Synchronous Counters, ripple counter & other counters Lecture 2

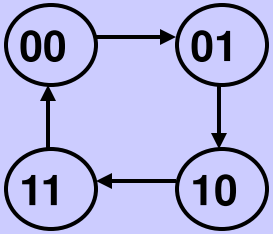
Dronacharya Group of Institutions

Introduction: Counters

- Counters are circuits that cycle through a specified number of states.
- Two types of counters:
  - synchronous (parallel) counters
  - asynchronous (ripple) counters
- Ripple counters allow some flip-flop outputs to be used as a source of clock for other flip-flops.
- Synchronous counters apply the same clock to all flip-flops.

Synchronous (Parallel) Counters

- Synchronous (parallel) counters: the flip-flops are clocked at the same time by a common clock pulse.
- We can design these counters using the sequential logic design process (covered in Lecture #12).
- Example: 2-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J,K inputs).



Present state		Next state		Flip-flop inputs	
A_1	A_0	A_1^+	A_0^+	TA_1	TA_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

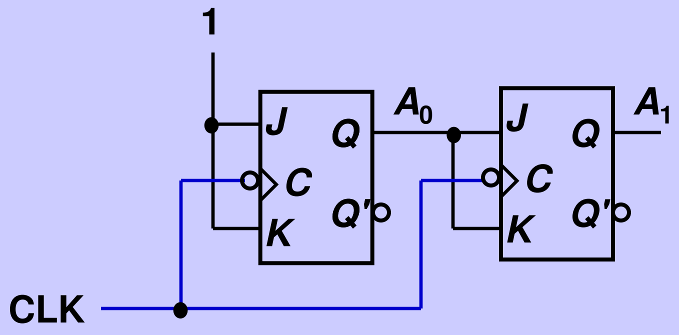
Synchronous (Parallel) Counters

- Example: 2-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J,K inputs).

Present state		Next state		Flip-flop inputs	
A_1	A_0	A_1^+	A_0^+	TA_1	TA_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

$TA_1 = A_0$

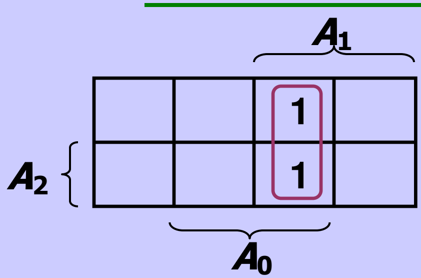
$TA_0 = 1$



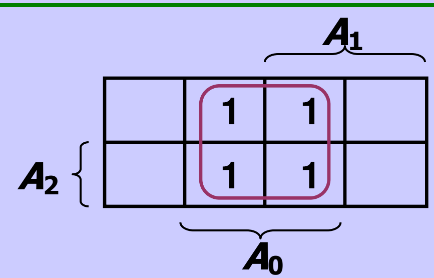
Synchronous (Parallel) Counters

- Example: 3-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J, K inputs).

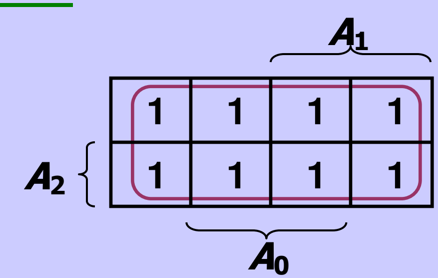
Present state			Next state			Flip-flop inputs		
A_2	A_1	A_0	A_2^+	A_1^+	A_0^+	TA_2	TA_1	TA_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1



