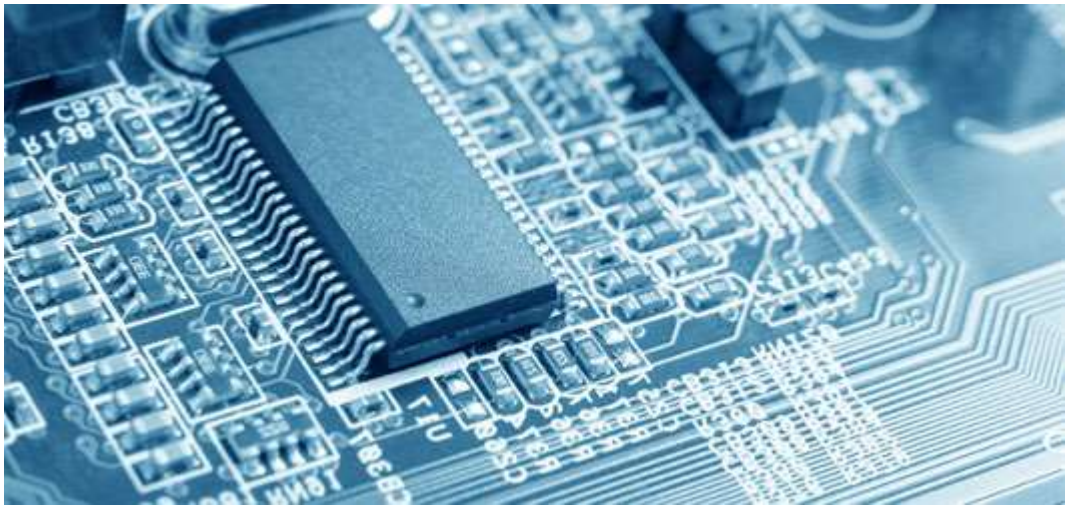
NAZARBAYEV UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY



ROBT206 – Microcontrollers with Lab

Lecture 19 – Counters

3 April, 2018

Topics

Today's Topics

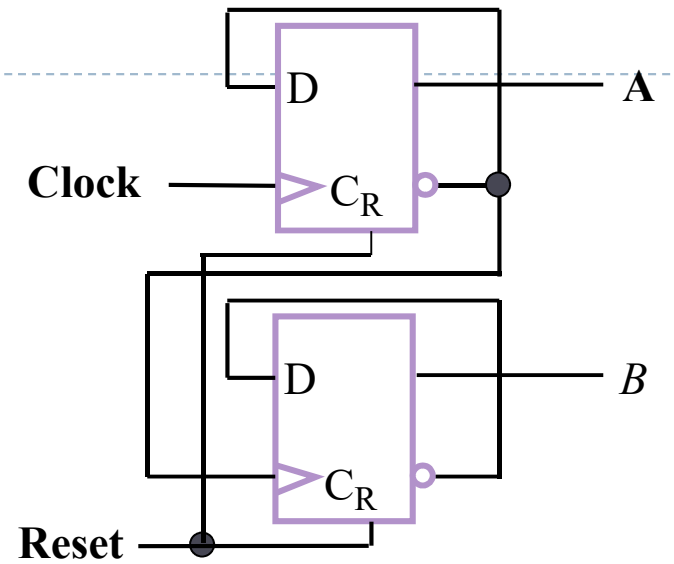
- Microoperations on single register (continued)
  - Counters

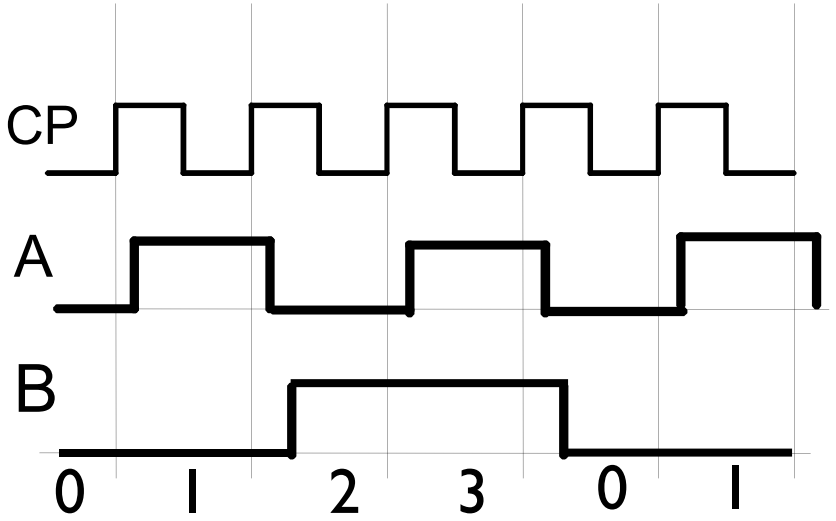
Counters

- **Counters** are sequential circuits which "count" through a specific state sequence. They can **count up**, **count down**, or **count through other fixed sequences**. Two distinct types are in common usage:
- Ripple Counters
  - Clock connected to the flip-flop clock input on the LSB bit flip-flop
  - For all other bits, a flip-flop output is connected to the clock input, thus circuit is not truly synchronous!
  - Output change is delayed more for each bit toward the MSB.
  - Resurgent because of low power consumption
- Synchronous Counters
  - Clock is directly connected to the flip-flop clock inputs
  - Logic is used to implement the desired state sequencing

