King Fahd University of Petroleum & Minerals
Computer Engineering Dept

COE 200 – Fundamentals of Computer Engineering

Term 043

Dr. Ashraf S. Hasan Mahmoud

Rm 22-148-3

Ext. 1724

Email: ashraf@ccse.kfupm.edu.sa

Counter

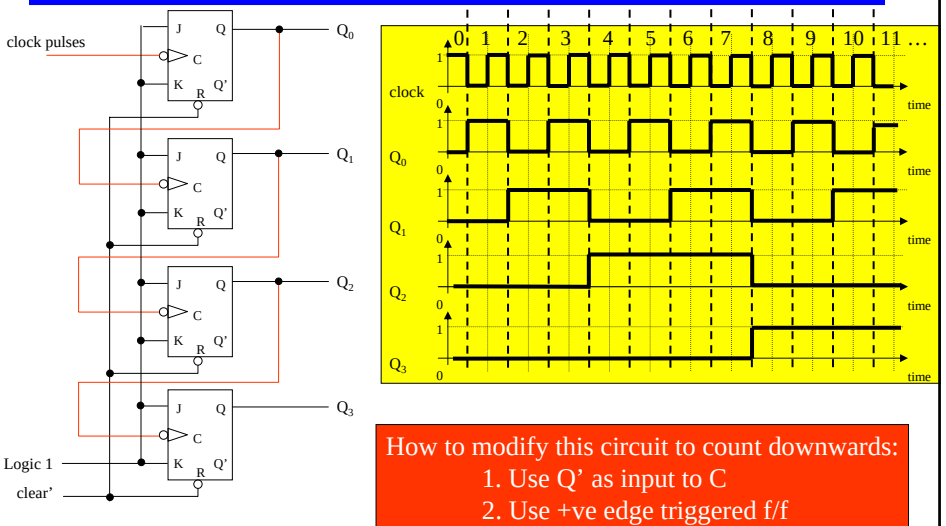
- **Counter**: A register (sequential circuit) that goes through a pre-determined sequence of states upon the application of input (clock or other source) pulses
- **Binary Counter**: The sequence of the states follows the binary number sequence (e.g. 000 → 001 → 010 → 011 → etc.)
  - n-bit binary counter requires n flip-flops – counts from 0 to 2^n-1
- Two Types of Counters:
  1. Ripple counter:
     - Flip-flop output transition serves as source for triggering the other flip-flops
  2. Synchronous counter: the clock input of all flip-flops receive the common clock signal

Binary Counter

Counting Sequence for Binary Number

No	Upward Counting Sequence				No	Downward Counting Sequence			
	Q3	Q2	Q1	Q0		Q3	Q2	Q1	Q0
0	0	0	0	0	15	1	1	1	1
1	0	0	0	1	14	1	1	1	0
2	0	0	1	0	13	1	1	0	1
3	0	0	1	1	12	1	1	0	0
4	0	1	0	0	11	1	0	1	1
5	0	1	0	1	10	1	0	1	0
6	0	1	1	0	9	1	0	0	1
7	0	1	1	1	8	1	0	0	0
8	1	0	0	0	7	0	1	1	1
9	1	0	0	1	6	0	1	1	0
10	1	0	1	0	5	0	1	0	1
11	1	0	1	1	4	0	1	0	0
12	1	1	0	0	3	0	0	1	1
13	1	1	0	1	2	0	0	1	0
14	1	1	1	0	1	0	0	0	1
15	1	1	1	1	0	0	0	0	0

4-bit Ripple Counter



How to modify this circuit to count downwards:
1. Use Q' as input to C
2. Use +ve edge triggered f/f

Ripple Counter

- Advantages:
  - Simple hardware
- Disadvantages:
  - Asynchronous – delay dependent
- Good for low power circuits

Synchronous Binary Counter

- Clock pulses are applied to the inputs of all of the flip-flops.
- Design of binary counters
  - Same as we learned in Chapter 4!

Example: 4-bit Binary Counter

State Table and Flip-Flops Inputs for Binary Counter

Present State				Next State				Flip-flips inputs							
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	J_{Q3}	K_{Q3}	J_{Q2}	K_{Q1}	J_{Q1}	K_{Q1}	J_{Q0}	K_{Q0}
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	1	0	1	0	X	0	0	X	1	X	X	1
1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	X
1	0	1	1	1	1	0	0	X	0	1	X	X	1	X	1
1	1	0	0	1	1	0	1	X	0	X	0	0	X	1	X
1	1	0	1	1	1	1	0	X	0	X	0	1	X	X	1
1	1	1	0	1	1	1	1	X	0	X	0	X	0	1	X
1	1	1	1	0	0	0	0	X	1	X	1	x	1	x	1

Example: 4-bit Binary Counter – cont'd

Q_1Q_0 / Q_3Q_2	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	x	x	x	x
10	x	x	x	x

$J_{Q3} = Q_0Q_1Q_2$

Q_1Q_0 / Q_3Q_2	00	01	11	10
00	0	0	1	0
01	x	x	x	x
11	x	x	x	x
10	0	0	1	0

$J_{Q2} = Q_0Q_1$

Q_1Q_0 / Q_3Q_2	00	01	11	10
00	0	1	x	x
01	0	1	x	x
11	0	1	x	x
10	0	1	x	x

$J_{Q1} = Q_0$

Q_1Q_0 / Q_3Q_2	00	01	11	10
00	x	x	x	x
01	x	x	x	x
11	0	0	1	0
10	0	0	0	0

$K_{Q3} = Q_0Q_1Q_2$

Q_1Q_0 / Q_3Q_2	00	01	11	10
00	x	x	x	x
01	0	0	1	0
11	0	0	1	0
10	x	x	x	x

$K_{Q2} = Q_0Q_1$

Q_1Q_0 / Q_3Q_2	00	01	11	10
00	x	x	1	0
01	x	x	1	0
11	x	x	1	0
10	x	x	1	0

$K_{Q1} = Q_0$

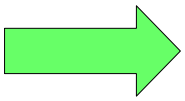
Example: 4-bit Binary Counter – Implementation

