CS 221 – Digital Design

Counters

JAISAL SINGH
(120123020)

NEERAJ YADAV
(120123026)

DHEERAJ KHATRI
(120101021)

DATE: 17 OCTOBER 2013

Counters

- In digital logic and computing, a **counter** is a device which stores (and sometimes displays) the number of times a particular event or process has occurred, often in relationship to a clock signal.
- We examine special types of addition and subtraction operations, which are used for the purpose of counting.

In electronics, counters can be implemented quite easily using register-type circuits such as the flip-flop, and a wide variety of classifications exist:

1. Ripple counters (Asynchronous)

The flip-flop output transition serves as a source for triggering other flip-flops i.e the C input of some or all flip-flops are triggered NOT by the common clock pulses

Eg:-Binary ripple counters BCD ripple counters

2. Synchronous counters

The C inputs of all flip-flops receive the common clock pulses.

E.g.:-Binary counter, Up-down Binary counter, BCD Binary counter, Ring counter, Johnson counter

BCD

- In computing and electronic systems, **binary-coded decimal** (**BCD**) (sometimes called **natural binary-coded decimal**, **NBCD**) or, in its most common modern implementation, **packed decimal**, is an encoding for decimal numbers in which each digit is represented by its own binary sequence. Its main virtue is that it allows easy conversion to decimal digits for printing or display, and allows faster decimal calculations. Its drawbacks are a small increase in the complexity of circuits needed to implement mathematical operations. Uncompressed BCD is also a relatively inefficient encoding—it occupies more space than a purely binary representation.
- In BCD, a digit is usually represented by four bits which, in general, represent the decimal digits 0 through 9. Other bit combinations are sometimes used for a sign or for other indications (e.g., error or overflow).
- Although uncompressed BCD is not as widely used as it once was, decimal fixed-point and floating-point are still important and continue to be used in financial, commercial, and industrial computing.

Basics for BCD

- To encode a decimal number using the common BCD encoding, each decimal digit is stored in a 4-bit nibble:
- Decimal: 0   1   2   3   4   5   6   7   8   9
- Thus, the BCD encoding for the number 127 would be:

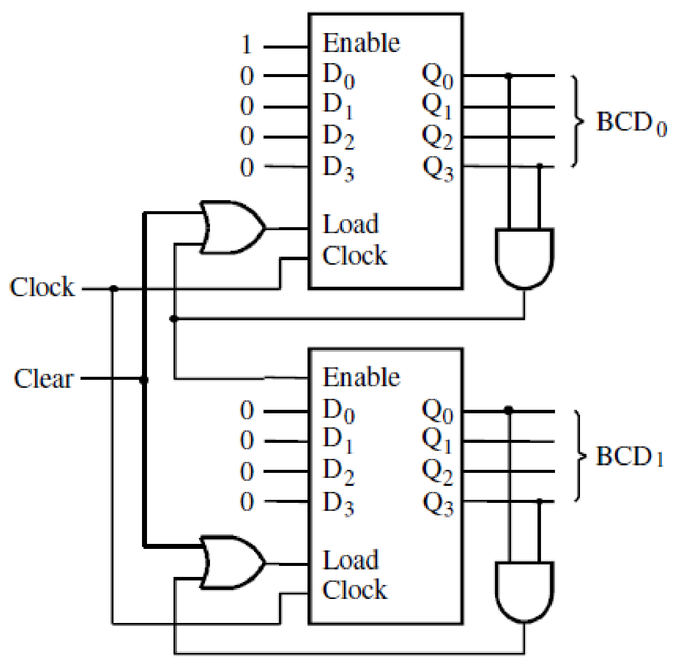
0001 0010 0111

- Whereas the pure binary number would be:

0111 1111

Binary-coded-decimal(BCD) counters

A Two-digit BCD Counter



- Consists of two modulo-10 counters, one for each BCD digit.
- It is necessary to reset the four flip-flops after the count of 9 has been obtained. Thus the *Load* input to each stage is equal to 1 when Q3=Q0=1, which causes 0s to be loaded into the flip-flops at the next positive edge of the clock signal.
- Keeping the Enable signal for BCD1 low at all times except when BCD0 = 9

Synchronous Counter

In Asynchronous counter tutorial, we saw that the output of one counter stage is connected directly to the clock input of the next counter stage and so on along the chain, and as a result the asynchronous counter suffers from what is known as "Propagation Delay" in which the timing signal is delayed a fraction through each flip-flop.