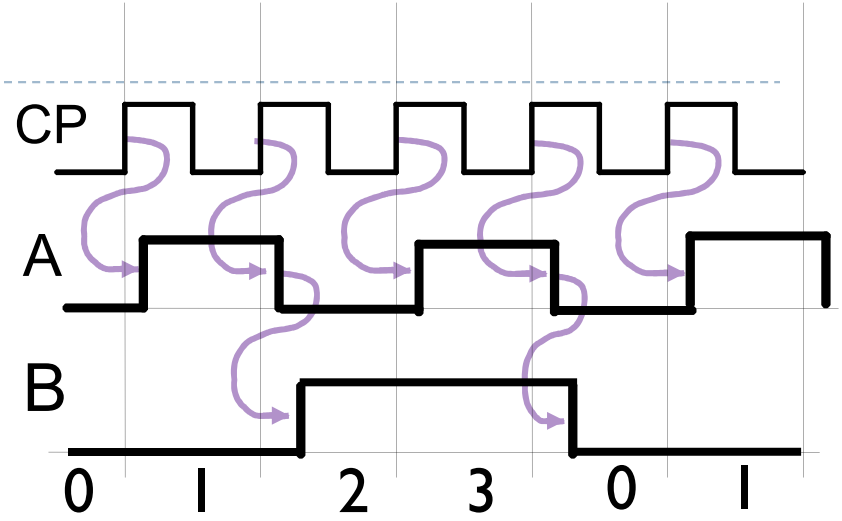
Ripple Counter

- How does it work?
  - When there is a positive edge on the clock input of A, A complements
  - The clock input for flip-flop B is the complemented output of flip-flop A
  - When flip-flop A changes from 1 to 0, there is a positive edge on the clock input of B causing B to complement





Ripple Counter



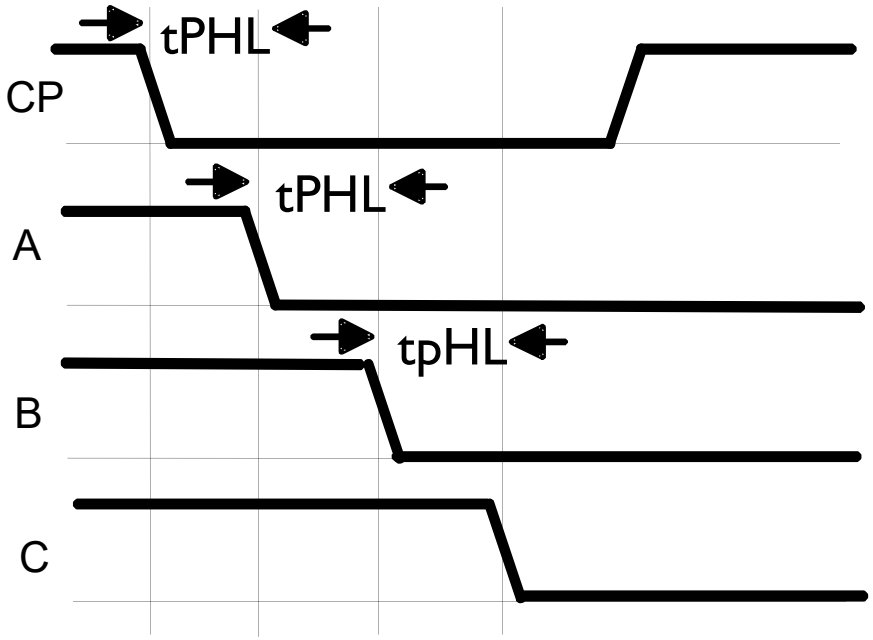
- The arrows show the cause-effect relation-ship from the prior slide
- The corresponding sequence of states (B,A) = (0,0),(0,1),(1,0),(1,1),(0,0),(0,1), …
- Each additional bit, C, D, …behaves like bit B, changing half as frequently as the bit before it.
- For 3 bits: (C,B,A) = (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1), (0,0,0), …

Ripple Counter (Continued)

- These circuits are called *ripple counters* because each edge sensitive transition (positive in the example) causes a change in the next flip-flop's state.
- The changes "ripple" upward through the chain of flip-flops, i. e., each transition occurs after a clock-to-output delay from the stage before.

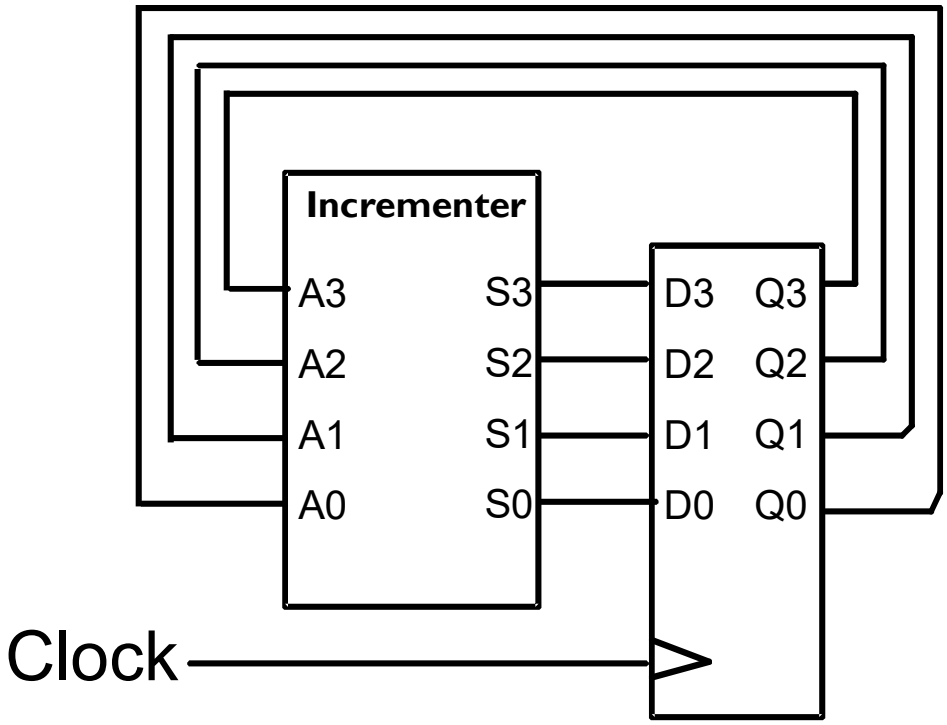
Ripple Counter (Continued)

- Starting with C = B = A = 1, equivalent to (C,B,A) = 7 base 10, the next clock increments the count to (C,B,A) = 0 base 10. In fine timing detail:
  - The clock to output delay t_{PHL} causes an increasing delay from clock edge for each stage transition.
  - Thus, the count "ripples" from least to most significant bit.
  - For *n* bits, total worst case delay is *n* t_{PHL}.



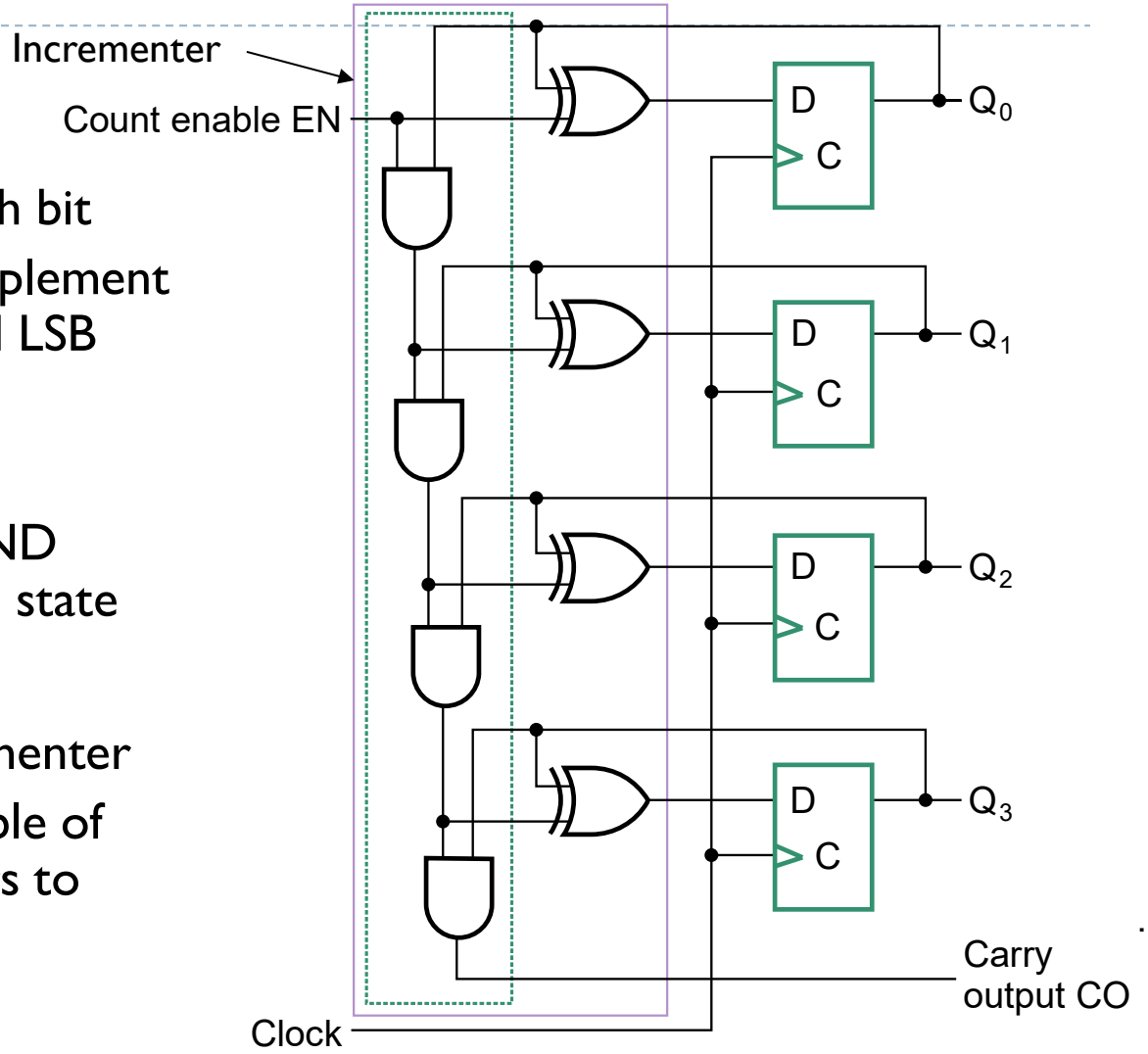
Synchronous Counters

- To eliminate the "ripple" effects, use a common clock for each flip-flop and a combinational circuit to generate the next state.
- For an up-counter, use an incrementer =>



Synchronous Counters

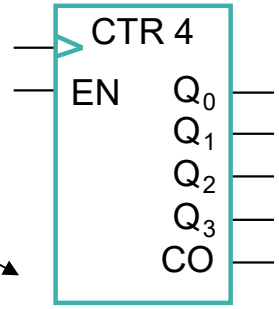
- Internal details →
- Internal Logic
  - XOR complements each bit
  - AND chain causes complement of a bit if all bits toward LSB from it equal 1
- Count Enable
  - Forces all outputs of AND chain to 0 to "hold" the state
- Carry Out
  - Added as part of incrementer
  - Connect to Count Enable of additional 4-bit counters to form larger counters



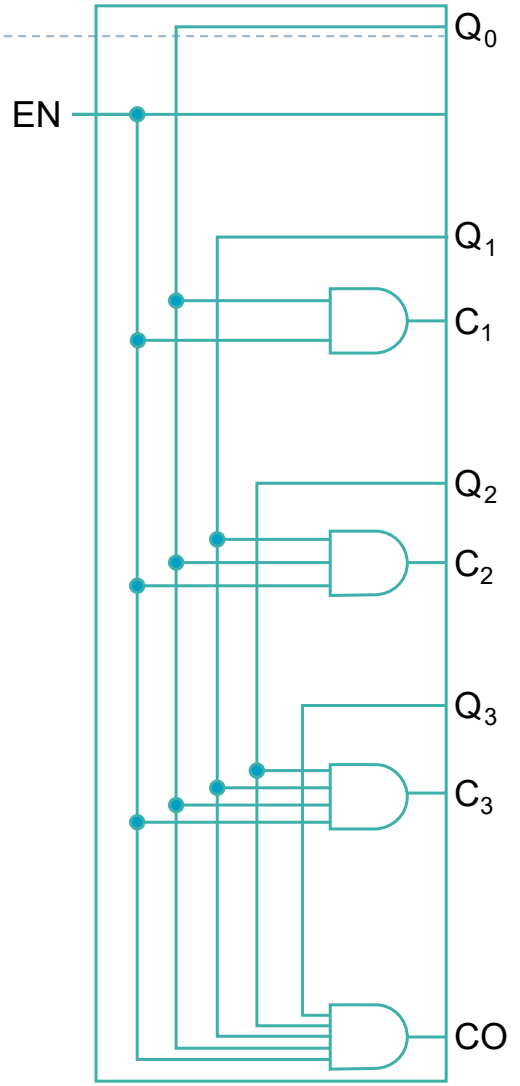
(a) Logic Diagram-Serial Gating

Synchronous Counters

- Carry chain
  - series of AND gates through which the carry "ripples"
  - Yields long path delays
  - Called *serial gating*
- Replace AND carry chain with ANDs => in parallel
  - Reduces path delays
  - Called *parallel gating*
  - Like carry lookahead
  - Lookahead can be used on COs and ENs to prevent long paths in large counters
- Symbol for Synchronous Counter



Symbol

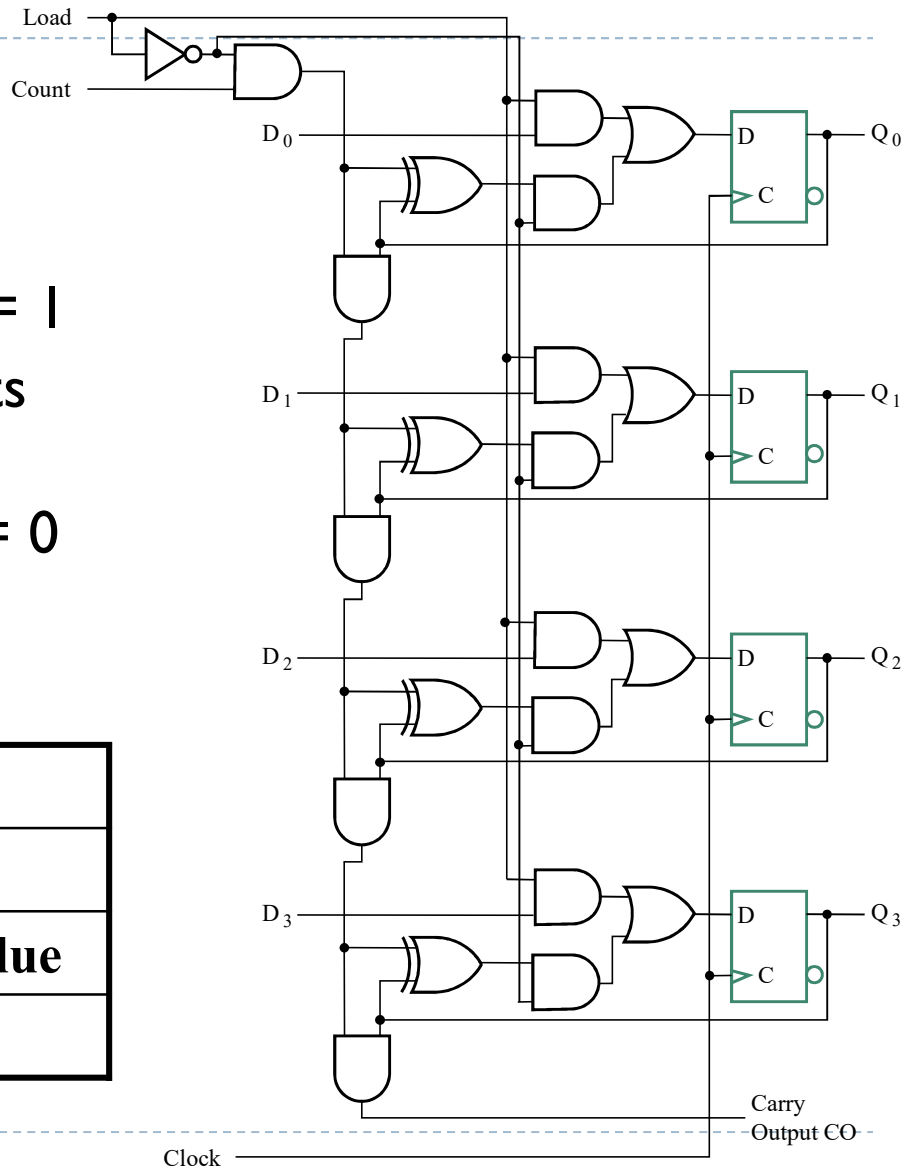


Logic Diagram-Parallel Gating

Counter with Parallel Load

- Add path for input data
  - enabled for Load = 1
- Add logic to:
  - disable count logic for Load = 1
  - disable feedback from outputs for Load = 1
  - enable count logic for Load = 0 and Count = 1
- The resulting function table:

Load	Count	Action
0	0	Hold Stored Value
0	1	Count Up Stored Value
1	X	Load D



Design Example: Synchronous BCD Counter

- Use the sequential logic model to design a synchronous BCD counter with D flip-flops
- State Table =>
- Input combinations 1010 through 1111 are don't cares

Current State Q8 Q4 Q2 Q1	Next State Q8 Q4 Q2 Q1
0 0 0 0	0 0 0 1
0 0 0 1	0 0 1 0
0 0 1 0	0 0 1 1
0 0 1 1	0 1 0 0
0 1 0 0	0 1 0 1
0 1 0 1	0 1 1 0
0 1 1 0	0 1 1 1
0 1 1 1	1 0 0 0
1 0 0 0	1 0 0 1
1 0 0 1	0 0 0 0

Synchronous BCD (Continued)

- Use K-Maps to two-level optimize the next state equations and manipulate into forms containing XOR gates:

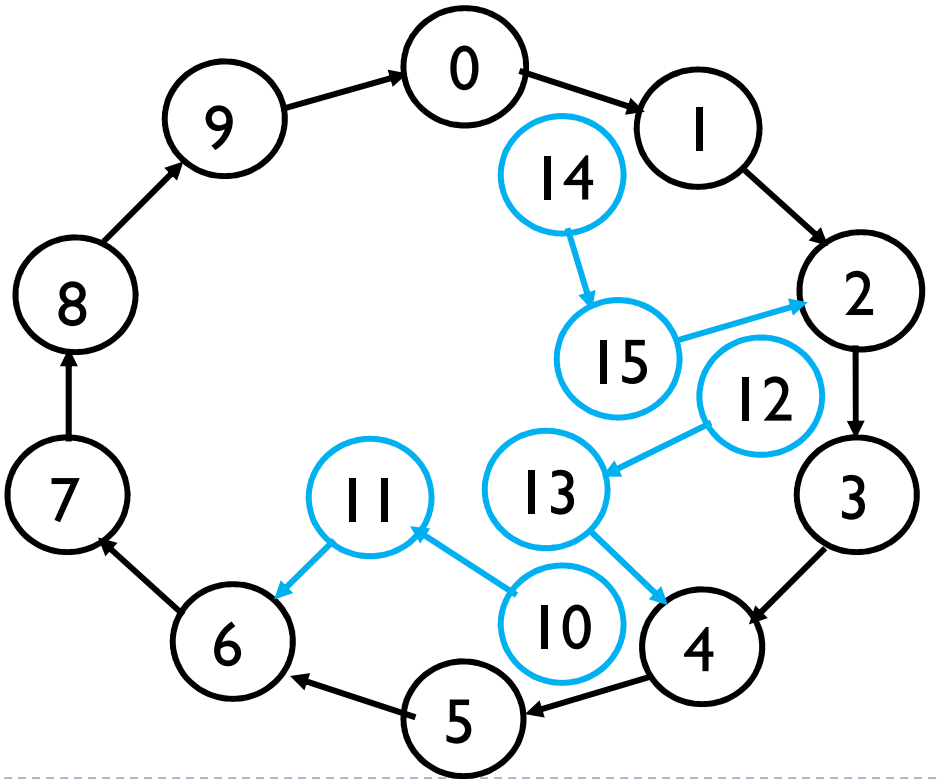
$$D1 = \overline{Q1}$$
$$D2 = Q2 \oplus Q1\overline{Q8}$$
$$D4 = Q4 \oplus Q1Q2$$
$$D8 = Q8 \oplus (Q1Q8 + Q1Q2Q4)$$

- The logic diagram can be drawn from these equations
  - An asynchronous or synchronous reset should be added
- For the BCD counter design, if an invalid state is entered, return to a valid state occurs within two clock cycles

Synchronous BCD (Continued)

- Find the actual values of the six next states for the don't care combinations from the equations
- Find the overall state diagram to assess behavior for the don't care states (states in decimal)

Present State				Next State			
Q8	Q4	Q2	Q1	Q8	Q4	Q2	Q1
1	0	1	0	1	0	1	1
1	0	1	1	0	1	1	0
1	1	0	0	1	1	0	1
1	1	0	1	0	1	0	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	1	0



