy and Moore machines



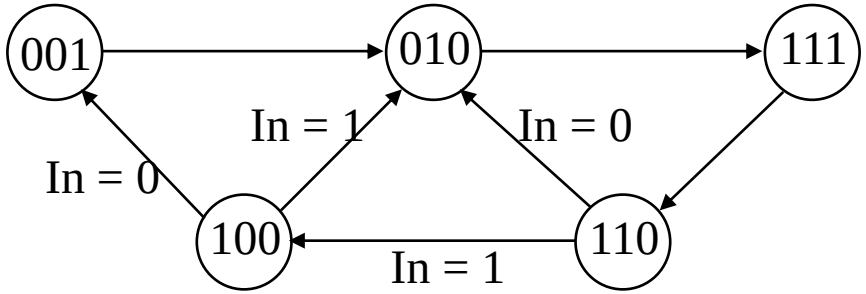
Moore (a) and Mealy (b)

State : SR, D, T, J-K, memory

Finite state machine representations

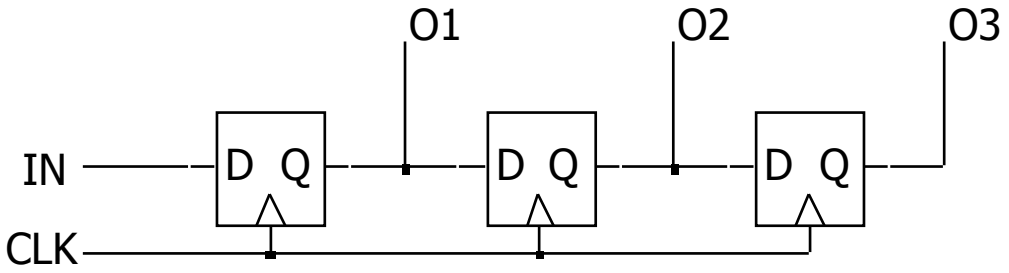
- States: determined by possible values in sequential storage elements
- Transitions: change of state
- Clock: controls when state can change by controlling storage elements

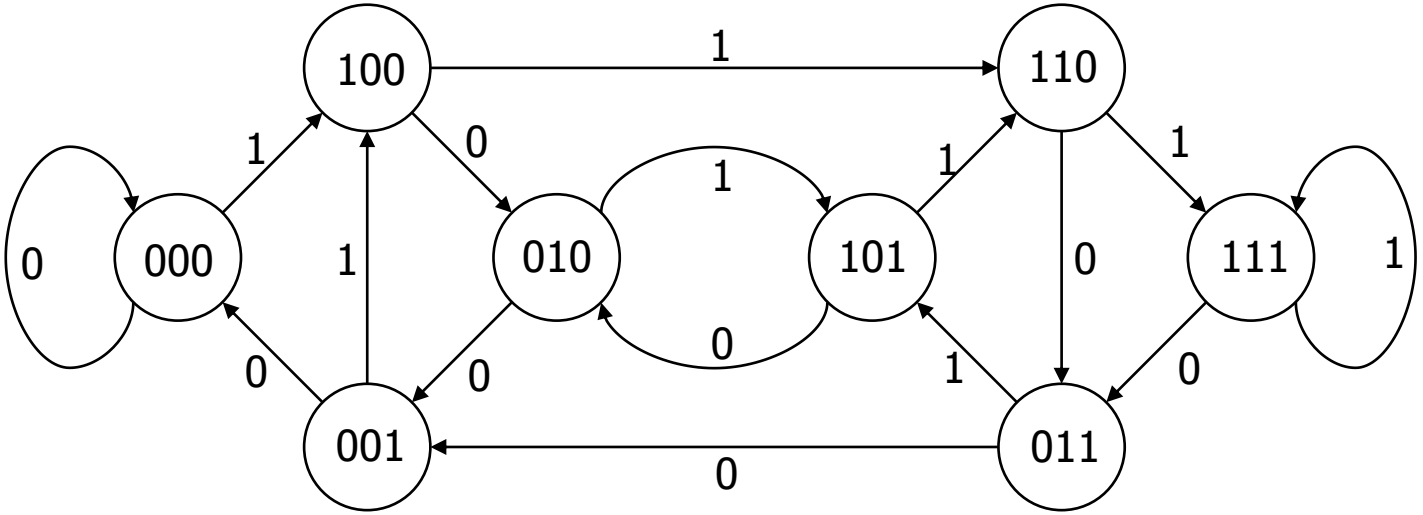
- Sequential logic
  - sequences through a series of states
  - based on sequence of values on input signals
  - clock period defines elements of sequence