$TA_2 = A_1.A_0$



$TA_1 = A_0$

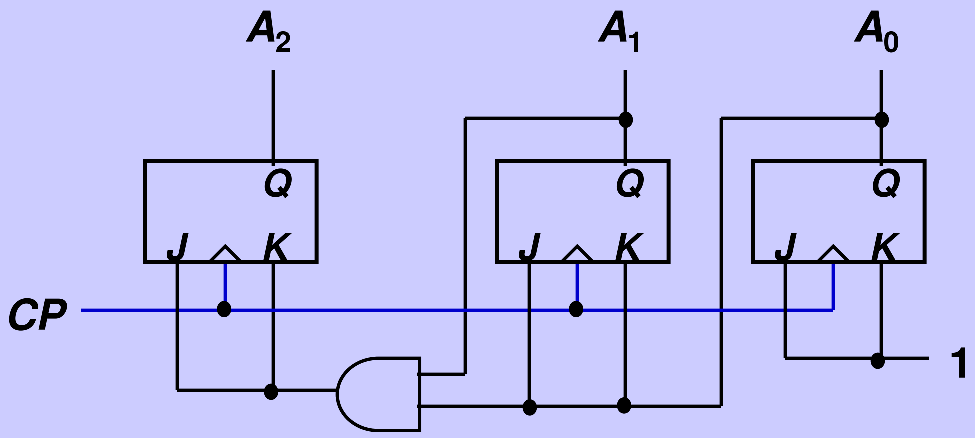


$TA_0 = 1$

Synchronous (Parallel) Counters

- Example: 3-bit synchronous binary counter (cont'd).

  $TA_2 = A_1.A_0$ $TA_1 = A_0$ $TA_0 = 1$



Synchronous (Parallel) Counters

- Note that in a binary counter, the n^{th} bit (shown underlined) is always complemented whenever

  $\underline{0}11\ldots11 \rightarrow \underline{1}00\ldots00$

  or $\underline{1}11\ldots11 \rightarrow \underline{0}00\ldots00$

- Hence, X_n is complemented whenever

  $X_{n-1}X_{n-2} \ldots X_1X_0 = 11\ldots11.$

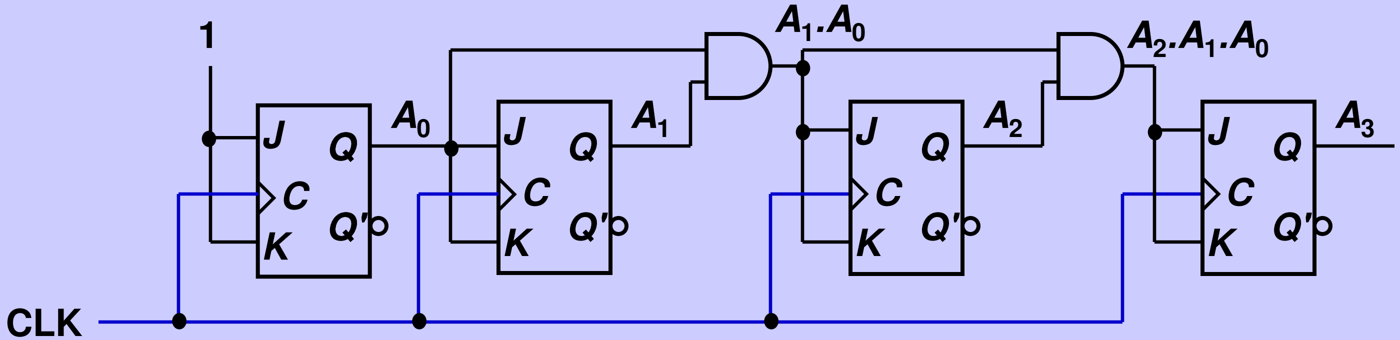
- As a result, if T flip-flops are used, then

  $TX_n = X_{n-1} \cdot X_{n-2} \cdot \ldots \cdot X_1 \cdot X_0$

Synchronous (Parallel) Counters

- Example: 4-bit synchronous binary counter.

$$TA_3 = A_2 \,.\, A_1 \,.\, A_0$$
$$TA_2 = A_1 \,.\, A_0$$
$$TA_1 = A_0$$
$$TA_0 = 1$$



Synchronous (Parallel) Counters

- Example: Synchronous decade/BCD counter.

Clock pulse	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10 (recycle)	0	0	0	0

$$T_0 = 1$$

$$T_1 = Q_3'.Q_0$$

$$T_2 = Q_1.Q_0$$

$$T_3 = Q_2.Q_1.Q_0 + Q_3.Q_0$$

Synchronous (Parallel) Counters

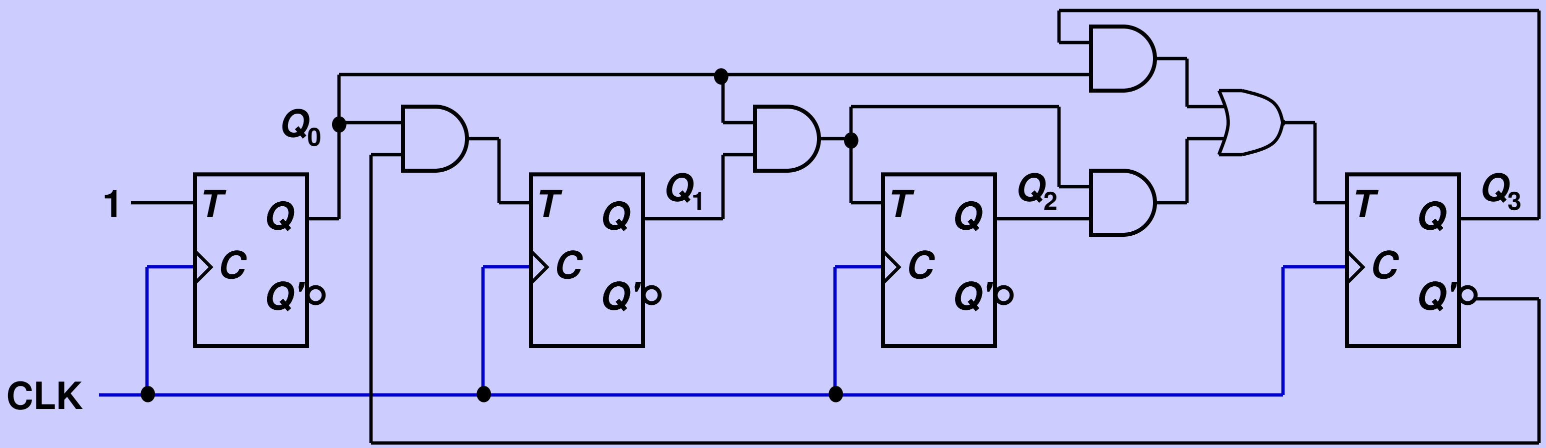
- Example: Synchronous decade/BCD counter (cont'd).

$T_0 = 1$

$T_1 = Q_3'.Q_0$

$T_2 = Q_1.Q_0$

$T_3 = Q_2.Q_1.Q_0 + Q_3.Q_0$



Up/Down Synchronous Counters

- Up/down synchronous counter: a *bidirectional* counter that is capable of counting either up or down.
- An input (control) line $Up/\overline{Down}$ (or simply Up) specifies the direction of counting.
  - $Up/\overline{Down} = 1 \rightarrow$ Count upward
  - $Up/\overline{Down} = 0 \rightarrow$ Count downward

Up/Down Synchronous Counters

- Example: A 3-bit up/down synchronous binary counter.

Clock pulse	Up	Q_2	Q_1	Q_0	Down
0		0	0	0	
1		0	0	1	
2		0	1	0	
3		0	1	1	
4		1	0	0	
5		1	0	1	
6		1	1	0	
7		1	1	1	