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	1	x	x	1
01	1	x	x	1
11	1	x	x	1
10	1	x	x	1

$J_{Q0} = 1$

Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	x	1	1	x
01	x	1	1	x
11	x	1	1	x
10	x	1	1	x

$K_{Q0} = 1$



$J_{Q3} = Q_0Q_1Q_2$

$K_{Q3} = Q_0Q_1Q_2$

$J_{Q2} = Q_0Q_1$

$K_{Q2} = Q_0Q_1$

$J_{Q1} = Q_0$

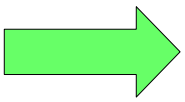
$K_{Q1} = Q_0$

$J_{Q0} = 1$

$K_{Q0} = 1$

Example: 4-bit Binary Counter – Implementation - continued

- Sometimes it is desirable to control the counter operation by added a count enable signal: En
  - When En = 1 ➔ counter counts
  - When En = 0 ➔ counter holds state (stops counting)
- Hence the new input equations are:



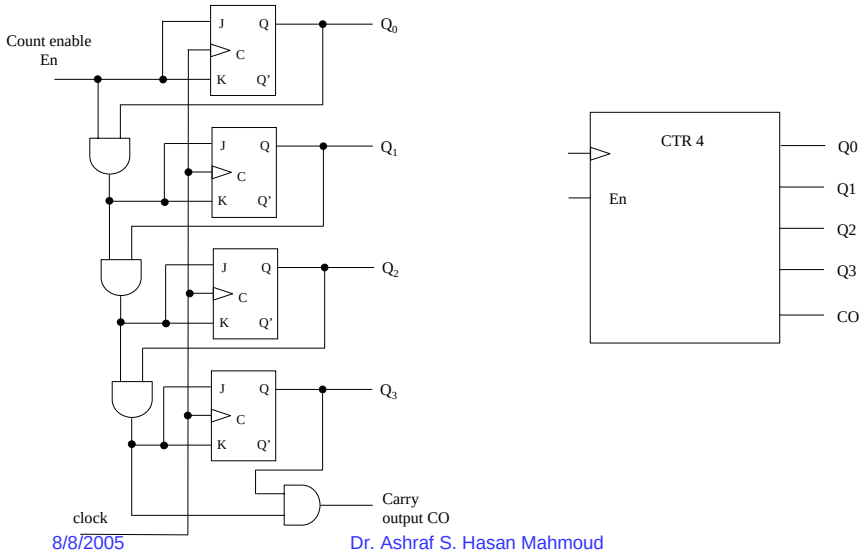
$J_{Q3} = Q_0Q_1Q_2En,\ K_{Q3} = Q_0Q_1Q_2En$

$J_{Q2} = Q_0Q_1En,\ K_{Q2} = Q_0Q_1En$

$J_{Q1} = Q_0En,\ K_{Q1} = Q_0En$

$J_{Q0} = En,\ K_{Q0} = En$

Example: 4-bit Binary Counter – Circuit



Notes on Binary Synchronous Counter

- When En = 0 ➔ J = K = 0 ➔ flip-flop holds state
- For general n-bit binary synchronous counter the input equation for the ith *JK* flip-flop is:

$$J_{Qi} = K_{Qi} = Q_0Q_1Q_2...Q_{i-1}Q_iEn$$

- The CO output can be used to extend the counter to more stages (How?)

☞ If we use a –ve edge triggered clock, will this circuit count downwards?
Draw the state diagram for this counter?

- Binary synchronous counters are MOST efficiently constructed with complementing *T* or *JK* flip-flops – but can be designed with *D* flip-flops

Problem 5-17: 4-bit Binary Counter with *D* flip-flops

Problem: Design a 4-bit binary synchronous counter with counter enable (En) signal

Solution:

State Table and Flip-Flop inputs for Binary Counter

Present State				Next State				Flip-flips inputs			
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	D_{Q3}	D_{Q2}	D_{Q1}	D_{Q0}
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	1	0	0	1	1
0	0	1	1	0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	1	0	1	1	1
0	1	1	1	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1	1	0	0	1
1	0	0	1	1	0	1	0	1	0	1	0
1	0	1	0	1	0	1	1	1	0	1	1
1	0	1	1	1	1	0	0	1	1	0	0
1	1	0	0	1	1	0	1	1	1	0	1
1	1	0	1	1	1	1	0	1	1	1	0
1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0	0

Problem 5-17: 4-bit Binary Counter with *D* flip-flops – cont'd

- The flip-flops input equations are given by:

Q_1Q_0 / Q_3Q_2	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

$D_{Q0} = Q_0' = Q_0 \oplus 1$

Q_1Q_0 / Q_3Q_2	00	01	11	10
00	0	0	1	0
01	1	1	0	1
11	1	1	0	1
10	0	0	1	0

$D_{Q2} = Q_0Q_1Q_2' + Q_1'Q_2 + Q_0'Q_2 = Q_2 \oplus (Q_0Q_1)$

Q_1Q_0 / Q_3Q_2	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$D_{Q1} = Q_0Q_1' + Q_0'Q_1 = Q_1 \oplus Q_0$

Q_1Q_0 / Q_3Q_2	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	1	1	0	1
10	1	1	1	1

$D_{Q3} = Q_0Q_1Q_2Q_3' + Q_2'Q_3 + Q_1'Q_3 + Q_0'Q_3$
$= Q_3 \oplus (Q_2Q_1Q_0)$