sequential system- represented with a state diagram

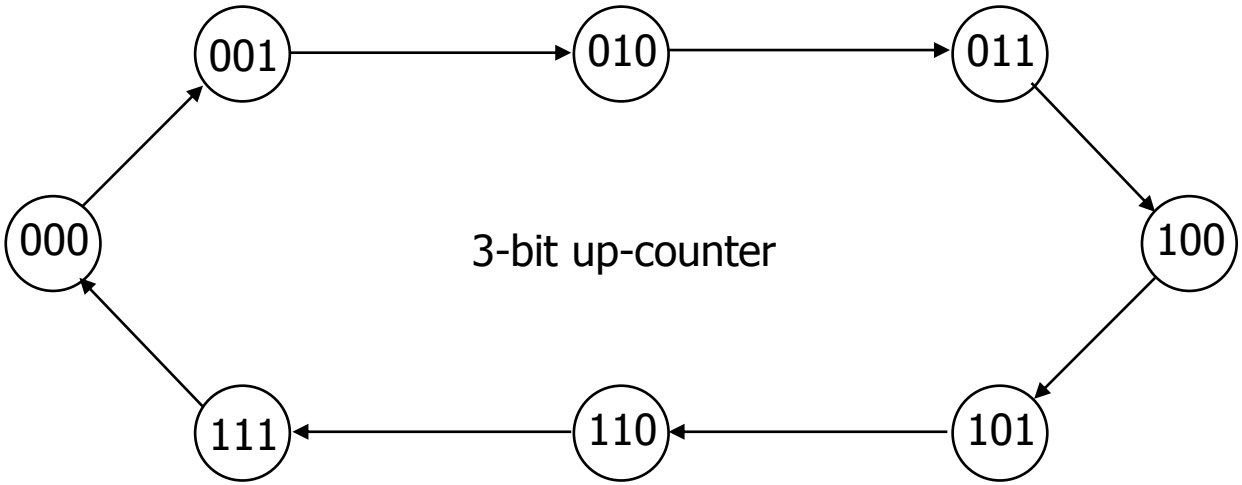
- Shift register
  - input value shown on transition arcs
  - output values shown within state node





Counters are simple finite state machines

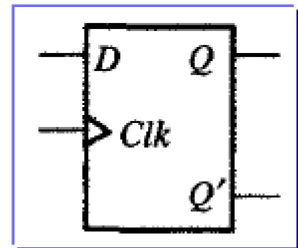
- Counters
  - proceed through well-defined sequence of states in response to enable
- Many types of counters: binary, BCD, Gray-code
  - 3-bit up-counter: 000, 001, 010, 011, 100, 101, 110, 111, 000, ...
  - 3-bit down-counter: 111, 110, 101, 100, 011, 010, 001, 000, 111, ...



How do we turn a state diagram into logic?

