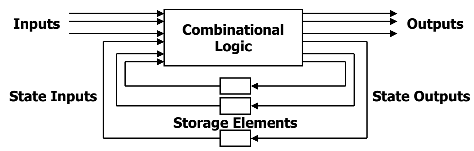
Overview

- Last lecture
  - Cascading flip-flops
  - Clock skew
  - Registers
- Today
  - Introduction to finite state machines
    - State diagrams
  - Counters as finite state machines
    - Counter design

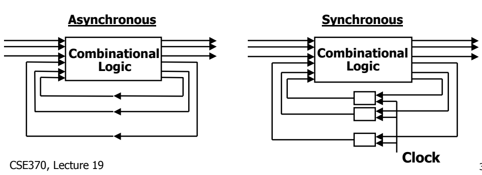
State machines

- Combinational logic and storage elements
  - Localized feedback loops
  - Choice of storage elements alters the logic
    - D flip-flop, T flip-flop, etc.



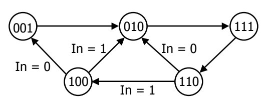
Asynchronous versus synchronous

- Asynchronous
  - State changes occur when state inputs change
  - Feedback elements may be wires or delays
- Synchronous
  - State changes occur synchronously
  - Feedback elements are clocked



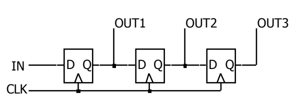
Finite-state machines

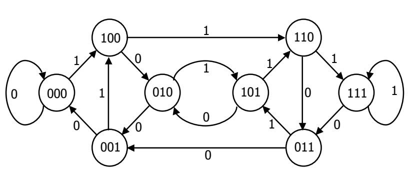
- States: Possible storage-element values
- Transitions: Changes in state
  - Clock synchronizes the state changes
- Sequential logic
  - Sequences through a series of states
  - Based on inputs and present state



Drawing state diagrams

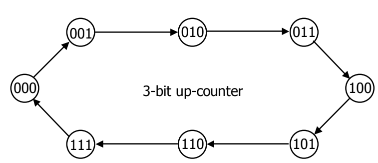
- Show input values on transition arcs
- Show output values in state nodes





Begin by studying counters

- Simple state machines
  - Output is the counter's state
- Next state is well defined
  - Does not depend on input (no inputs)

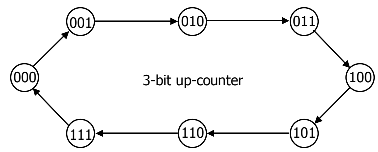


Counter design procedure

1. Draw a state diagram
2. Draw a state-transition table
3. Encode the next-state functions
   - Minimize the logic using k-maps
4. Implement the design

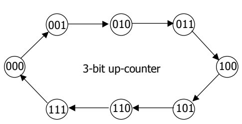
Example: Design the 3-bit up counter

1. Draw a state diagram



2. Draw a state-transition table

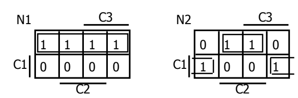
- Like a truth-table
  - State encoding is easy for counters → Use count value



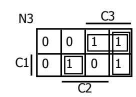
	current state	next state	
0	000	001	1
1	001	010	2
2	010	011	3
3	011	100	4
4	100	101	5
5	101	110	6
6	110	111	7
7	111	000	0

3. Encode the next state functions

- Assume D flip-flops as state elements



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



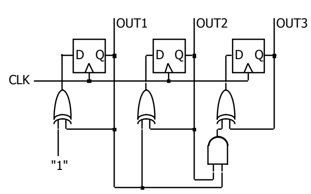
N1 := C1'

N2 := C1C2' + C1'C2
:= C1 xor C2

N3 := C1C2C3' + C1'C3 + C2'C3
:= C1C2C3' + (C1' + C2')C3
:= (C1C2) xor C3

4. Implement the design

- 3 flip-flops hold state
  - Counter is synchronously clocked
- Minimized logic computes next state

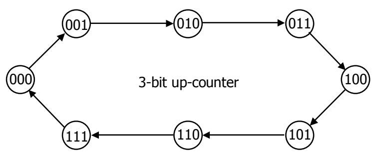


Class example

- Redesign the 3-bit up counter using T flip-flops

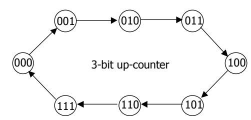
1. Draw a state diagram
2. Draw a state-transition table
3. Encode the next-state functions
   - Minimize the logic using k-maps
4. Implement the design

1. Draw a state diagram



2. Draw a state-transition table

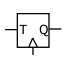
- Like a truth-table
  - State encoding is easy for counters → Use count value

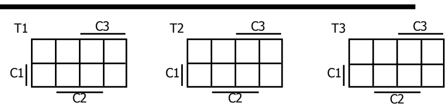


current state		next state	
0	000	001	1
1	001	010	2
2	010	011	3
3	011	100	4
4	100	101	5
5	101	110	6
6	110	111	7
7	111	000	0

3. Encode the next state functions

T flip-flops





T1 :=

T2 :=

T3 :=

C3	C2	C1	N3	N2	N1	T3	T2	T1
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			

4. Implement the design: