OKOH JOSHUA CHUKUKA

ABUAD

COMPUTER ENGINEERING

17/ENG02/066

COE 312

Synchronous Counter



Synchronous Counters are so called because the clock input of all the individual flip-flops within the counter are all clocked together at the same time by the same clock signal

In the previous Asynchronous binary 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.

The result of this is that 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 “synchronisation” with the clock signal.

The result of this synchronisation is that 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.





# Asynchronous Counter

Asynchronous Counters use flip-flops which are serially connected together so that the input clock pulse appears to ripple through the counter



An **Asynchronous counter** can have 2n-1 possible counting states e.g. MOD-16 for a 4-bit counter, (0-15) making it ideal for use in Frequency Division applications. But it is also possible to use the basic asynchronous counter configuration to construct special counters with counting states less than their maximum output number. For example, modulo or MOD counters.

This is achieved by forcing the counter to reset itself to zero at a pre-determined value producing a type of asynchronous counter that has truncated sequences. Then an n-bit counter that counts up to its maximum modulus ( 2n ) is called a full sequence counter and a n-bit counter whose modulus is less than the maximum possible is called a **truncated counter**.

But why would we want to create an asynchronous truncated counter that is not a MOD-4, MOD-8, or some other modulus that is equal to the power of two. The answer is that we can by using combinational logic to take advantage of the asynchronous inputs on the flip-flop.

If we take the modulo-16 asynchronous counter and modified it with additional logic gates it can be made to give a decade (divide-by-10) counter output for use in standard decimal counting and arithmetic circuits.

Such counters are generally referred to as Decade Counters. A decade counter requires resetting to zero when the output count reaches the decimal value of 10, ie. when DCBA = 1010 and to do this we need to feed this condition back to the reset input. A counter with a count sequence from binary “0000” (BCD = “0”) through to “1001” (BCD = “9”) is generally referred to as a BCD binary-coded-decimal counter because its ten state sequence is that of a BCD code but binary decade counters are more common.

|  |  |  |
| --- | --- | --- |
| Clock Count | Output bit Pattern | Decimal Value |
| QD | QC | QB | QA |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 1 | 0 | 2 |
| 4 | 0 | 0 | 1 | 1 | 3 |
| 5 | 0 | 1 | 0 | 0 | 4 |
| 6 | 0 | 1 | 0 | 1 | 5 |
| 7 | 0 | 1 | 1 | 0 | 6 |
| 8 | 0 | 1 | 1 | 1 | 7 |
| 9 | 1 | 0 | 0 | 0 | 8 |
| 10 | 1 | 0 | 0 | 1 | 9 |
| 11 | Counter Resets its Outputs back to Zero |

### Decade Counter Timing Diagram



# MOD Counters

MOD Counters are cascaded counter circuits which count to a set modulus value before resetting



The job of a counter is to count by advancing the contents of the counter by one count with each clock pulse. Counters which advance their sequence of numbers or states when activated by a clock input are said to operate in a “count-up” mode. Likewise, counters which decrease their sequence of numbers or states when activated by a clock input are said to operate in a “count-down” mode. Counters that operate in both the UP and DOWN modes, are called bidirectional counters.

Counters are sequential logic devices that are activated or triggered by an external timing pulse or clock signal. A counter can be constructed to operate as a synchronous circuit or as an asynchronous circuit. With synchronous counters, all the data bits change synchronously with the application of a clock signal. Whereas an asynchronous counter circuit is independent of the input clock so the data bits change state at different times one after the other.

Then counters are sequential logic devices that follow a predetermined sequence of counting states which are triggered by an external clock (CLK) signal. The number of states or counting sequences through which a particular counter advances before returning once again back to its original first state is called the modulus (MOD). In other words, the modulus (or just modulo) is the number of states the counter counts and is the dividing number of the counter.

# Frequency Division

Frequency Division uses divide-by-2 toggle flip-flops as binary counters to reduce the frequency of the input clock signal



In the Sequential Logic tutorials we saw how D-type Flip-Flop´s work and how they can be connected together to form a Data Latch.

Another useful feature of the D-type Flip-Flop is as a binary divider, for **Frequency Division** or as a “divide-by-2” counter. Here the inverted output terminal Q (NOT-Q) is connected directly back to the Data input terminal D giving the device “feedback” as shown below.

ADVANTAGES OF SYNCHRONOUS COUNTER OVER ASYNCHRONOUSE COUNTER

1. The one advantage of synchronous counter over asynchronous counter is, it can operate on higher frequency than asynchronous counter as it does not have cumulative delay because of same clock is given to each flip flop.

2. Synchronous counters are easier to design than asynchronous counters. are all clocked together at the same time with the same clock signal. Due to this common clock pulse all output states switch or change simultaneously. ... Overall faster operation may be achieved compared to Asynchronous counters.

3. Asynchronous counters can be easily designed by T flip flop or D flip flop. These are also called as Ripple counters, and are used in low speed circuits. They are used as Divide by- n counters, which divide the input by n, where n is an integer

## **JK Flip Flop-**

|  |
| --- |
| JK flip flop is a refined & improved version of [**SR Flip Flop**](https://www.gatevidyalay.com/sr-flip-flop/)that has been introduced to solve the problem of indeterminate statethat occurs in SR flip flop when both the inputs are 1. |

In JK flip flop,

* Input J behaves like input S of SR flip flop which was meant to set the flip flop.
* Input K behaves like input R of SR flip flop which was meant to reset the flip flop.

## **Construction of JK Flip Flop-**

There are following two methods for constructing a JK flip flop-



1. By using SR flip flop constructed from NOR latch
2. By using SR flip flop constructed from NAND latch

### **1. Construction of JK Flip Flop By Using SR Flip Flop Constructed From NOR Latch-**

This method of constructing JK Flip Flop uses-

* SR Flip Flop constructed from NOR latch
* Two other connections

### **Logic Circuit-**

The logic circuit for JK Flip Flop constructed using SR Flip Flop constructed from NOR latch is as shown below-



### **2. Construction of JK Flip Flop By Using SR Flip Flop Constructed From NAND Latch-**

This method of constructing JK Flip Flop uses-

* SR Flip Flop constructed from NAND latch
* Two other connections

### **Logic Circuit-**

The logic circuit for JK Flip Flop constructed using SR Flip Flop constructed from NAND latch is as shown below-

### **Logic Symbol-**

The logic symbol for JK Flip Flop is as shown below-



### **Truth Table-**

The truth table for JK Flip Flop is as shown below-

|  |  |
| --- | --- |
| **INPUTS** | **OUTPUTS** |
| **J** | **K** | **Qn****(Present State)** | **Qn+1****(Next State)** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

### **Truth Table**

The above truth table may be reduced as-

|  |  |  |
| --- | --- | --- |
| **INPUTS** | **OUTPUTS** | **REMARKS** |
| **J** | **K** | **Qn****(Present State)** | **Qn+1****(Next State)** | **States and Conditions** |
| 0 | 0 | X | Qn | Hold State condition J = K = 0 |
| 0 | 1 | X | 0 | Reset state condition J = 0 , K = 1 |
| 1 | 0 | X | 1 | Set state condition J = 1 , K = 0 |
| 1 | 1 | X | Q’n | Toggle state condition J = K = 1 |

### **Truth Table**

### **Excitation Table-**

The excitation table of any flip flop is drawn using its truth table.

|  |
| --- |
| **What is excitation table?**For a given combination of present state Qn and next state Qn+1, excitation table tell the inputs required. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Qn** | **Qn+1** | **S** | **R** |
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

### **Excitation Table**