Indian Institute of Technology Jodhpur, Year 2018-2019

Digital Logic and Design

(Course Code: EE222)

Lecture 20-21: Counters contd….

Course Instructor: Shree Prakash Tiwari

Email: sptiwari@iitj.ac.in

Webpage: http://home.iitj.ac.in/~sptiwari/

Course related documents will be uploaded on http://home.iitj.ac.in/~sptiwari/DLD/

Note: The information provided in the slides are taken form text books Digital Electronics (including Mano & Ciletti), and various other resources from internet, for **teaching/academic use only**

Counters Overview

- **Counters are important components in computers**
  - **The increment or decrement by one in response to input**
- **Two main types of counters**
  - **Ripple (asynchronous) counters**
  - **Synchronous counters**
- **Ripple counters**
  - **Flip flop output serves as a source for triggering other flip flops**
- **Synchronous counters**
  - **All flip flops triggered by a clock signal**
- **Synchronous counters are more widely used in industry.**

Counters

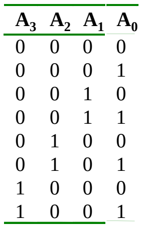
- Asynchronous counters: the flip-flops do not change states at exactly the same time as they do not have a common clock pulse.
- Known as ripple counters, as the input clock pulse "ripples" through the counter – cumulative delay is a drawback.
- *n* flip-flops → a MOD (modulus) 2^n counter. (Note: A MOD-*x* counter cycles through *x* states.)
- Output of the last flip-flop (MSB) divides the input clock frequency by the MOD number of the counter, hence a counter is also a *frequency divider*.

Counters

- Counter: A register that goes through a prescribed series of states
- Binary counter
  - Counter that follows a binary sequence
  - N bit binary counter counts in binary from n to 2^{n-1}
- Ripple counters triggered by initial Count signal
- Applications:
  - Watches
  - Clocks
  - Alarms
  - Web browser refresh

Asynchronous Counters

- Each FF output drives the CLK input of the next FF.
- FFs do not change states in exact synchronism with the applied clock pulses.
- *There is delay between the responses of successive FFs.*
- *Ripple counter* due to the way the FFs respond one after another in a kind of rippling effect.



A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
1	0	0	0
1	0	0	1

Binary Ripple Counters with T and D-FF

- Reset signal sets all outputs to 0
- Count signal toggles output of low-order flip flop
- Low-order flip flop provides trigger for adjacent flip flop
- Not all flops change value simultaneously
  - Lower-order flops change first
- Focus on D flip flop implementation

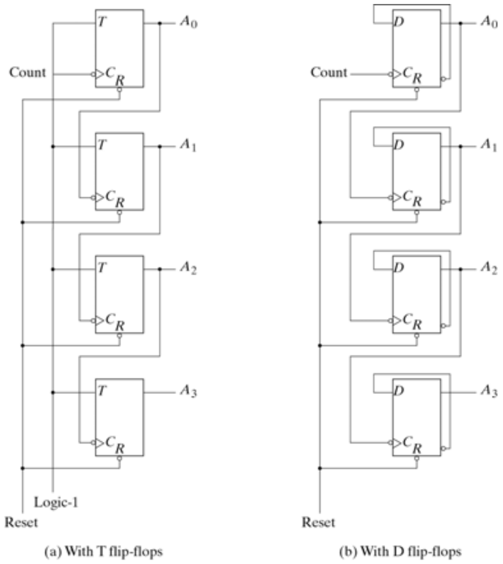
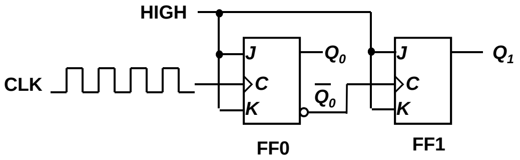
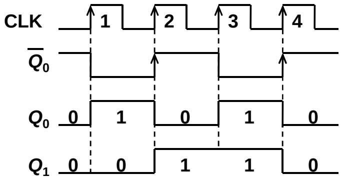