- Counter
  - 3 flip-flops to hold state
  - logic to compute next state
  - clock signal controls when flip-flop memory can change
    - wait long enough for combinational logic to compute new value
    - don't wait too long as that is low performance

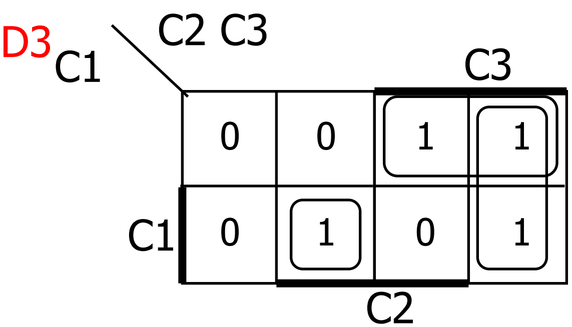
How do we turn a state diagram into logic?

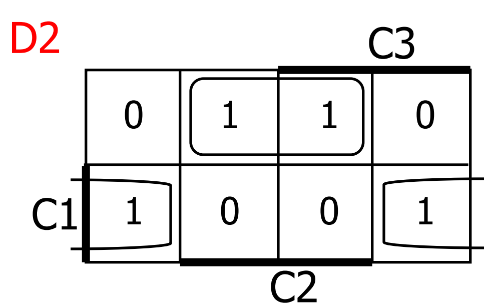


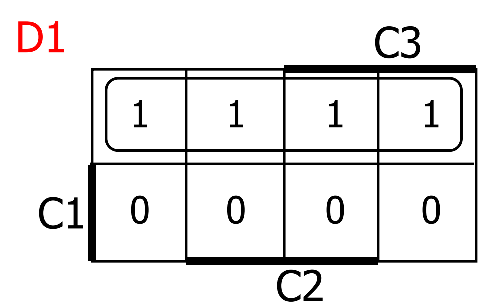
Q	Q(next)	D
0	0	0
0	1	1
1	0	0
1	1	1

Present state			Next state			Flip flop input		
C3	C2	C1	N3	N2	N1	D3	D2	D1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

D1 = C1′
D2 = C1C2′ + C1′C2
= C1 xor C2
D3 = C1C2C3′ + C1′C3 + C2′C3
= (C1C2)C3′ + (C1′ + C2′)C3
= (C1C2)C3′ + (C1C2)′C3
= (C1C2) xor C3

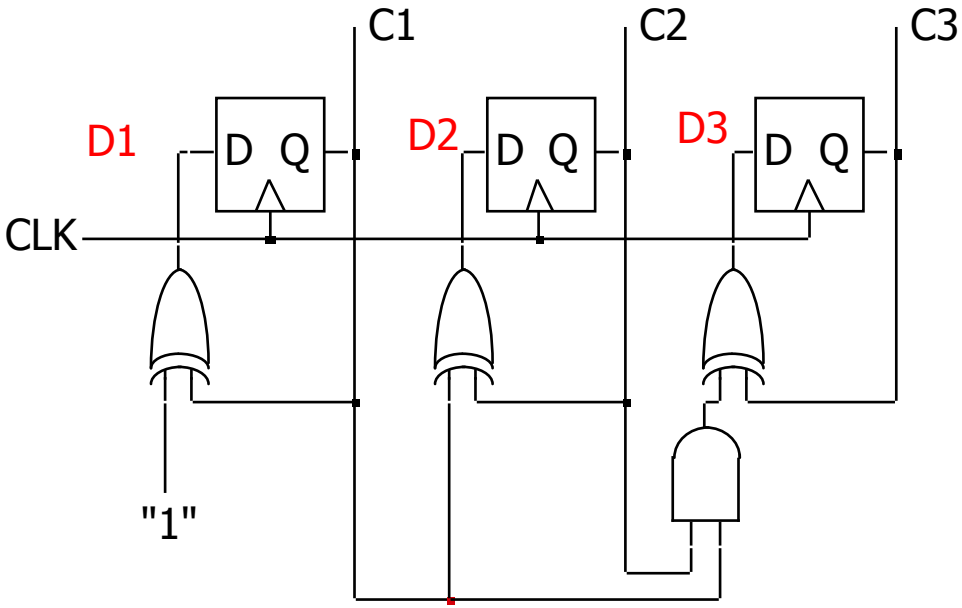






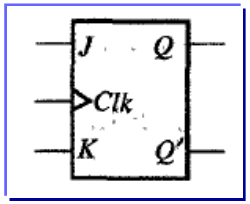
How do we turn a state diagram into logic? (3 bit counter)