Problem 5-17: 4-bit Binary Counter with *D* flip-flops – cont'd

- Rewriting the input equations and using the count enable signal En:

$D_{Q0} = Q_0' = Q_0 \oplus 1$

$D_{Q1} = Q_0Q_1'+Q_0'Q_1 = Q_1 \oplus Q_0$

$D_{Q2} = Q_0Q_1Q_2'+Q_1'Q_2+Q_0'Q_2 = Q_2 \oplus (Q_0Q_1)$

$D_{Q3} = Q_0Q_1Q_2Q_3'+Q_2'Q_3+Q_1'Q_3+Q_0'Q_3$
$= Q_3 \oplus (Q_2Q_1Q_0)$

Note: when En = 0, the Q_i is input to D_{Qi} ➔ i.e flip-flop holds state

$D_{Q0} = Q_0 \oplus En = Q0'$ if En = 1
$= Q0$ if En = 0

$D_{Q1} = Q_1 \oplus (Q_0En) = Q_0Q_1'+Q_0'Q_1$ if En = 1
$= Q1$ if En = 0

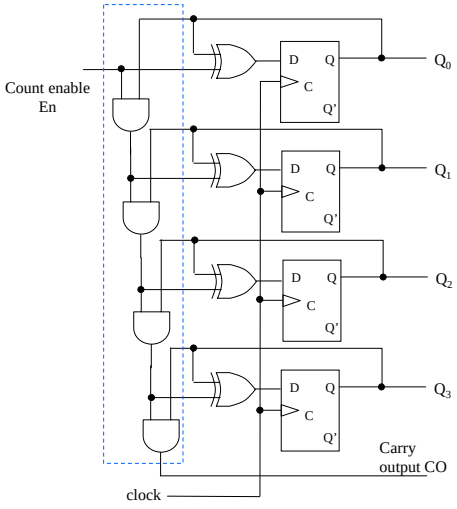
$D_{Q2} = Q_2 \oplus (Q_0Q_1En) = Q_0Q_1Q_2'+Q_1'Q_2+Q_0'Q_2$ if En = 1
$= Q_2$ if En = 0

$D_{Q3} = Q_3 \oplus (Q_2Q_1Q_0En) = Q_0Q_1Q_2Q_3'+Q_2'Q_3+Q_1'Q_3+Q_0'Q_3$ if En = 1
$= Q_3$ if En = 0

Problem 5-17: 4-bit Binary Counter with *D* flip-flops – Circuit Diagram

- Notes:
  - For a general n-bit counter, the i^{th} *D* flip-flop input equation is given by:

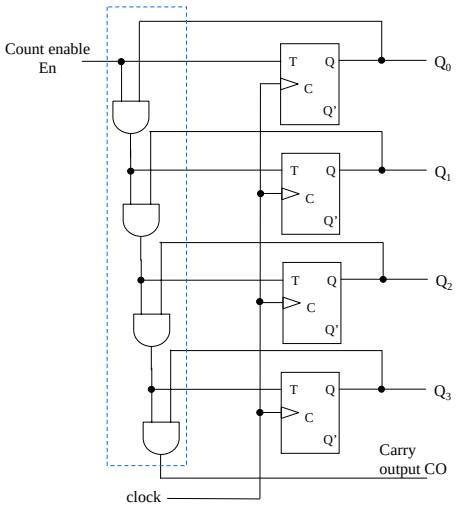
$D_{Qi} = Q_i \oplus (Q_0Q_1...Q_{i-1}En)$



Exercise: 4-bit Binary Counter with *T* flip-flops

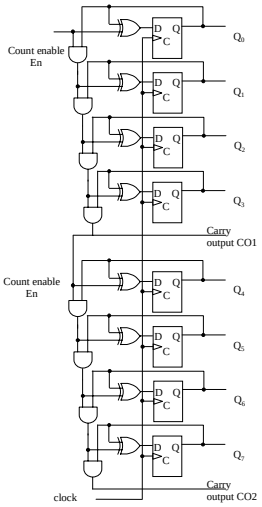
- **<u>Problem</u>: Verify that the circuit shown in Figure is a 4-bit synchronous binary counter**

(Hint: Draw the timing diagram)



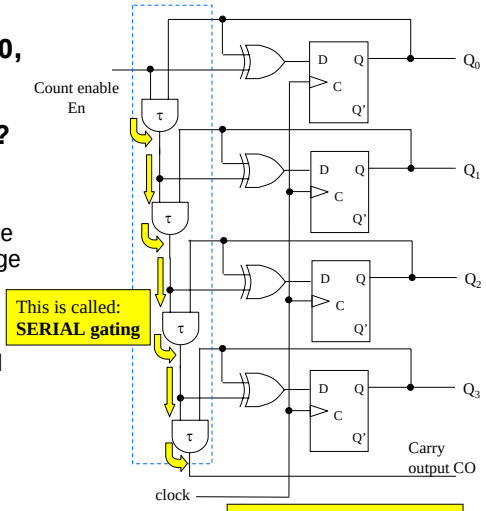
What is the Use of the CO signal? Design of 8-bit Binary Counter

- Using 4-bit counters with carry out signal one can construct larger counters by using the carry out signal from one stage as the enable signal of the next
- Figure shows 2 4-bit binary counters connected in series to form an 8-bit binary counter
- **<u>Exercise</u>**: Draw the timing diagram showing clock, Q0, Q1, Q2, Q3, CO1, Q4, Q5, Q6, Q7, and CO2