However, with the **Synchronous Counter**, the external clock signal is connected to the clock input of EVERY individual flip-flop within the counter so that all of the flip-flops are clocked together simultaneously (in parallel) at the same time giving a fixed time relationship. In other words, changes in the output occur in "synchronization" with the clock signal. This results in all the individual output bits changing state at exactly the same time in response to the common clock signal with no ripple effect and therefore, no propagation delay.

EXCITATION TABLES FOR DIFFERENT FLIP FLOPS

Excitation Table of T FLIP-FLOP is

Present State	Next State	T	Operation
0	0	0	Hold
1	0	1	Toggle
1	1	0	Toggle
0	1	1	Hold

Excitation Table of D FLIP-FLOP is

Present State	Next State	D
0	0	0
1	0	0
1	1	1
0	1	1

Excitation Table of JK FLIP-FLOP is as shown below:

Present State	Next State	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

TRUTH TABLE for 4 bit SYNCHRONOUS BCD COUNTER using T Flip Flop

Persent State Q3Q2Q1Q0	Next State	T3	T2	T1	T0
0000	0001	0	0	0	1
0001	0010	0	0	1	1
0010	0011	0	0	0	1
0011	0100	0	1	1	1
0100	0101	0	0	0	1
0101	0110	0	0	1	1
0110	0111	0	0	0	1
0111	1000	1	1	1	1
1000	1001	0	0	0	1
1001	0000	1	0	0	1

T3,T2,T1,T0 are input T Flip-Flops.

Calculating Values of T Flip Flops

1.Here as T0 is Flipping every Time so value of T0=1(ever)