Fig. 6-8 4-Bit Binary Ripple Counter

Asynchronous Counters

- **Example: 2-bit ripple binary counter.**
- **Output of one flip-flop is connected to the clock input of the next more-significant flip-flop.**

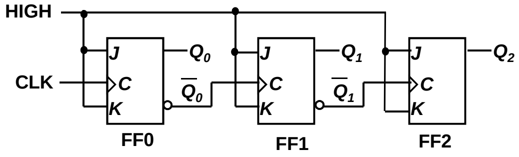


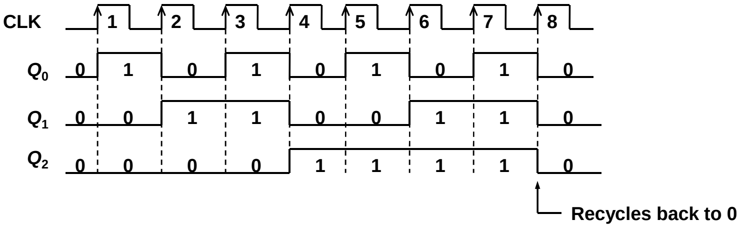


Timing diagram
00 → 01 → 10 → 11 → 00 ...

Asynchronous Counters

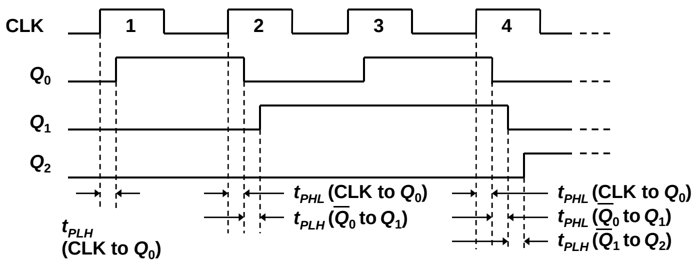
- **Example: 3-bit ripple binary counter.**





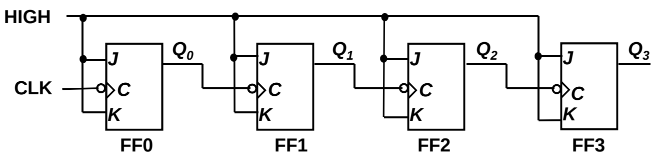
Asynchronous Counters

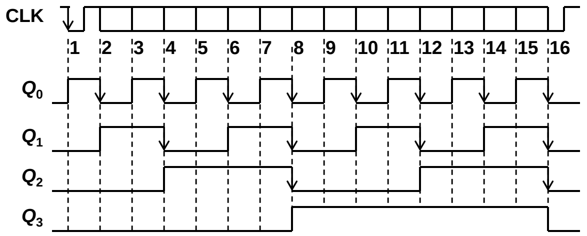
- Propagation delays in an asynchronous (ripple-clocked) binary counter.
- If the accumulated delay is greater than the clock pulse, some counter states may be misrepresented!



Asynchronous Counters

- Example: 4-bit ripple binary counter (negative-edge triggered).

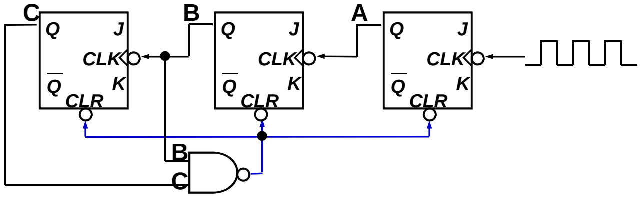




Asynchronous Counters with MOD no. < 2^n

- States may be skipped resulting in a truncated sequence.
- Technique: force counter to *recycle before going through all of the states* in the binary sequence.
- Example: Given the following circuit, determine the counting sequence (and hence the modulus no.)