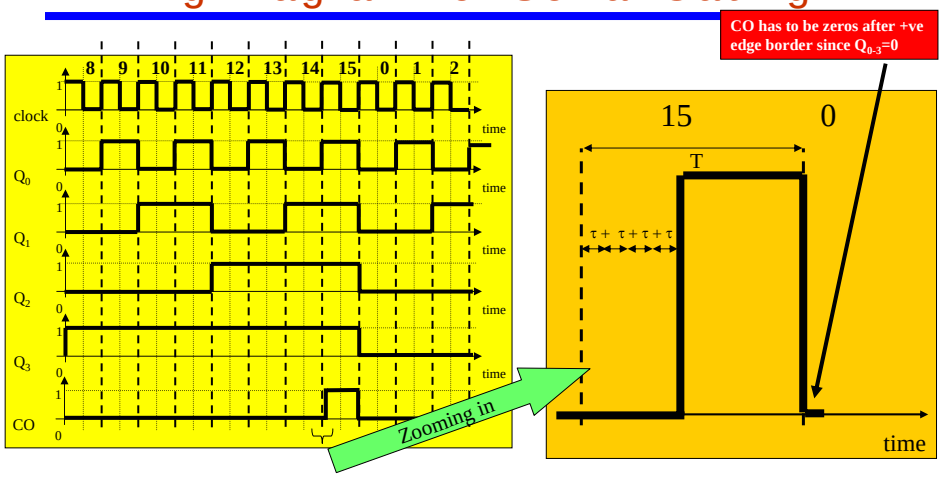
Serial and Parallel Counters

- **When counter changes state from 1111 to 0000, how many AND gate delays there will be before CO is generated?**
- **Answer**: FOUR. Because stage 1 requires one AND gate delay to know the state change in Q0; stage 2 requires one additional AND gate delay to know the state change in Q1; stage 3 requires on additional AND gate delay to know the state change in Q2; and to generate CO, one more AND gate delay is required.



SERIAL counter

Timing Diagram for Serial Gating



☞ **What happens if 4τ is greater than T?**

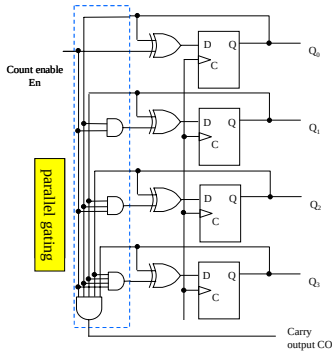
Why Serial Gating is Bad?

- Delay taken when CO signal is generated – if this CO signal is used to drive another counter (refer to 8-bit counter construction) ➔ cumulative delay may exceed clock pulse (or cumulative delay limits speed of counting – clock frequency)
- Not good for design of fast counters
- **Solution**: Use parallel gating!

This problem is similar to the delay problem we faced with the ripple carry adder. The parallel gating is analogous to the carry lookahead adder idea where the required signals are Generated without propagating the f/f status from Q0 to Q1 to Q2 to Q3

Parallel Gating

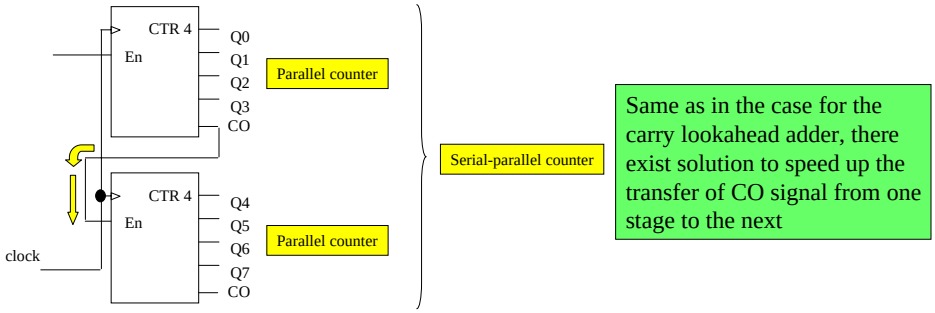
- For the D flip-flop synchronous binary counter, the equations for parallel gating can be taken directly from the AND terms in the flip-flop input equations on slide 15



parallel counter: ONE AND gate delay (τ) is required to generate CO

Serial-Parallel Counter

- Each stage (4-bit counter) is constructed using parallel gating
- Stages are connected in series by using the CO output of one stage as the counter enable input of the next state



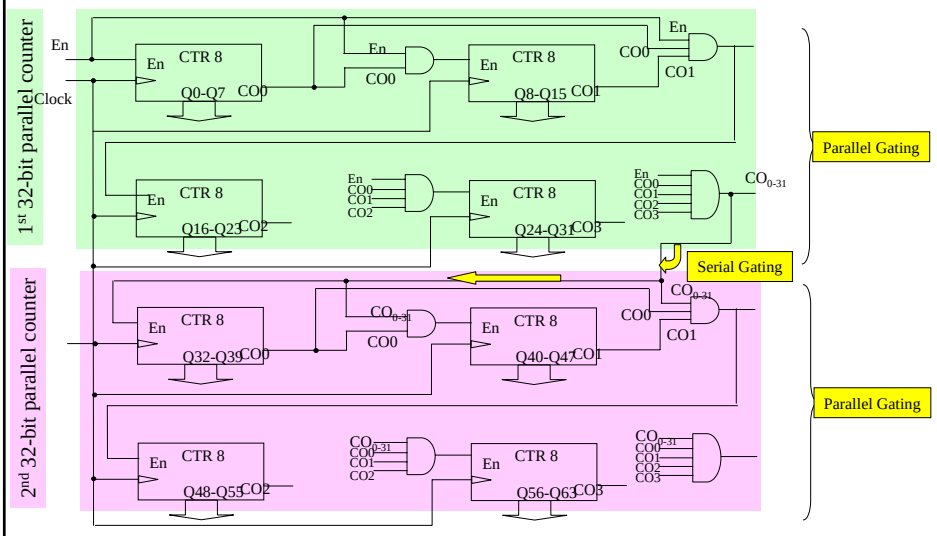
Same as in the case for the carry lookahead adder, there exist solution to speed up the transfer of CO signal from one stage to the next