- Counter
  - 3 flip-flops to hold state
  - logic to compute next state
  - clock signal controls when flip-flop memory can change
    - wait long enough for combinational logic to compute new value
    - don't wait too long as that is low performance



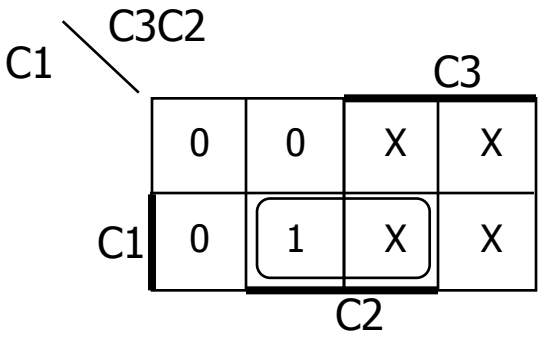
How do we turn a state diagram into logic? (using J-K)

Q	Q(next)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

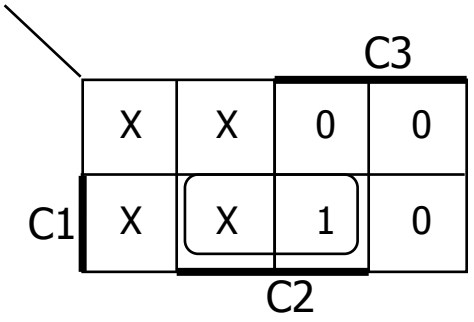


Present state Next state Flip flop input

C3	C2	C1	N3	N2	N1	J3	K3	J2	K2	J1	K1
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1



J3=C1C2

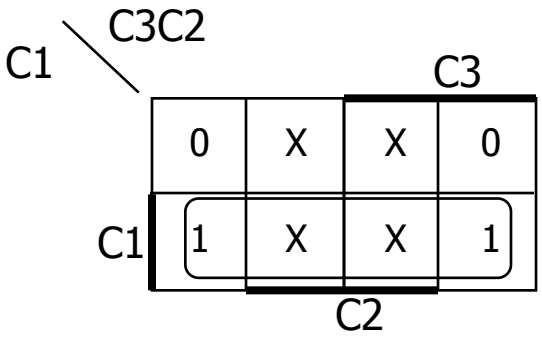


K3=C1C2

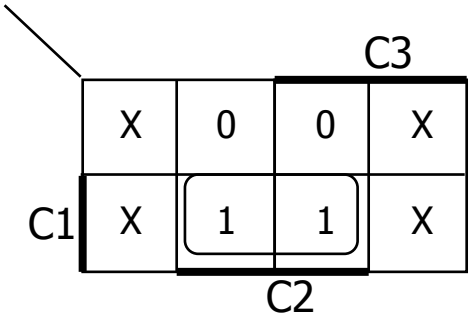
How do we turn a state diagram into logic? (using J-K)

Present state Next state Flip flop input

C3	C2	C1	N3	N2	N1	J3	K3	J2	K2	J1	K1
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1