$$TQ_0 = 1$$

$$TQ_1 = (Q_0.Up) + (Q_0'.Up')$$

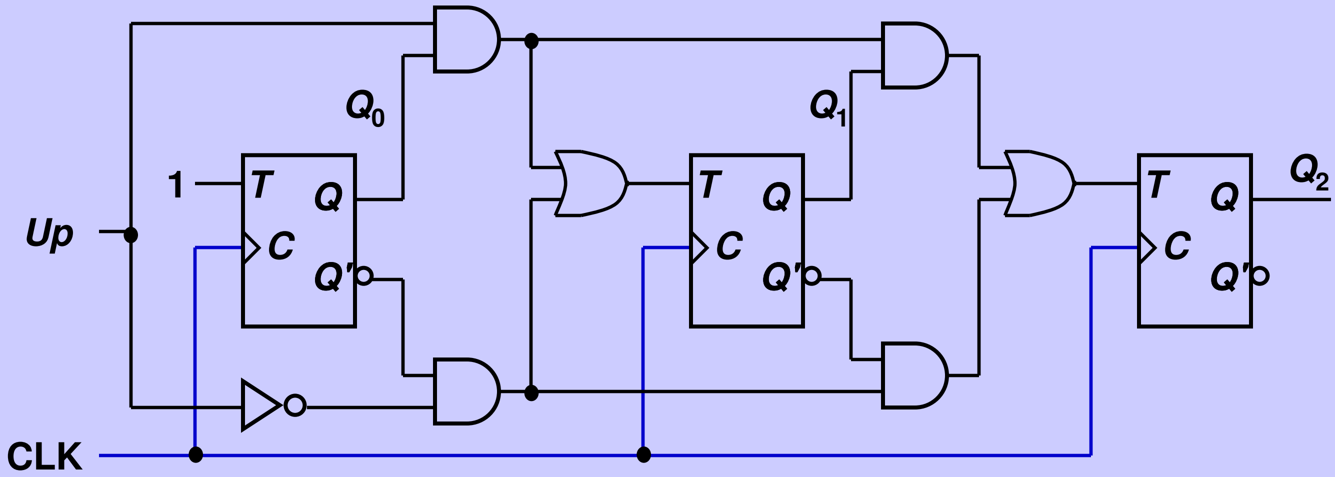
$$TQ_2 = (Q_0.Q_1.Up) + (Q_0'.Q_1'.Up')$$

Up counter	Down counter
$TQ_0 = 1$	$TQ_0 = 1$
$TQ_1 = Q_0$	$TQ_1 = Q_0'$
$TQ_2 = Q_0.Q_1$	$TQ_2 = Q_0'.Q_1'$

Up/Down Synchronous Counters

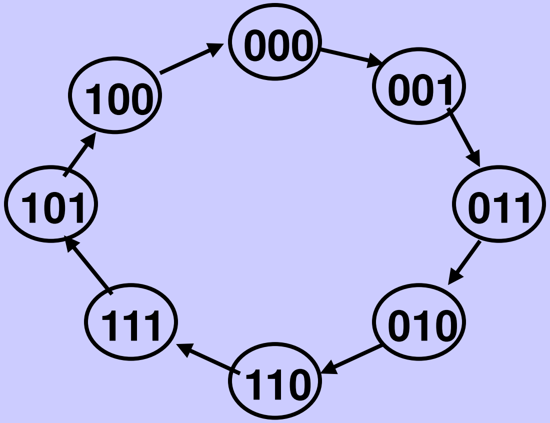
- Example: A 3-bit up/down synchronous binary counter (cont'd).

$$TQ_0 = 1$$
$$TQ_1 = (Q_0.Up) + (Q_0'.Up')$$
$$TQ_2 = (\ Q_0.Q_1.Up\ ) + (Q_0'.\ Q_1'.\ Up'\ )$$



Designing Synchronous Counters

- Covered in Lecture #12.
- Example: A 3-bit Gray code counter (using JK flip-flops).



Present state			Next state			Flip-flop inputs					
Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	JQ_2	KQ_2	JQ_1	KQ_1	JQ_0	KQ_0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	1	0	X	1	X	X	0
0	1	0	1	1	0	1	X	X	0	0	X
0	1	1	0	1	0	0	X	X	0	X	1
1	0	0	0	0	0	X	1	0	X	0	X
1	0	1	1	0	0	X	0	0	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	1	0	1	X	0	X	1	X	0

Designing Synchronous Counters

- 3-bit Gray code counter: flip-flop inputs.

