**Experiment-5**

**Aim: To design n-bit synchronous and asynchronous counter**

**Tools Required : Logisim 2.7.1(Open Source)**

**Theory :**

**Counter :** A counter is a sequential circuit that goes through a prescribed sequence of states upon the application of input pulses. The input pulses called count pulses, may be clock pulses, or they may originate from an external source and may occur at prescribed intervals of time or at random.

**Asynchronous Counter :**

Asynchronous counters, or ripple counters, apply the clock pulse only to the first flip-flop, with each subsequent flip-flop triggered by the previous one’s output. This causes a "ripple" effect, where flip-flops toggle sequentially rather than simultaneously.

In a 3-bit asynchronous counter using J-K flip-flops, the design can be described as follows: The J and K inputs of all three flip-flops (FF0, FF1, and FF2) are connected to HIGH (logic 1), configuring them to toggle on each clock transition.

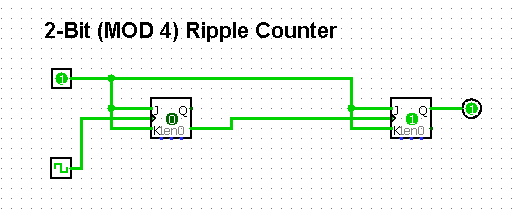
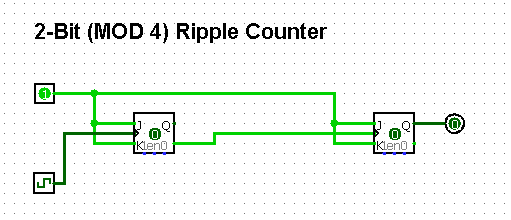
**Synchronous Counter :**

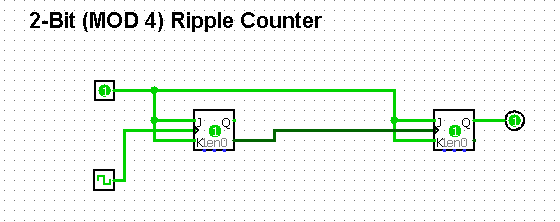
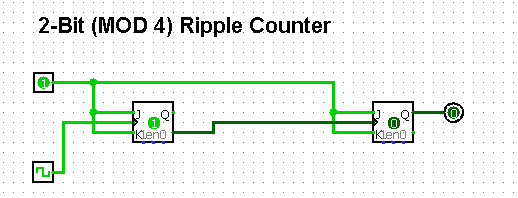
Synchronous counters are designed in such a way that the clock pulses are applied to the CP inputs of all the flip-flops. The common pulse triggers all the flip-flops simultaneously, rather than one at a time in succesion.

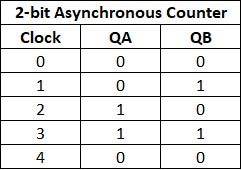
In the 3-bit synchronous counter, we have used three j-k flip-flops. As in the diagram, The J and K inputs of FF0 are connected to HIGH. The inputs J and K of FF1 are connected to the output of FF0, and the J and K inputs of FF2 are connected to the output of an AND gate, which is fed by the outputs of FF0 and FF1.

**Observation :**

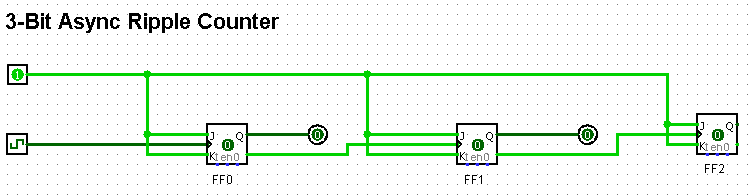
## **2-bit Asynchronous Counter(2-bit Ripple Counter):**

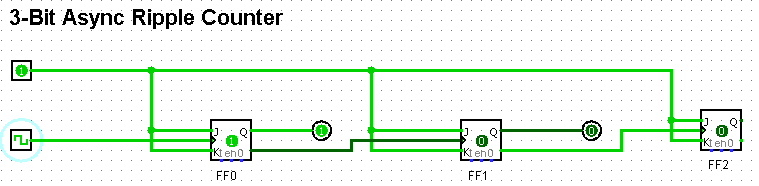


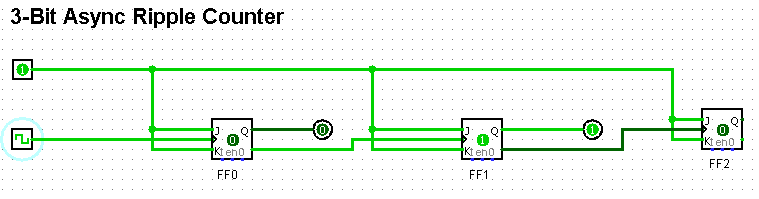


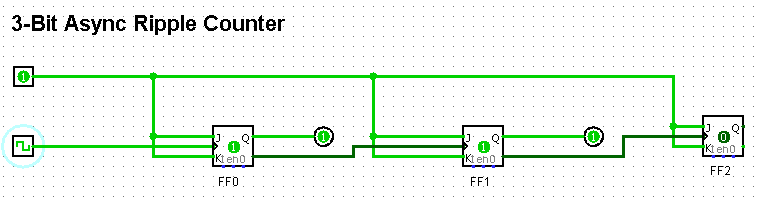
**Truth Table :**

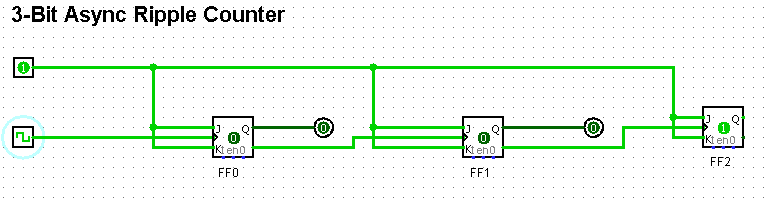
* **3-bit Asynchronous Counter(3-bit Ripple Counter :**