Synchronous BCD

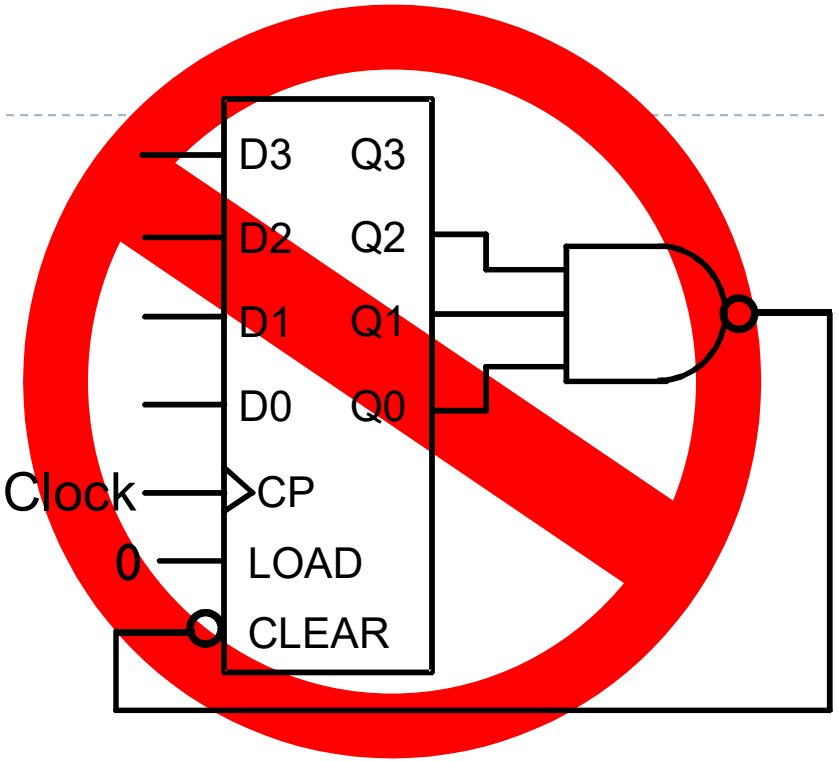
- For the BCD counter design, if an invalid state is entered, return to a valid state occurs within two clock cycles
- Is this adequate? If not:
  - Is a signal needed that indicates that an invalid state has been entered? What is the equation for such a signal?
  - Does the design need to be modified to return from an invalid state to a valid state in one clock cycle?
  - Does the design need to be modified to return from a invalid state to a specific state (such as 0)?
- The action to be taken depends on:
  - the application of the circuit
  - design group policy

Counting Modulo N

- The following techniques use an *n*-bit binary counter with asynchronous or synchronous clear and/or parallel load:
  - Detect a *terminal count* of N in a Modulo-N count sequence to asynchronously Clear the count to 0 or asynchronously Load in value 0 (These lead to counts which are present for only a very short time and can fail to work for some timing conditions!)
  - Detect a terminal count of N - 1 in a Modulo-N count sequence to Clear the count synchronously to 0
  - Detect a terminal count of N - 1 in a Modulo-N count sequence to synchronously Load in value 0
  - Detect a terminal count and use Load to preset a count of the terminal count value minus (N - 1)
- Alternatively, custom design a modulo N counter as done for BCD

Counting Modulo 7: Detect 7 and Asynchronously Clear

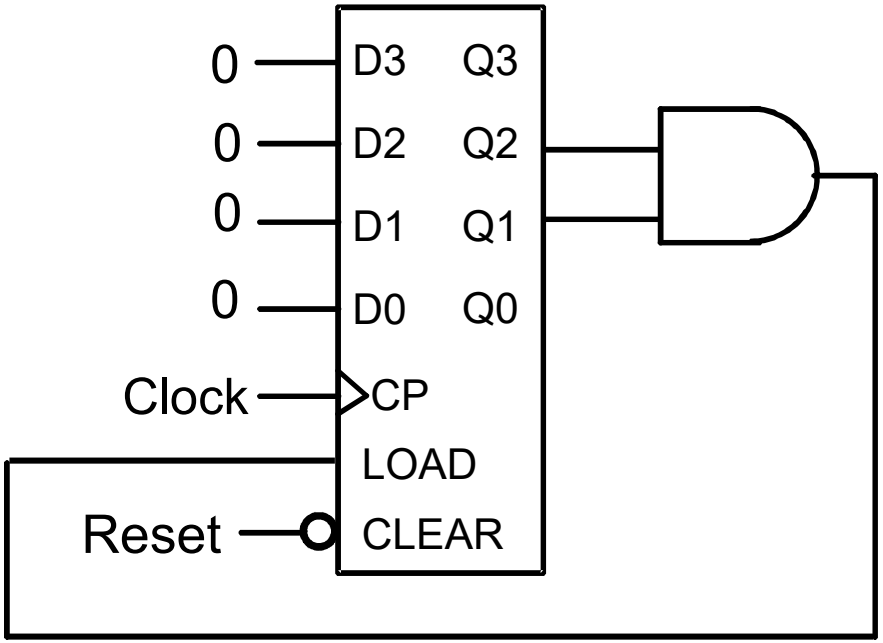
- A synchronous 4-bit binary counter with an asynchronous Clear is used to make a Modulo 7 counter.
- Use the Clear feature to detect the count 7 and clear the count to 0. This gives a count of 0, 1, 2, 3, 4, 5, 6, 7(short)0, 1, 2, 3, 4, 5, 6, 7(short)0, etc.



- **DON'T DO THIS! Existence of state 7 may not be long enough to reliably reset all flip-flops to 0. Referred to as a "suicide" counter! (Count "7" is "killed," but the designer's job may be dead as well!)**