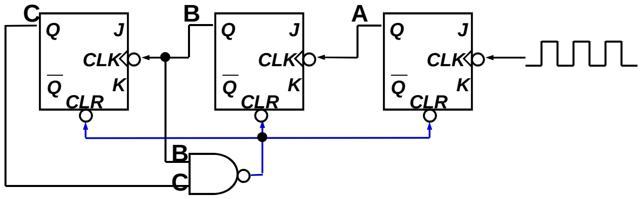
All *J*, *K* inputs are 1 (HIGH).

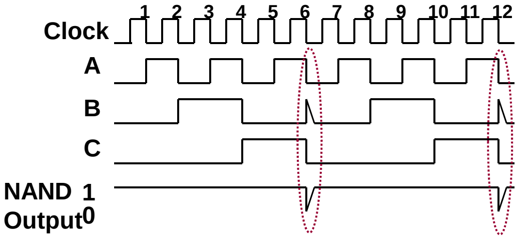


Asynchronous Counters with MOD no. < 2^n

- Example (cont'd):

All *J*, *K* inputs are 1 (HIGH).

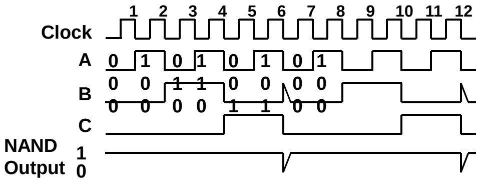


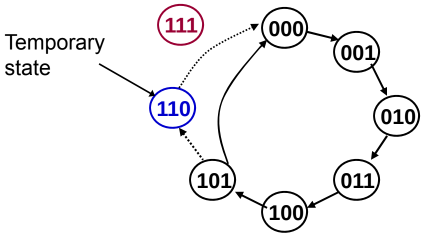


MOD-6 counter produced by clearing (a MOD-8 binary counter) when count of six (110) occurs.

Asynchronous Counters with MOD no. < 2^n

- Example (cont'd): Counting sequence of circuit (in CBA order).

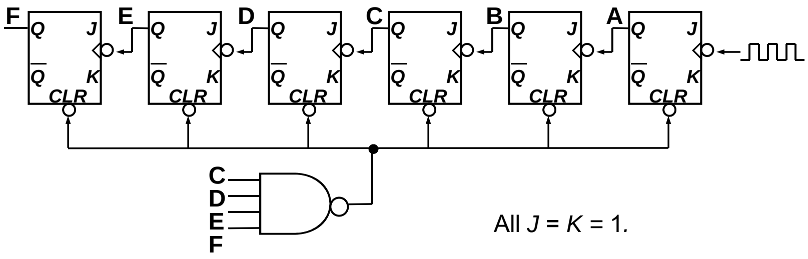




Counter is a MOD-6 counter.

Asynchronous Counters with MOD no. < 2^n

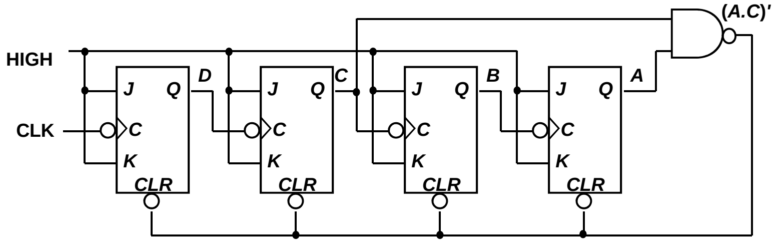
- *Exercise:* How to construct an asynchronous MOD-5 counter? MOD-7 counter? MOD-12 counter?
- *Question:* The following is a MOD-? counter?



All $J = K = 1$.