Problem 5-20: Parallel Counter Design – Maximum Frequency

Problem: A 64-bit synchronous parallel counter is to be designed

(a) Draw the logic diagram of a 64-bit parallel counter using 8-bit parallel counter blocks and two levels of parallel gating connections between the blocks. In these blocks CO is not driven by En

Problem 5-20: Solution



Problem 5-20: Solution – cont'd

- Notes:
  - Each of the CTR 8 blocks is a parallel counter (i.e. it uses parallel gating)
  - The 32-bit counter resulting from connecting the first (or second) 4 8-bit block counters is a also a parallel counter since it uses parallel gating to generate the CO signals
  - The 2 32-bit counters are serially connected to form the required 64-bit counter
  - Would it be possible to use only parallel gating to obtain the 64-bit counter? What would be the maximum fan in for the used AND gates in this case?

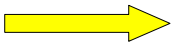
Synchronous Binary Down-Counter

- **Problem**: Design a 4-bit synchronous *T* flip-flop down-counter.
- Solution: (show that the previous circuits work as a down-counter if we use Q' instead of Q in the input equations; i.e the as input to the AND gates – This applies to the *JK* flip-flop counter on slide 11, the *D* flip-flop counter on slide 16 and the *T* flip-flop counter on slide 17)

Refer to problem 4-33 solved in the previous package – same problem but using JK F/Fs

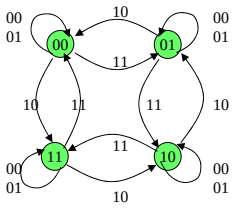
Up-Down Binary Counter

- **Problem**: Design a synchronous up-down *T* flip-flop 2-bit binary counter with a select input line S and a count enable En input. When S = 0, the counter counts down; and when S = 1, the counter counts up. When En = 1, the counter is in normal up- or down- counting; and En = 0 for disabling both counts.
- **Solution**: Required mode of operation:

Functional Table: 

Inputs		Operation
En	S	
0	x	Hold status
1	0	Count Down
1	1	Count Up

State Diagram/Table for 2-bit Up-Down Binary Counter



Arc Label: EnS

Present State			Inputs		Next State			T flip-flops	
No	Q1	Q0	En	S	**No**	Q1	Q0	T_{Q1}	T_{Q0}
0	0	0	0	0	**0**	0	0	0	0
0	0	0	0	1	**0**	0	0	0	0
0	0	0	1	0	**3**	1	1	1	1
0	0	0	1	1	**1**	0	1	0	1
1	0	1	0	0	**1**	0	1	0	0
1	0	1	0	1	**1**	0	1	0	0
1	0	1	1	0	**0**	0	0	0	1
1	0	1	1	1	**2**	1	0	1	1
2	1	0	0	0	**2**	1	0	0	0
2	1	0	0	1	**2**	1	0	0	0
2	1	0	1	0	**1**	0	1	1	1
2	1	0	1	1	**3**	1	1	0	1
3	1	1	0	0	**3**	1	1	0	0
3	1	1	0	1	**3**	1	1	0	0
3	1	1	1	0	**2**	1	0	0	1
3	1	1	1	1	**0**	0	0	1	1

T Flip-Flop

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Input Equations for 2-bit Up-Down Binary Counter

EnS / Q_1Q_0	00	01	11	10
00	0	0	0	1
01	0	0	1	0
11	0	0	1	0
10	0	0	0	1

$T_{Q1} = Q_0EnS + Q_0'EnS'$

EnS / Q_1Q_0	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

$T_{Q0} = En$

- The carry out signals:
  - CO_{up} and CO_{down}

$CO_{up} = Q_0Q_1EnS$ ➔ counter reached 11 and it is counting up

$CO_{down} = Q_0'Q_1'EnS'$ ➔ counter reached 00 and it is counting down

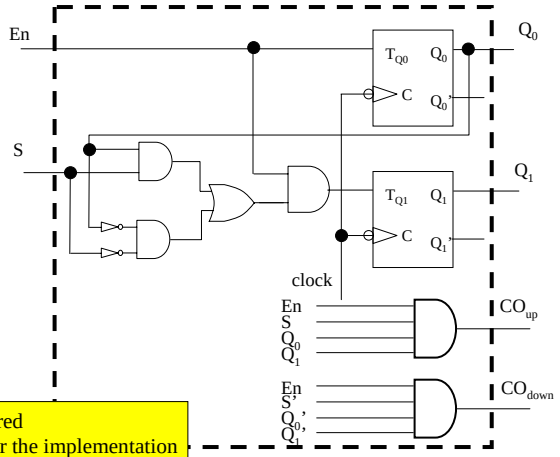
Circuit for 2-bit Up-Down Binary Counter