J2=C1

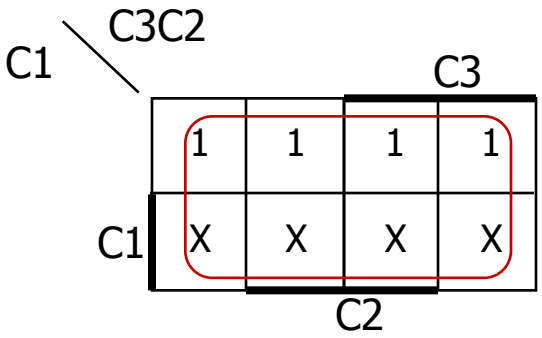


K2=C1

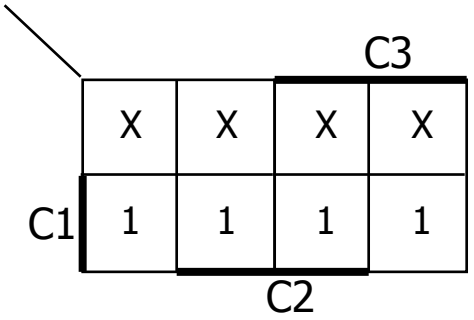
How do we turn a state diagram into logic? (using J-K)

Present state Next state Flip flop input

C3	C2	C1	N3	N2	N1	J3	K3	J2	K2	J1	K1
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1



J1=1



K1=1

How do we turn a state diagram into logic? (using J-K)

Present state Next state

C3	C2	C1	N3	N2	N1
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Flip flop input

J3	K3	J2	K2	J1	K1
0	X	0	X	1	X
0	X	1	X	X	1
0	X	X	0	1	X
1	X	X	1	X	1
X	0	0	X	1	X
X	0	1	X	X	1
X	0	X	0	1	X
X	1	X	1	X	1

J3=C1C2

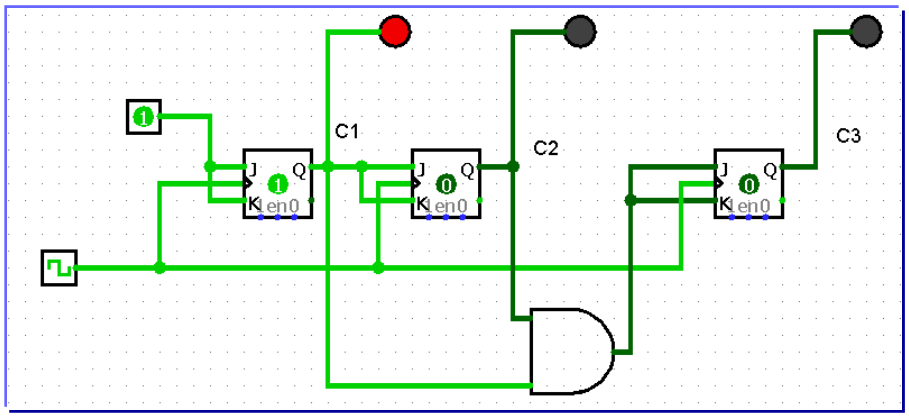
K3=C1C2

J2=C1

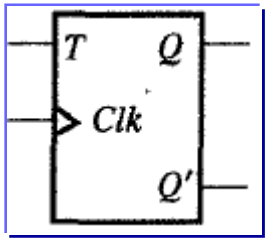
K2=C1

J1=1

K1=1



How do we turn a state diagram into logic? (Using T-FF)



Q	Q(next)	T
0	0	0
0	1	1
1	0	1
1	1	0

Present state			Next state			Flip flop input		
C3	C2	C1	N3	N2	N1	T3	T2	T1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

T1 = 1
T2= C1
T3 = C2C1

T1

C1 \ C2 C2			C3	
	1	1	1	1
C1	1	1	1	1
		C2		

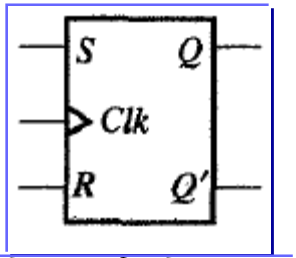
T2

			C3	
	0	0	0	0
	1	1	1	1
		C2		

T1

			C3	
	0	0	0	0
C1	0	1	1	0
		C2		

How do we turn a state diagram into logic? (Using SR-FF)