$Q_2 \backslash Q_1Q_0$	00	01	11	10
0				1
1	X	X	X	X

$JQ_2 = Q_1.Q_0'$

$Q_2 \backslash Q_1Q_0$	00	01	11	10
0		1	X	X
1			X	X

$JQ_1 = Q_2'.Q_0$

$Q_2 \backslash Q_1Q_0$	00	01	11	10
0	1	X	X	
1		X	X	1

$JQ_0 = Q_2.Q_1 + Q_2'.Q_1'$
$= (Q_2 \oplus Q_1)'$

$Q_2 \backslash Q_1Q_0$	00	01	11	10
0	X	X	X	X
1	1			

$KQ_2 = Q_1'.Q_0'$

$Q_2 \backslash Q_1Q_0$	00	01	11	10
0	X	X		
1	X	X	1	

$KQ_1 = Q_2.Q_0$

$Q_2 \backslash Q_1Q_0$	00	01	11	10
0	X		1	X
1	X	1		X

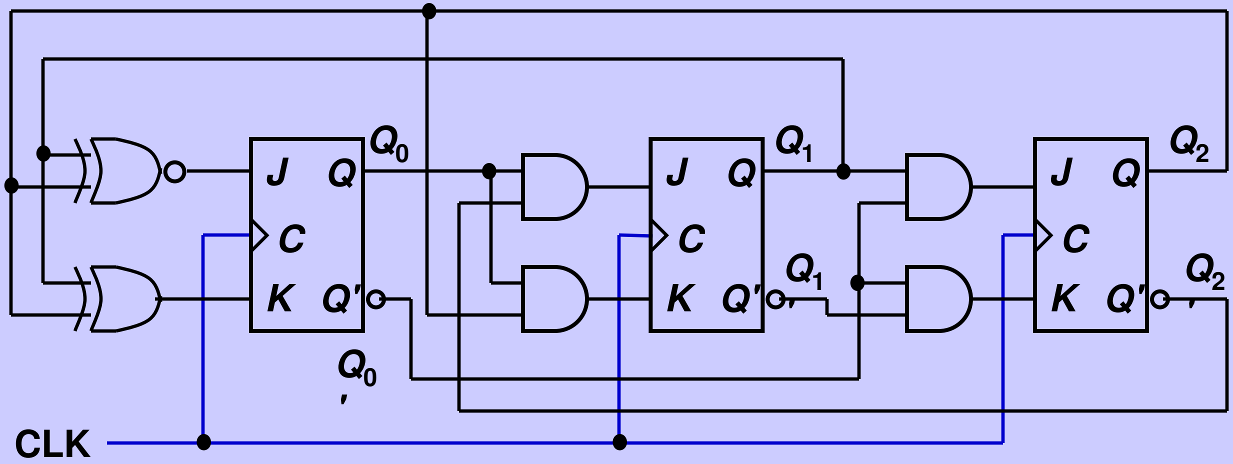
$KQ_0 = Q_2.Q_1' + Q_2'.Q_1$
$= Q_2 \oplus Q_1$

Designing Synchronous Counters

- 3-bit Gray code counter: logic diagram.