2.For T1 we can create Truth Table of the same and calculate value -

Truth Table for T1

Q2Q3	00	01	11	10
Q0Q1				
00			X	
01	T1	T1	X	
11	T1	T1	X	X
10			X	X

So value of T1=Q3` Q0

3.For T2 Truth Table is

Q2Q3	00	01	11	10
Q0Q1				
00			X	
01			X	
11	T2	T2	X	X
10			X	X

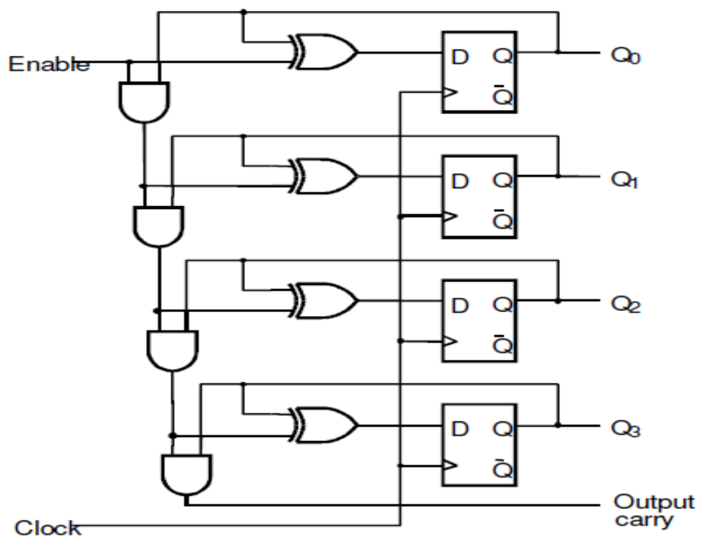
T2=Q0Q1

4.For T3 Truth Table

Q2Q3	00	01	11	10
Q0Q1				
00			X	
01			X	T3
11		T3	X	X
10			X	X

T3 =Q2Q1Q0+Q3Q0

Four-bit Counter with D Flip-flops



D0 = Q0 XOR Enable

D1 = Q1 XOR Q0 & Enable

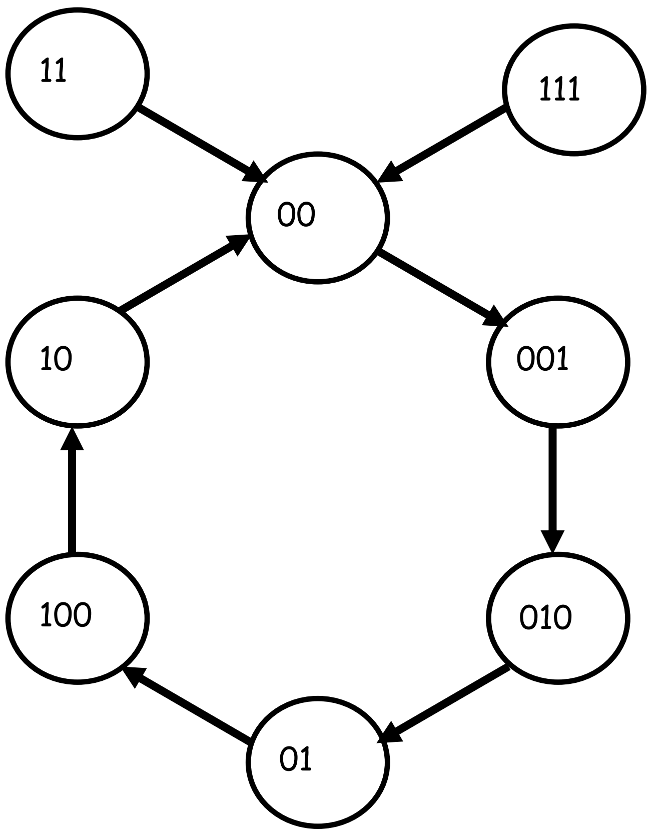
D2 = Q2 XOR Q1 & Q0 & Enable

D3 = Q3 XOR Q2 & Q1 & Q0 & Enable

Unused states

- The examples shown so far have all had 2^n states, and used n flip-flops. But sometimes you may have unused, leftover states.
    - For example, here is a state table and diagram for a counter that repeatedly counts from 0 (000) to 5 (101).
    - What should we put in the table for the two unused states?

Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
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	0	0	0
1	1	0	0	0	0
1	1	1	0	0	0



11
111
00
10
001
100
010
01

Unused states can be don't cares

- To get the simplest possible circuit, you can fill in don't cares for the next states. This will also result in don't cares for the flip-flop inputs, which can simplify the hardware.

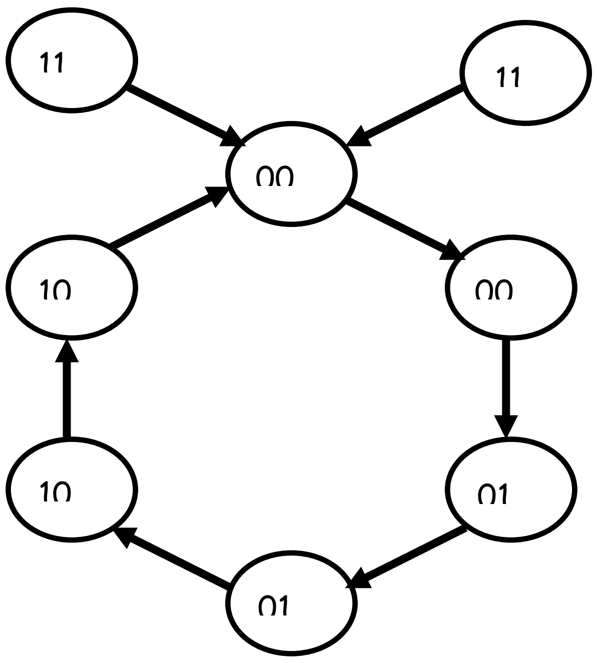
- If the circuit somehow ends up in one of the unused states (110 or 111), its behaviour will depend on exactly what the don't cares were filled in with.

Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
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	0	0	0
1	1	0	x	x	x
1	1	1	x	x	x

…or maybe you do care

- To get the *safest* possible circuit, you can explicitly fill in next states for the unused states 110 and 111.
    - This guarantees that even if the circuit somehow enters an unused state, it will eventually end up in a valid state.
    - This is called a self-starting counter.

Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
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	0	0	0
1	1	0	0	0	0
1	1	1	0	0	0


11
11
00
10
00
10
01
01