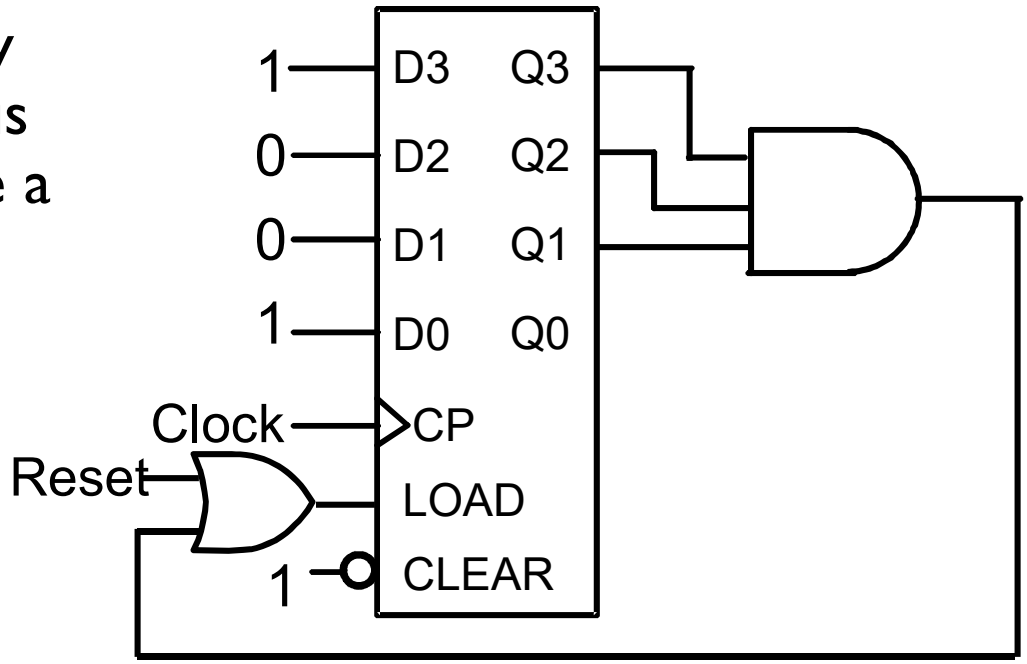
Counting Modulo 7: Synchronously Load on Terminal Count of 6

- A synchronous 4-bit binary counter with a synchronous load and an asynchronous clear is used to make a Modulo 7 counter
- Use the Load feature to detect the count "6" and load in "zero". This gives a count of 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, ...
- Using don't cares for states above 0110, detection of 6 can be done with Load = Q2 Q1



Counting Modulo 6: Synchronously Preset 9 on Reset and Load 9 on Terminal Count 14

- A synchronous, 4-bit binary counter with a synchronous Load is to be used to make a Modulo 6 counter.
- Use the Load feature to preset the count to 9 on Reset and detection of count 14.



- This gives a count of 9, 10, 11, 12, 13, 14, 9, 10, 11, 12, 13, 14, 9, …
- If the terminal count is 15 detection is usually built in as Carry Out (CO)

Counter Design Example

- Design a counter to repeat sequence of six states as in table

State Table and Flip-Flop Inputs for Counter

Present State			Next State		
A	B	C	DA = A(t+1)	DB = B(t+1)	DC= C(t+1)
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0

Counter Design Example

- Design a counter to repeat sequence of six states as in table

State Table and Flip-Flop Inputs for Counter

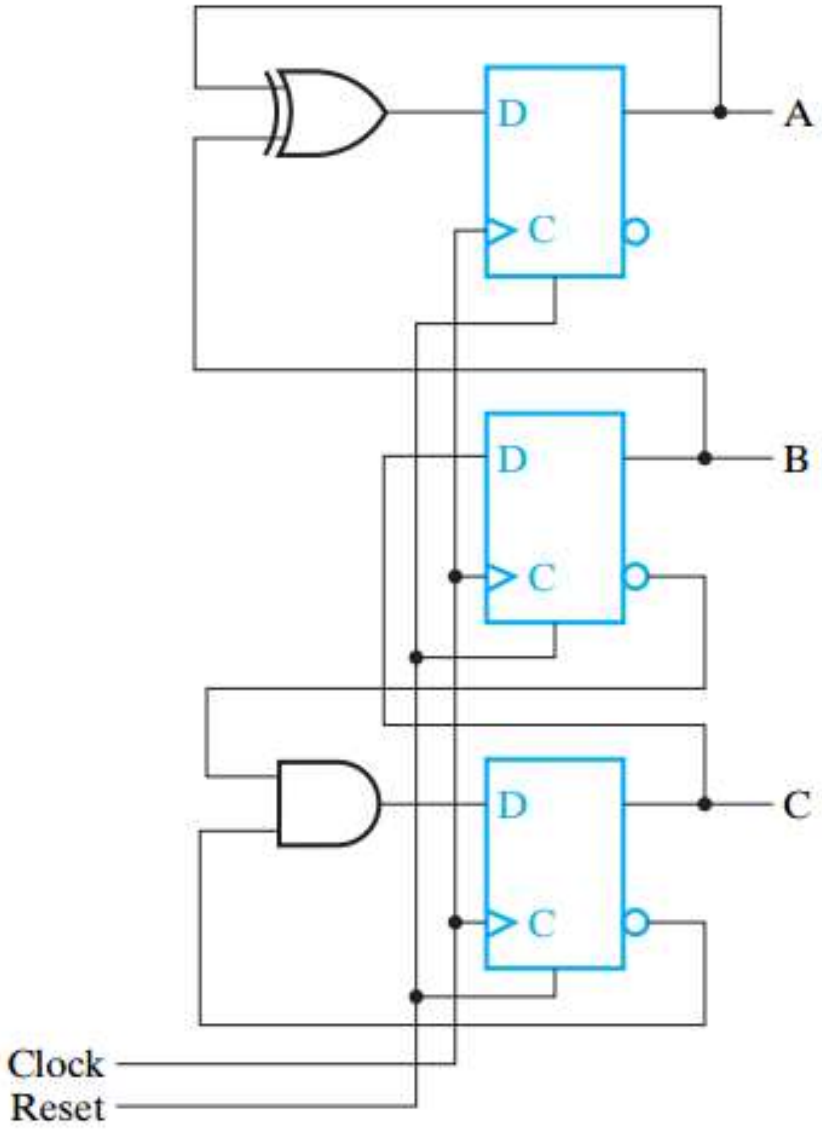
Present State			Next State		
A	B	C	DA = A(t+1)	DB = B(t+1)	DC= C(t+1)
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0