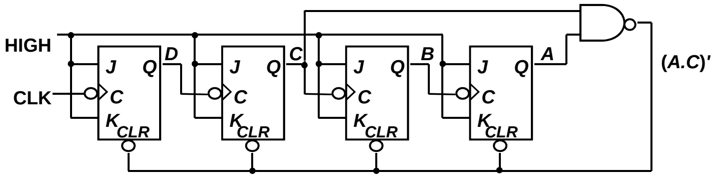
Asynchronous Counters with MOD no. < 2^n

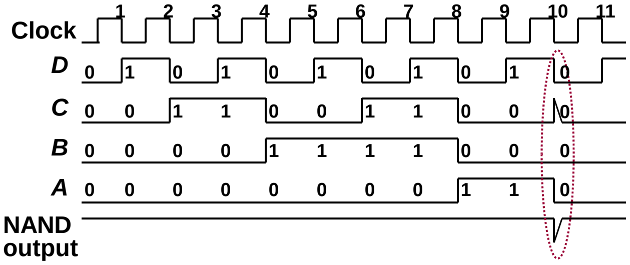
- **Decade counters (or BCD counters) are counters with 10 states (modulus-10) in their sequence. They are commonly used in daily life (e.g.: utility meters, odometers, etc.).**
- **Design an asynchronous decade counter.**



Asynchronous Counters with MOD no. < 2^n

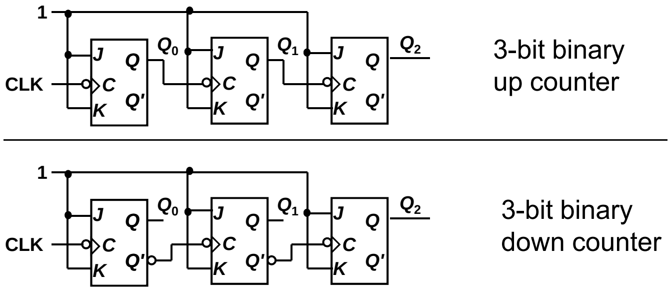
- **Asynchronous decade/BCD counter (cont'd).**





Asynchronous Down Counters

- So far we are dealing with *up counters*. ***Down counters***, on the other hand, count downward from a maximum value to zero, and repeat.
- Example: A 3-bit binary (MOD-2^3) down counter.

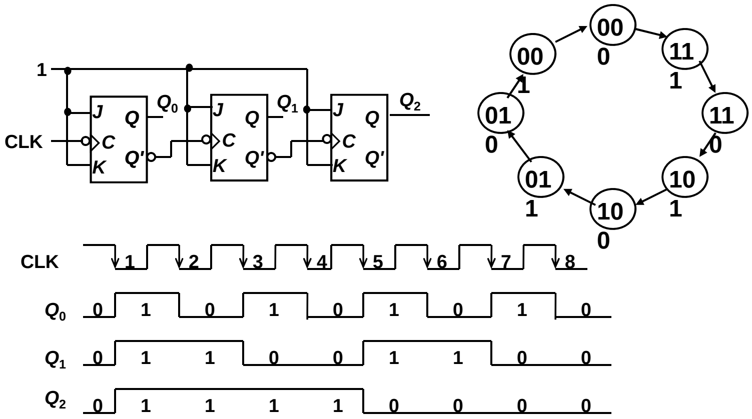


3-bit binary up counter

3-bit binary down counter

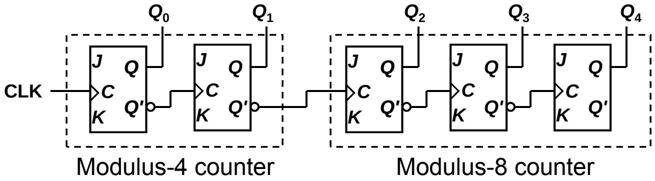
Asynchronous Down Counters

- Example: A 3-bit binary (MOD-8) down counter.