$T_{Q1} = Q_0EnS + Q_0'EnS'$

$T_{Q0} = En$

$CO_{up} = Q_0Q_1EnS$

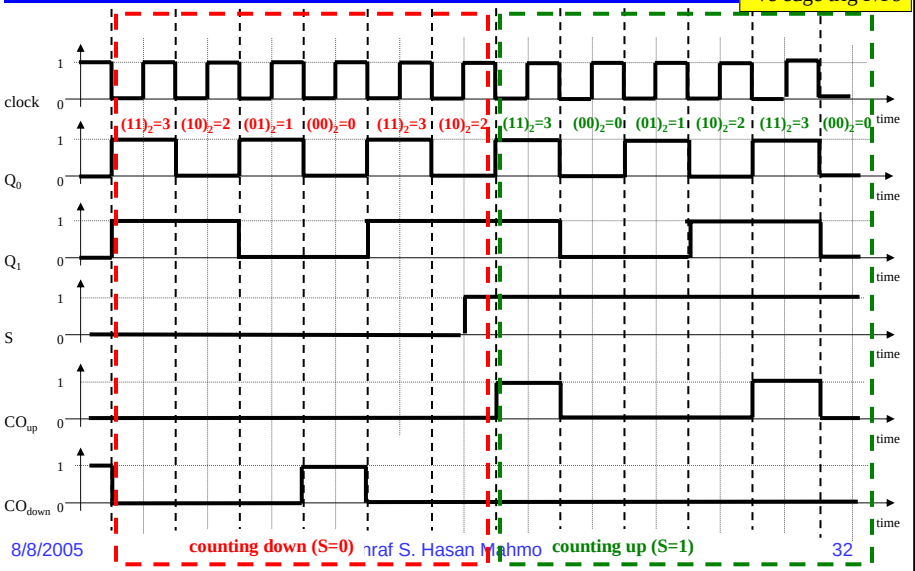
$CO_{down} = Q_0'Q_1'EnS'$



The F/Fs can be +ve or -ve edge triggered
This would NOT affect the equations or the implementation
It would, however, effect when the F/Fs can change state
– see the timing diagram on next slide

Circuit for 2-bit Up-Down Binary Counter –Timing Diagrams

Assume En = 1
-ve edge trig F/Fs



4-bit Up-Down Binary Counter

- You can show that input equations for 4-bit up-down binary counter are given by (compare to 2-bit case):

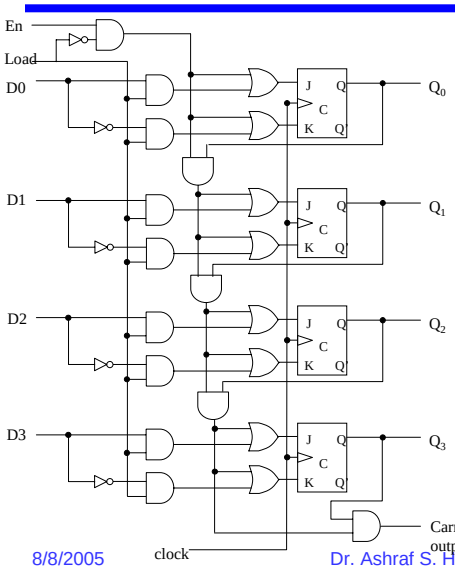
TQ0 = En
TQ1 = Q0.S.En + Q0'.S'.En
TQ2 = Q1.Q0.S.En + Q1'.Q0'.S'.En
TQ3 = Q2.Q1.Q0.S.En + Q2'.Q1'.Q0'.S'.En

CO_{up} = Q0.Q1.Q2.Q3.S.En
CO_{down} = Q0'.Q1'.Q2'.Q3'.S'.En

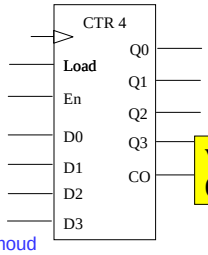
Binary Counter with Parallel Load

- **Counter with parallel load**: capability for transferring an initial binary number into the counter prior to the count operation

Binary Counter with Parallel Load - Circuit



Inputs		Operation
Load	En	
0	0	Hold status
0	1	Count Up
1	x	Transfer of data

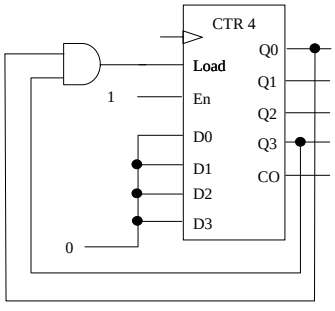


VERY useful (versatile) circuit

BCD Counter

- **<u>BCD Counter</u>**: 0000→0001→0010→…→1000→1001→0000→0010→…
- We can use the use binary counter with load (previous slide) as follows:

Note: When $Q_3Q_2Q_1Q_0 = 1001$, the load signal is equal to 1 and the counter loads 0000 and starts counting again



Other Counters

- **Divide-by-N counter OR Modulo-N counter**: A counter that goes through a repeated sequence of N state
- This is the general definition of the counter
- The 4-bit Binary counter ➔ divide-by-16 counter or modulo-16 counter (because we have 16 states)

BCD Counter - Revisited

- **Problem**: Design a BCD counter (without En) using *T* flip-flops with an output signal that can be used to extend the counter
- **Solution**:
  - Let's design one state (4-bit BCD which counts 0000→0001→ ... →1000→1001→0000→ ...)
  - Use the output 1001 to generate the carry out signal

BCD Counter – State Table

Present State				Next State				Output	Flip-flops inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	Y	T_{Q8}	T_{Q4}	T_{Q2}	T_{Q1}
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	1	0	0	1

T Flip-Flop

Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

BCD Counter – Input Equation

X

Q_2Q_1 / Q_8Q_4	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	x	x	x
10	1	1	x	x

$T_{Q1} = 1$

Q_2Q_1 / Q_8Q_4	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	x	x	x	x
10	0	0	x	x

$T_{Q4} = Q_1Q_2$

Q_2Q_1 / Q_8Q_4	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	x	x	x	x
10	0	0	x	x

$T_{Q2} = Q_1Q_8'$

Q_2Q_1 / Q_8Q_4	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	x	x	x	x
10	0	1	x	x