$JQ_2 = Q_1.Q_0'$ $\quad$ $JQ_1 = Q_2'.Q_0$ $\quad$ $JQ_0 = (Q_2 \oplus Q_1)'$

$KQ_2 = Q_1'.Q_0'$ $\quad$ $KQ_1 = Q_2.Q_0$ $\quad$ $KQ_0 = Q_2 \oplus Q_1$

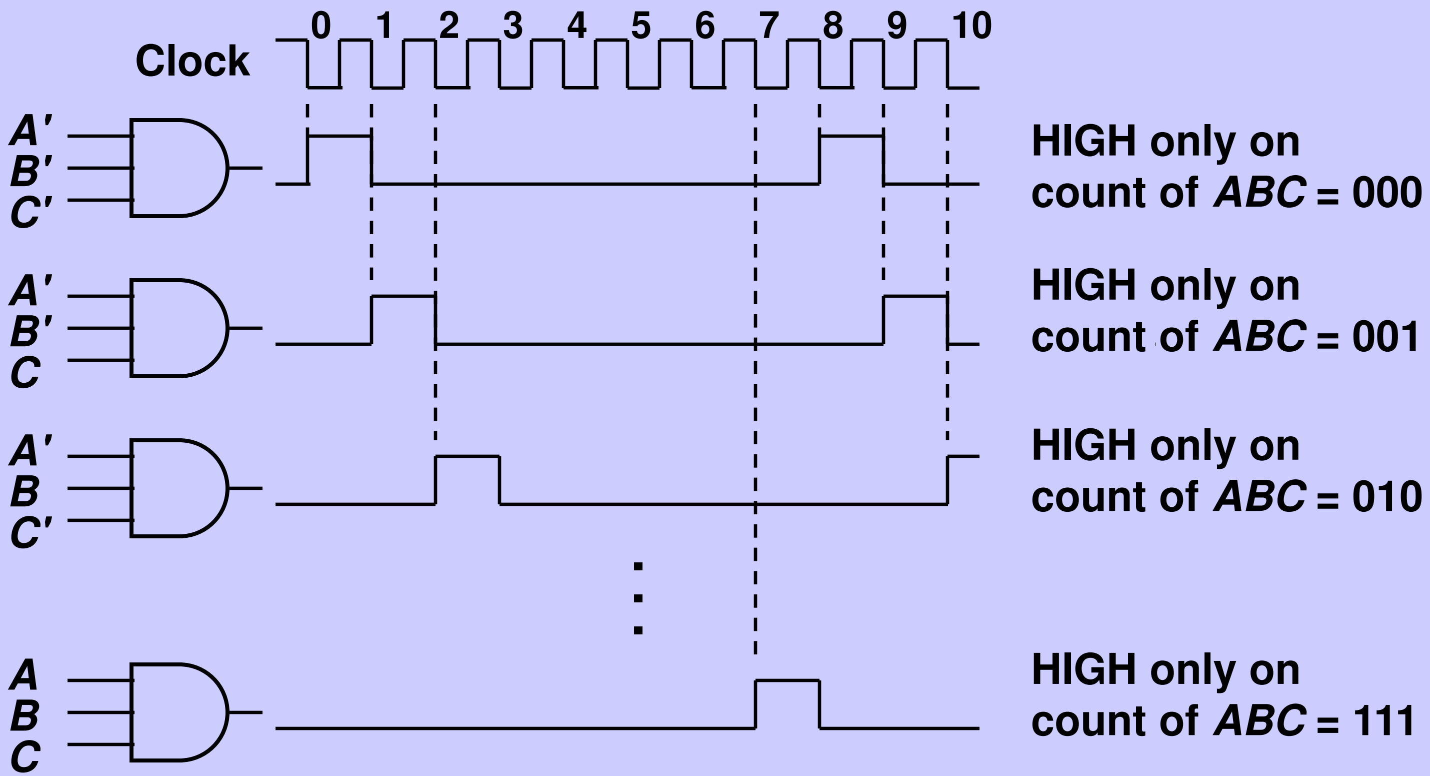


Decoding A Counter

- Decoding a counter involves determining which state in the sequence the counter is in.
- Differentiate between *active-HIGH* and *active-LOW* decoding.
- Active-HIGH decoding: output HIGH if the counter is in the state concerned.
- Active-LOW decoding: output LOW if the counter is in the state concerned.