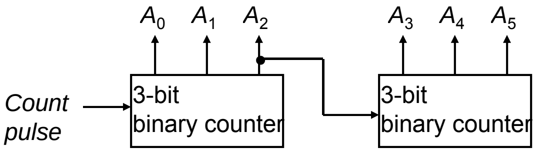
Cascading Asynchronous Counters

- Larger asynchronous (ripple) counter can be constructed by cascading smaller ripple counters.
- Connect last-stage output of one counter to the clock input of next counter so as to achieve higher-modulus operation.
- Example: A modulus-32 ripple counter constructed from a modulus-4 counter and a modulus-8 counter.



Cascading Asynchronous Counters

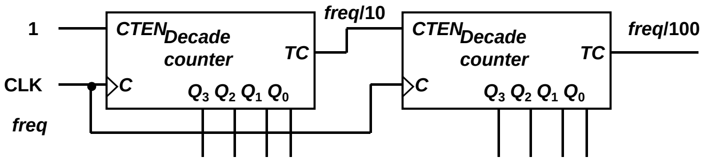
- Example: A 6-bit binary counter (counts from 0 to 63) constructed from two 3-bit counters.



A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	:	:	:
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	0	1
:	:	:	:	:	:

Cascading Asynchronous Counters

- If counter is a not a binary counter, requires additional output.
- Example: A modulus-100 counter using two decade counters.



TC = 1 when counter recycles to 0000

Synchronous counters

- Synchronous(parallel) counters
  - All of the FFs are triggered simultaneously by the clock input pulses.
  - All FFs change at same time
- Remember
  - If J=K=0, flop maintains value
  - If J=K=1, flop toggles
- Most counters are synchronous in computer systems.
- Can also be made from D flops
- Value increments on positive edge

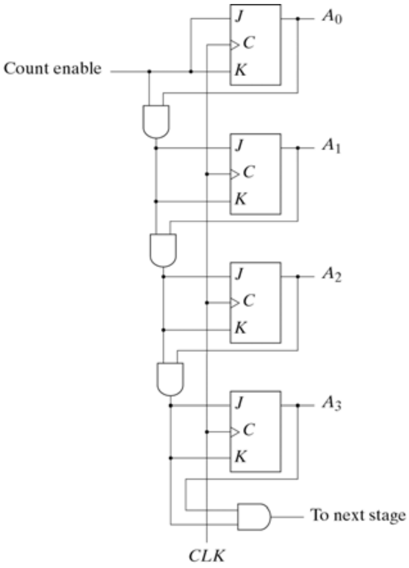
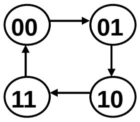


Fig. 6-12 4-Bit Synchronous Binary Counter

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 (will be covered in coming Lectures).**
- **Example: 2-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J,K inputs).**



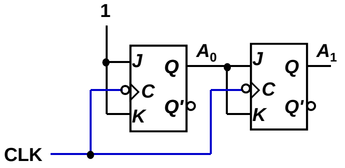
Present state		Next state		Flip-flop inputs	
A_1	A_0	A_1^+	A_0^+	TA_1	TA_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

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

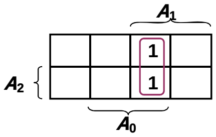
$TA_1 = A_0$
$TA_0 = 1$



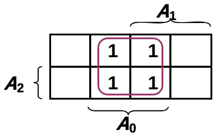
Synchronous (Parallel) Counters

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

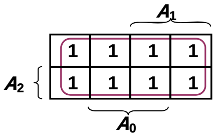
Present state			Next state			Flip-flop inputs		
A_2	A_1	A_0	A_2^+	A_1^+	A_0^+	TA_2	TA_1	TA_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1