$T_{Q8} = Q_1Q_8 + Q_1Q_2Q_4$

BCD Counter – Input Equation

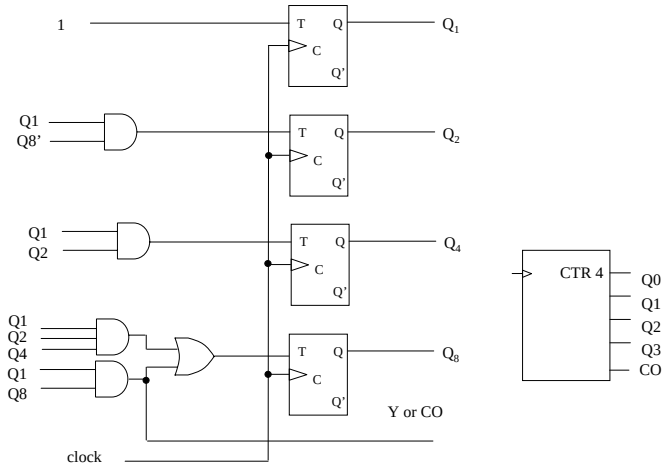
$T_{Q1} = 1$

$T_{Q2} = Q_1Q_8'$

$T_{Q4} = Q_1Q_2$

$T_{Q8} = Q_1Q_8 + Q_1Q_2Q_4$

$Y = Q_8Q_1$



Decimal Numbers Counters

- Synchronous BCD counters (same as the one built in previous slide) can be cascaded to form counters for decimal numbers of any length

- **Exercise 1**: Using the logic diagram of previous slide, design a counter which counts from 0 till 99 and then repeats
- **Exercise 2**: Given synchronous BCD counter with parallel load blocks (on slide 31) and perhaps using other logic gates design a stop watch that times up to one minute
  - What should be the frequency of the driving clock signal?

Arbitrary Count Sequence

- **Problem**: Design a counter that has a repeated sequence of 6 states, as listed in table. In this sequence, flip-flops B and C repeat the binary count 00, 01, 10, while flip-flop A alternates between 0 and 1 every three counts.
- **Solution**:

Present State			Next State		
A	B	C	A	B	C
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

Arbitrary Count Sequence – State Table

- Assuming *JK* flip-flops

Present State			Next State			Flip-flop Inputs					
A	B	C	A	B	C	JA	KA	JB	KB	JC	KC
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	1	0	0	1	X	X	1	0	X
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	0	0	0	X	1	x	1	0	x

Arbitrary Count Sequence – Input Equations

A \ BC	00	01	11	10
0	0	0	x	1
1	x	x	x	x

JA = B

A \ BC	00	01	11	10
0	0	1	x	x
1	0	1	x	x

JB = C

A \ BC	00	01	11	10
0	1	x	x	0
1	1	x	x	0

JC = B'

A \ BC	00	01	11	10
0	x	x	x	x
1	0	0	x	1

KA = B

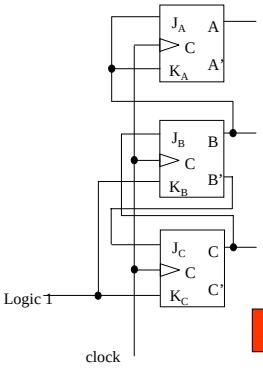
A \ BC	00	01	11	10
0	x	x	x	1
1	x	x	x	1

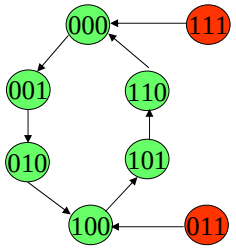
KB = 1

A \ BC	00	01	11	10
0	x	1	x	x
1	x	1	x	x

KC = 1

Arbitrary Count Sequence – Circuit and Unused States



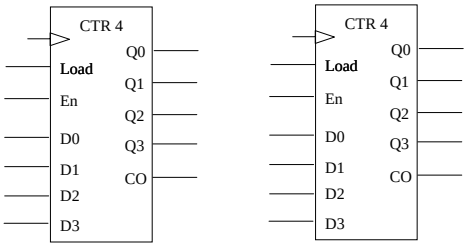


1) What if the counter "*finds itself*" in state 111 or state 011? Will the counter be able to proceed (count) normally afterward? How?

2) Is this circuit safe or reliable?

Problem 5-23:

- **Problem**: Using the two binary counters shown in figure and logic gates, construct a *binary* counter that counts from 9 through *binary* 69. Add an additional input to the counter that initializes it synchronously to 9 when the INIT is 1



Problem 5-23: Solution

- **Solution**:

The key word in the problem statement is "*BINARY*"

Hence the designed counter should count in binary from binary 9 through binary 69 and then back to binary 9 and so on

Binary 9 ➔ 0000 1001

Binary 69 ➔ 0100 0101

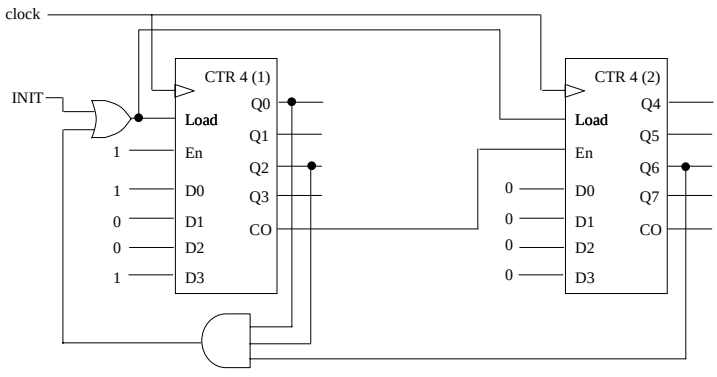
Hence it is obvious that the two counters providing 8 Qs are enough to represent 69

Problem 5-23: Solution-cont'd

- **Solution**:
- The counting sequence is as shown in table
- Note that:
  - The we can assign first bit counter to count $Q_3Q_2Q_1Q_0$ while the second is assigned $Q_7Q_6Q_5Q_4$
  - Note that second counter increments count only if $Q_3Q_2Q_1Q_0$ is equal to 1111; i.e the CO signal first counter is 1
  - During count – first counter counts normally from 01001 all the way to 1111 and back to 0000, 0001, etc except when binary 69 is reached – 1001 is reloaded again
  - During count – second counter counts normally from 0000 all the way to 0100 and then it restarts from 0000 again.