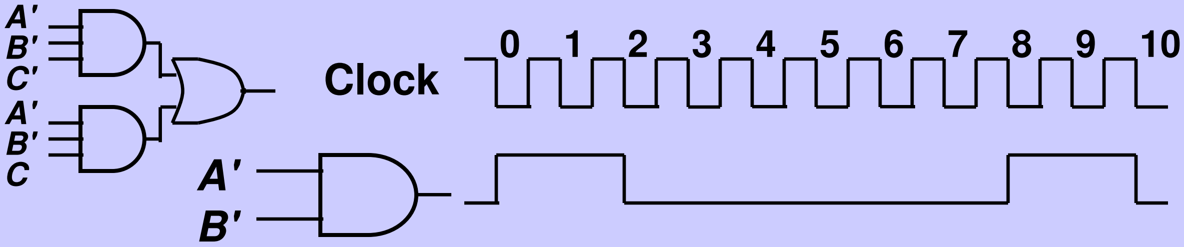
Decoding A Counter

- Example: MOD-8 ripple counter (active-HIGH decoding).



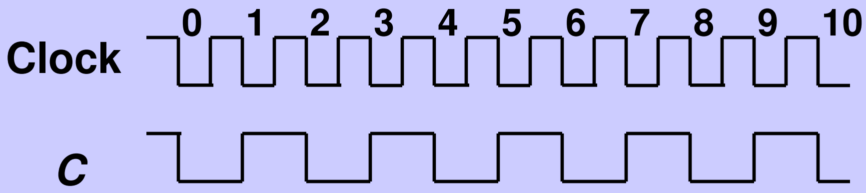
Decoding A Counter

- Example: To detect that a MOD-8 counter is in state 0 (000) or state 1 (001).



HIGH only on count of *ABC* = 000 or *ABC* = 001

- Example: To detect that a MOD-8 counter is in the odd states (states 1, 3, 5 or 7), simply use *C*.



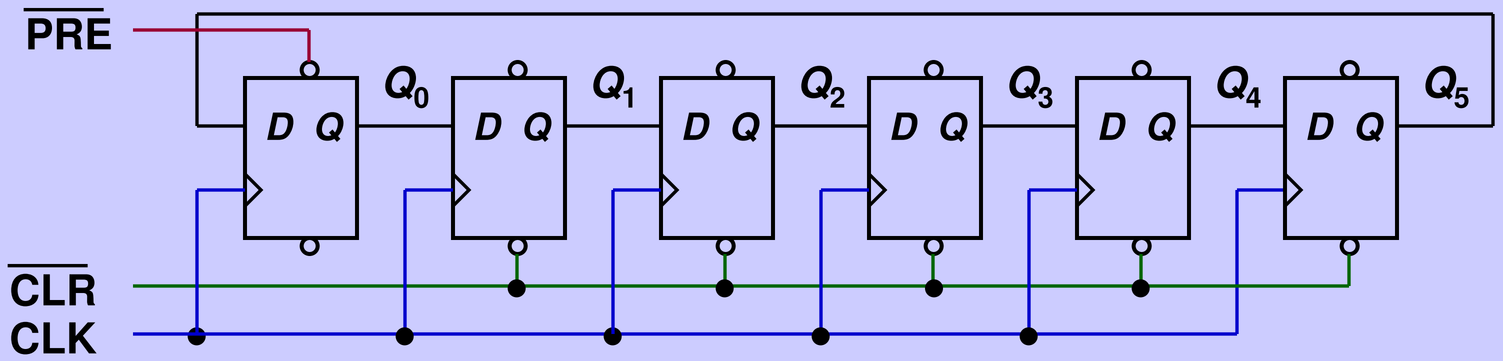
HIGH only on count of *odd states*

Ring Counters

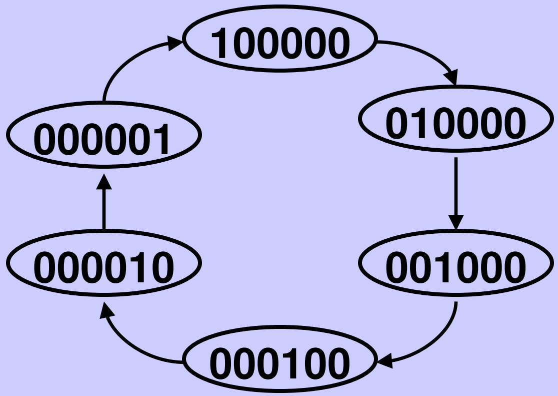
- One flip-flop (stage) for each state in the sequence.
- The output of the last stage is connected to the D input of the first stage.
- An *n*-bit ring counter cycles through *n* states.
- No decoding gates are required, as there is an output that corresponds to every state the counter is in.