$TA_2 = A_1.A_0$



$TA_1 = A_0$

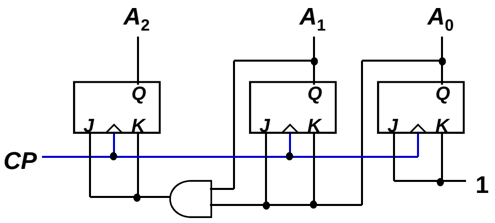


$TA_0 = 1$

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

Synchronous (Parallel) Counters

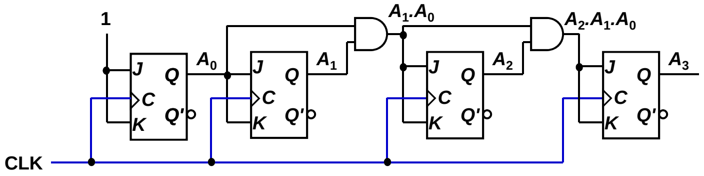
- Example: 4-bit synchronous binary counter.

  $TA_3 = A_2 . A_1 . A_0$

  $TA_2 = A_1 . A_0$

  $TA_1 = A_0$

  $TA_0 = 1$



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

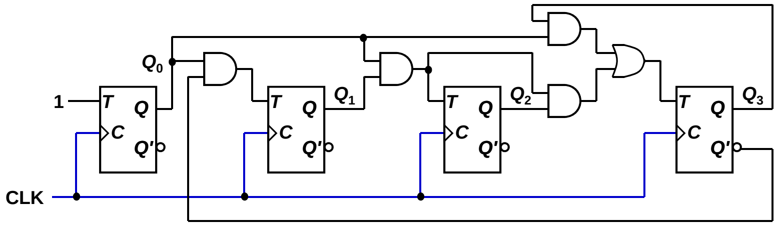
- **Example: Synchronous decade/BCD counter (cont'd).**

$T_0 = 1$

$T_1 = Q_3'.Q_0$

$T_2 = Q_1.Q_0$

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



Synchronous UP/Down counters

- Up/Down Counter can either count up or down on each clock cycle
- Up counter counts from 0000 to 1111 and then changes back to 0000
- Down counter counts from 1111 to 0000 and then back to 1111
- Counter counts up or down each clock cycle
- Output changes occur on clock rising edge

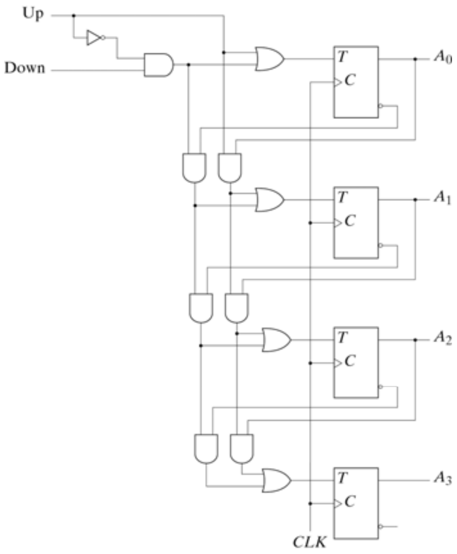


Fig. 6-13 4-Bit Up-Down Binary Counter

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*/*Down* (or simply *Up*) specifies the direction of counting.
  - ❖*Up*/*Down* = 1 → Count upward
  - ❖*Up*/*Down* = 0 → 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
$TQ_0 = 1$	$TQ_0 = 1$
$TQ_1 = Q_0$	$TQ_1 = Q_0'$
$TQ_2 = Q_0.Q_1$	$TQ_2 = Q_0'.Q_1'$