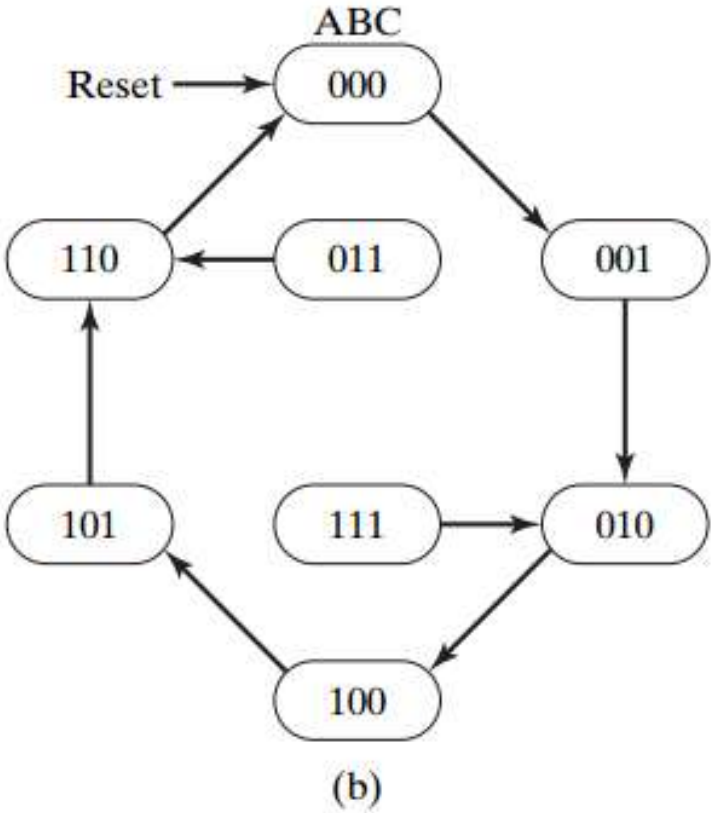
$DA = A \oplus B$

$DB = C$

$DC = \overline{B}\,\overline{C}$

Counter Design Example





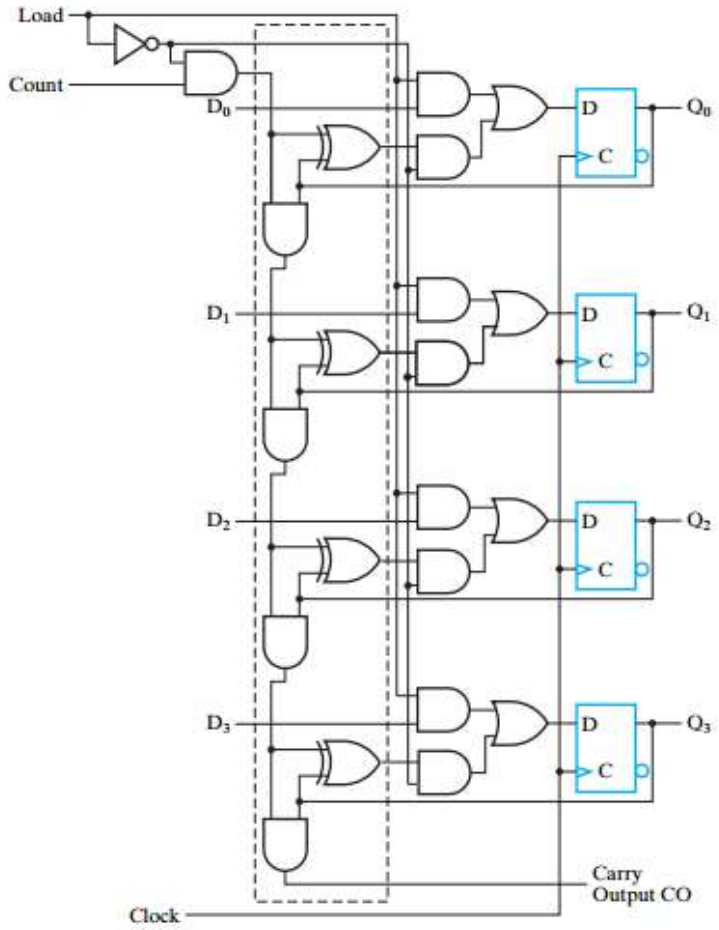
(b)

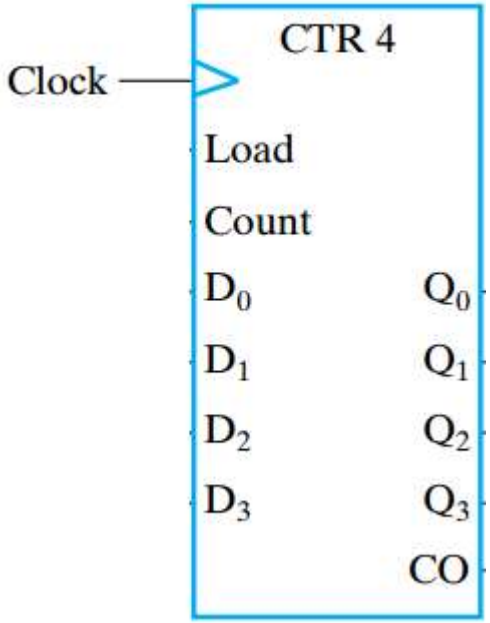
Counter Design Example 2

- Using D flip flops and gates design a binary counter to repeat sequence of states 1, 2, 6

Counter Design Example 3

- Using synchronous binary counter and an AND gate construct a counter that counts from 0000 through 0101





Any Questions?