Ring Counters

- Example: A 6-bit (MOD-6) ring counter.



Clock	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
0	1	0	0	0	0	0
1	0	1	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1

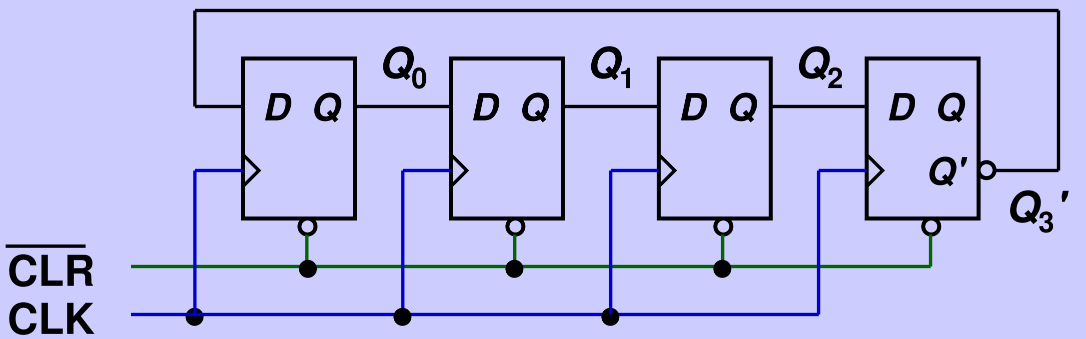


Johnson Counters

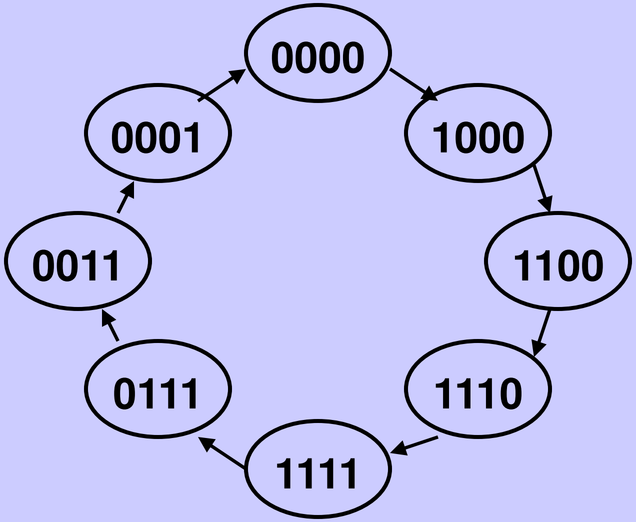
- The complement of the output of the last stage is connected back to the D input of the first stage.
- Also called the *twisted-ring counter*.
- Require fewer flip-flops than ring counters but more flip-flops than binary counters.
- An *n*-bit Johnson counter cycles through 2*n* states.
- Require more decoding circuitry than ring counter but less than binary counters.

Johnson Counters

- Example: A 4-bit (MOD-8) Johnson counter.



Clock	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1



Johnson Counters

- Decoding logic for a 4-bit Johnson counter.

Clock	*A*	*B*	*C*	*D*	Decoding
0	0	0	0	0	*A'.D'*
1	1	0	0	0	*A.B'*
2	1	1	0	0	*B.C'*
3	1	1	1	0	*C.D'*
4	1	1	1	1	*A.D*
5	0	1	1	1	*A'.B*
6	0	0	1	1	*B'.C*
7	0	0	0	1	*C'.D*