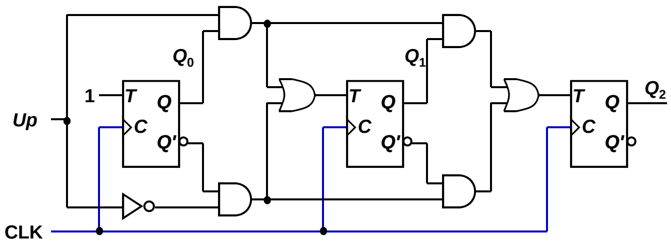
Up/Down Synchronous Counters

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

$TQ_0 = 1$

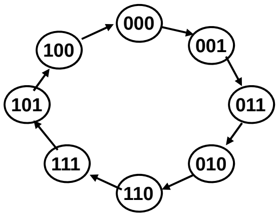
$TQ_1 = (Q_0.Up) + (Q_0'.Up')$

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



Designing

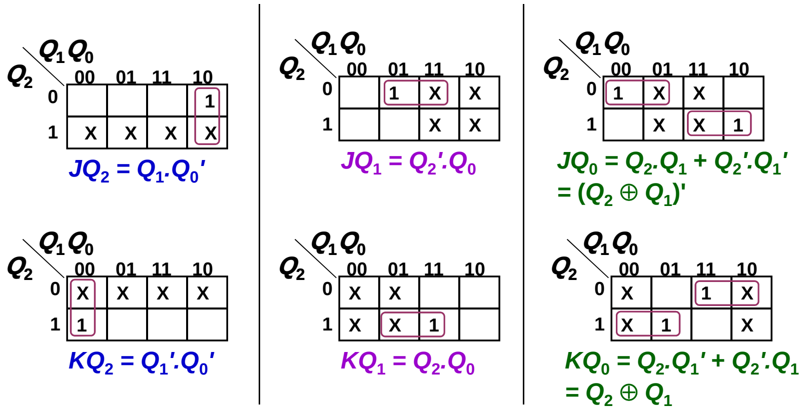
- As Covered Earlier
- Example: A 3-bit Gray code counter (using JK flip-flops).



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

Up/Down Synchronous Counters

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



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

$JQ_2 = Q_1 \cdot Q_0'$

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

$JQ_1 = Q_2' . Q_0$

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

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

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

$KQ_2 = Q_1' . Q_0'$

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

$KQ_1 = Q_2 \cdot Q_0$

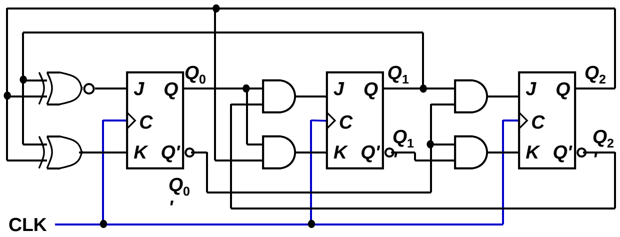
Q_2 \ Q_1Q_0	00	01	11	10
0	X		1	X
1	X	1		X

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

Up/Down Synchronous Counters

- 3-bit Gray code counter: logic diagram.

$JQ_2 = Q_1.Q_0'$ $JQ_1 = Q_2'.Q_0$ $JQ_0 = (Q_2 \oplus Q_1)'$
$KQ_2 = Q_1'.Q_0'$ $KQ_1 = Q_2.Q_0$ $KQ_0 = Q_2 \oplus Q_1$



Counters with Parallel Load

- Counters with parallel load can have a preset value
- Load signal indicates that data ($I_3 \ldots I_0$) should be loaded into the counter
- Clear resets counter to all zeros
- Carry output could be used for higher-order bits

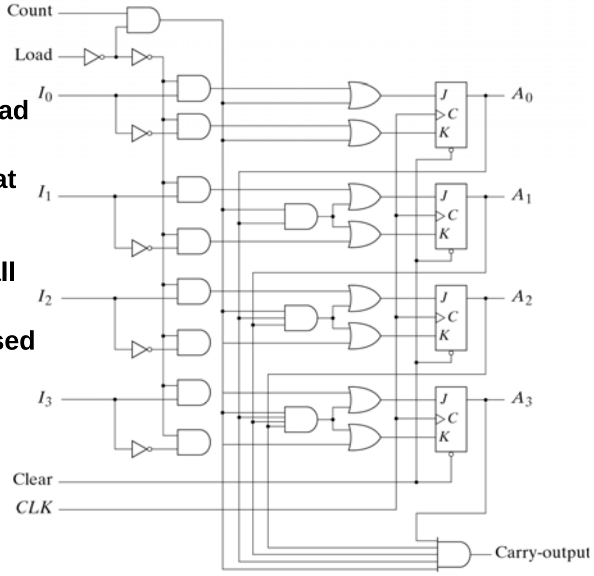


Fig. 6-14 4-Bit Binary Counter with Parallel Load

Counters with Parallel Load

Clear	Clk	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count
1	↑	0	0	No Change