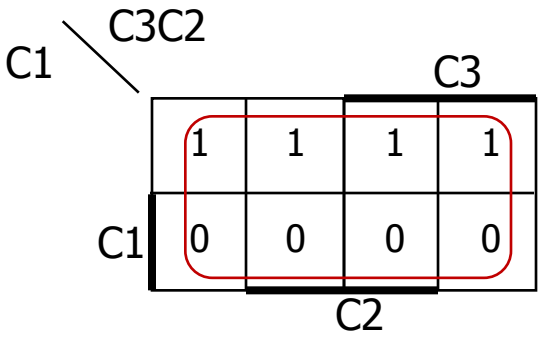
Q	Q(next)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Present state Next state

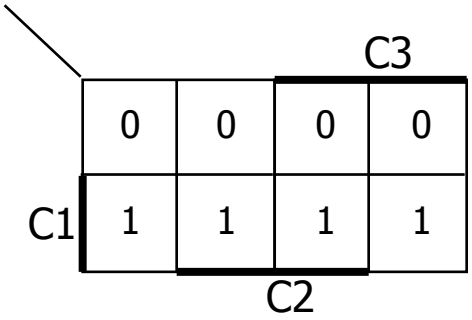
C3	C2	C1	N3	N2	N1
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Flip flop input

S3	R3	S2	R2	S1	R1
0	X	0	X	1	0
0	X	1	0	0	1
0	X	X	0	1	0
1	0	0	1	0	1
X	0	0	X	1	0
X	0	1	0	0	1
X	0	X	0	1	0
0	1	0	1	0	1

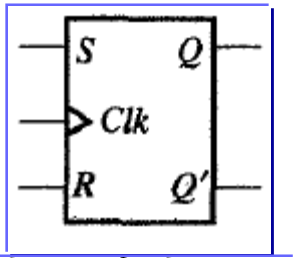


S1=C1'



R1=C1

How do we turn a state diagram into logic? (Using S-R FF)



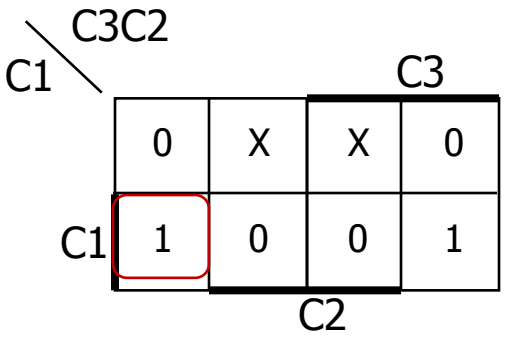
Q	Q(next)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Present state Next state

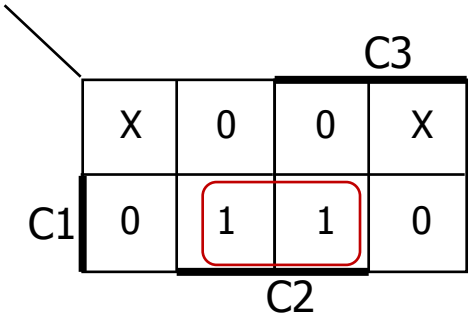
C3	C2	C1	N3	N2	N1
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Flip flop input

S3	R3	S2	R2	S1	R1
0	X	0	X	1	0
0	X	1	0	0	1
0	X	X	0	1	0
1	0	0	1	0	1
X	0	0	X	1	0
X	0	1	0	0	1
X	0	X	0	1	0
0	1	0	1	0	1

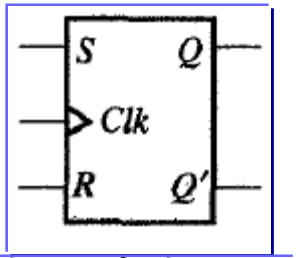


S2=C1C2'



R2=C1C2

How do we turn a state diagram into logic? (Using S-R FF)