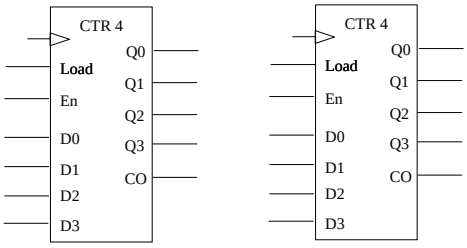
No	Binary $Q_7Q_6Q_5Q_4$ $Q_3Q_2Q_1Q_0$
9	0000 1001
10	0000 1010
11	0000 1011
12	0000 1100
.	.
15	0000 1111
16	0001 0000
17	0001 0001
18	0001 0010
19	0001 0011
20	0001 0100
21	0001 0101
.	.
68	0100 0100
69	0100 0101
9	0000 1001
10	0000 1010
.	.

Problem 5-23: Solution-cont'd



Exercise: BCD Counter

- **Problem**: Using the two binary counters shown in figure and logic gates, construct a *BCD* counter that counts from 9 through *BCD* 69. Add an additional input to the counter that initializes it synchronously to 9 when the INIT is 1



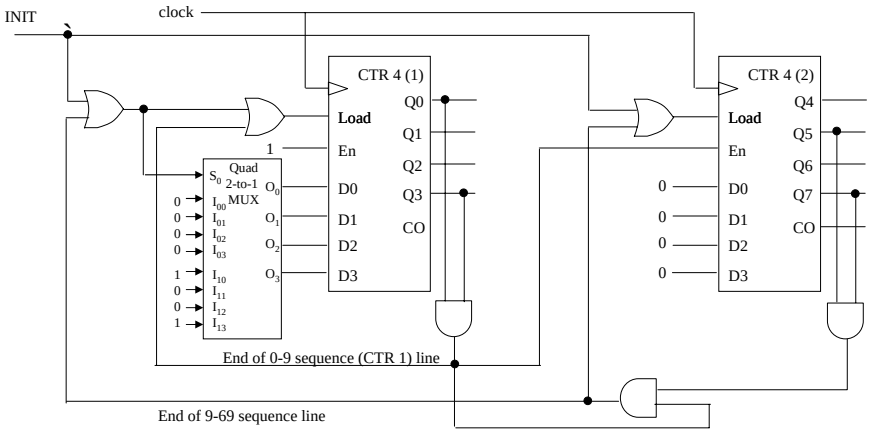
Exercise: BCD Counter – cont'd

Solution:

- The key word in the problem statement is "*BCD*"
- The counting sequence is as shown in table:
  - Counter 1 ($Q_3Q_2Q_1Q_0$) counts 9, 0, 1,2, …, 9, 0 and so on
  - Counter 1 increments every clock cycle
  - Counter 2 ($Q_7Q_6Q_5Q_4$) counts 0, 1, 2, 3, 4, 5, 6, 0, 1, and so on
  - Counter 2 increments every time counter 1 reaches 1001
  - Counter 1 loads
    - 0000 when 9 is reached; OR
    - 1001 if INIT is 1 or 69 is reached
  - Counter 2 loads 0000 when 69 is reached or when INIT is 1

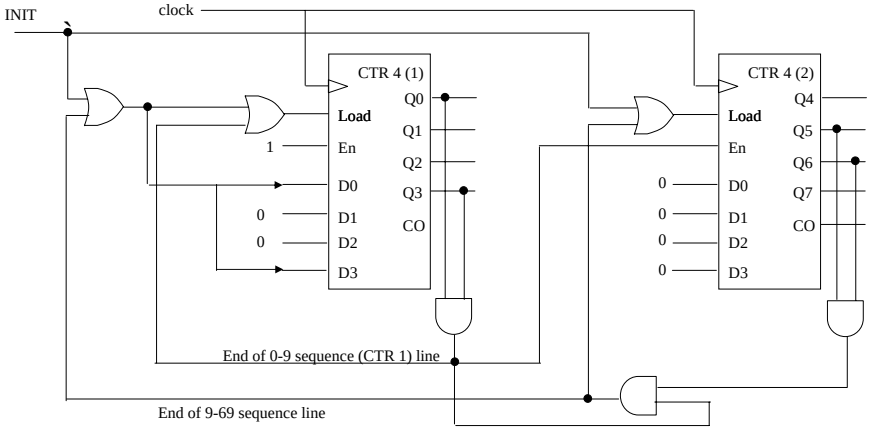
No	Binary $Q_7Q_6Q_5Q_4\ Q_3Q_2Q_1Q_0$
9	0000 1001
10	0001 0000
11	0001 0001
12	0001 0010
.	.
15	0001 0101
16	0001 0110
17	0001 0111
18	0001 1000
19	0001 1001
20	0010 0000
21	0010 0001
.	.
68	0110 1000
69	0110 1001
9	0000 1001
10	0001 0000
.	.

Exercise: BCD Counter – cont'd



The circuit can be built without using the MUX as shown in next slide

Exercise: BCD Counter – cont'd



Problems of Interest:

- Problems (registers): 5-3, 5-4, 5-6, 5-7, 5-8, 5-9, 5-10, 5-11, 5-12,
- Problems (counters) : 5-16, 5-17[S], 5-18, 5-19, 5-20 (a) [S]