Function Table

- **If Clear is asserted (0), the counter is cleared**
- **If Load is asserted data inputs are loaded**
- **If Count asserted counter value is incremented**

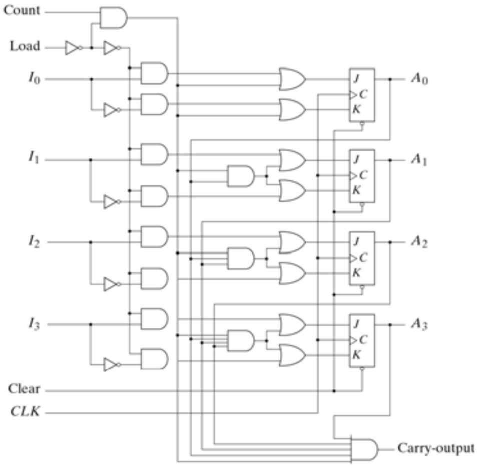
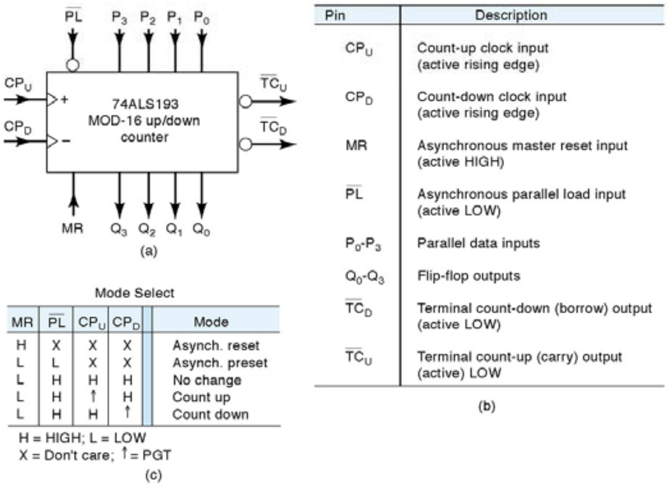


Fig. 6-14 4-Bit Binary Counter with Parallel Load

Binary Counter with Parallel Load and Preset

- **Commercial version of binary counter**



(a)

Pin	Description
CP_U	Count-up clock input (active rising edge)
CP_D	Count-down clock input (active rising edge)
MR	Asynchronous master reset input (active HIGH)
$\overline{PL}$	Asynchronous parallel load input (active LOW)
P_0-P_3	Parallel data inputs
Q_0-Q_3	Flip-flop outputs
$\overline{TC}_D$	Terminal count-down (borrow) output (active LOW)
$\overline{TC}_U$	Terminal count-up (carry) output (active) LOW

(b)

Mode Select

MR	$\overline{PL}$	CP_U	CP_D	Mode
H	X	X	X	Asynch. reset
L	L	X	X	Asynch. preset
L	H	H	H	No change
L	H	↑	H	Count up
L	H	H	↑	Count down

H = HIGH; L = LOW
X = Don't care; ↑ = PGT

(c)

Summary

- Binary counters can be ripple or synchronous
- Ripple counters use flip flop outputs as flop triggers
  - Some delay before all flops settle on a final value
  - Do no require a clock signal
- Synchronous counters are controlled by a clock
  - All flip flops change at the same time
- Up/Down counters can either increment or decrement a stored binary value
  - Control signal determines if counter counts up or down
- Counters with parallel load can be set to a known value before counting begins.