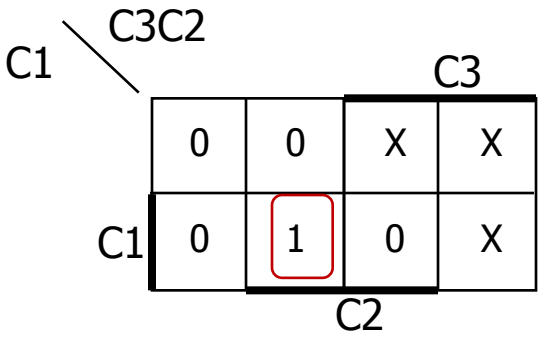
Q	Q(next)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Present state Next state

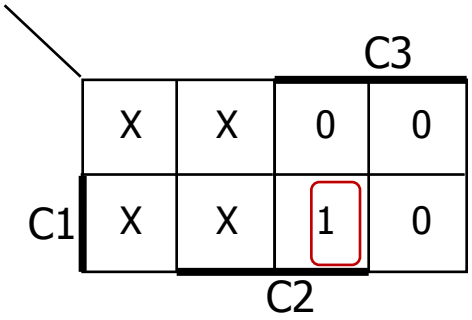
C3	C2	C1	N3	N2	N1
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Flip flop input

S3	R3	S2	R2	S1	R1
0	X	0	X	1	0
0	X	1	0	0	1
0	X	X	0	1	0
1	0	0	1	0	1
X	0	0	X	1	0
X	0	1	0	0	1
X	0	X	0	1	0
0	1	0	1	0	1



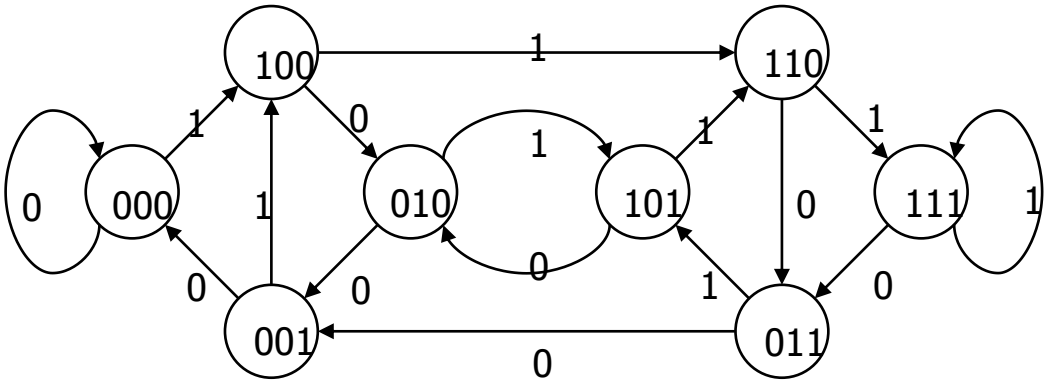
S3=C1C2C3'



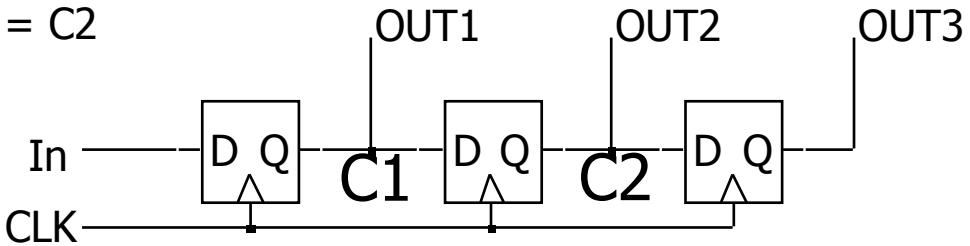
R3=C1C2C3

shift register

	Present state			Next state		
In	C1	C2	C3	N1	N2	N3
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	0	1	1	1	0	1
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	1
1	1	1	1	1	1	1



D1 = In
D2 = C1
D3 = C2



Do as an exercise:flip flop inputs table, logic (D1, D2, D3) and Simplification