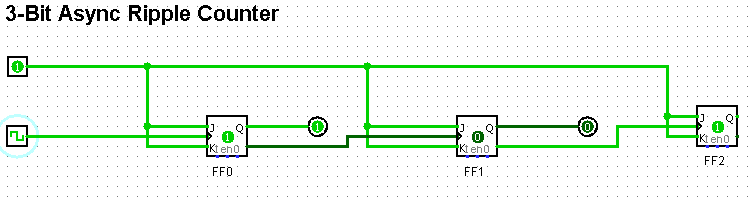
****

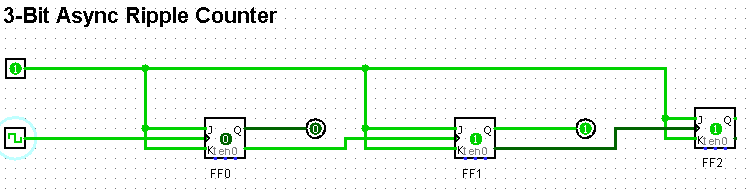
****

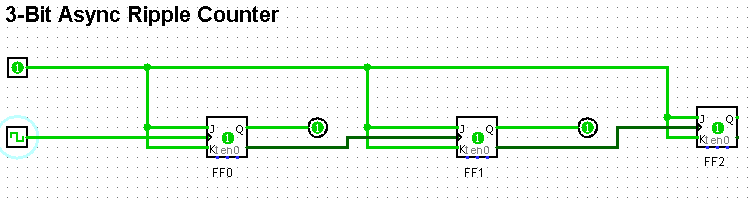


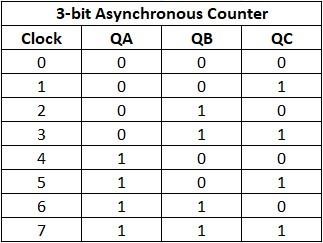




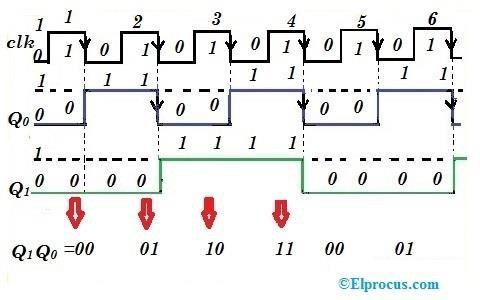




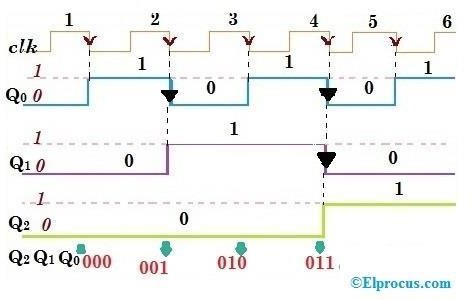


**Truth Table :**

* **Timing Diagram of 2-bit Asynchronous Counter:**

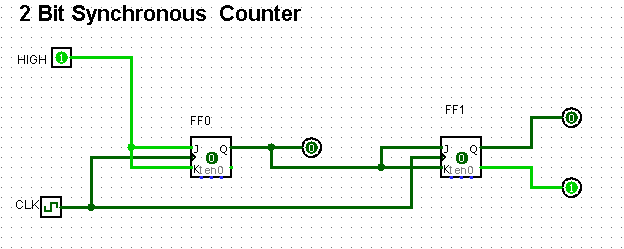


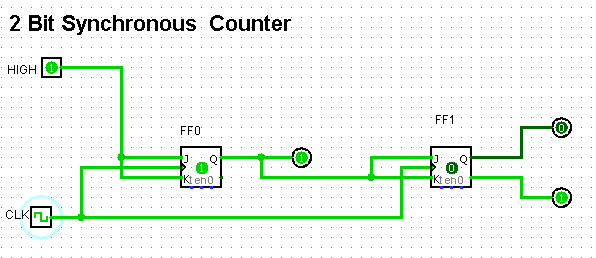
* **Timing Diagram of 3-bit Asynchronous Counter:**

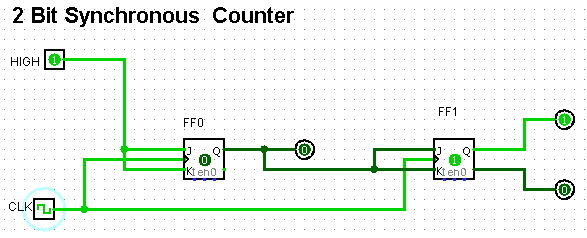


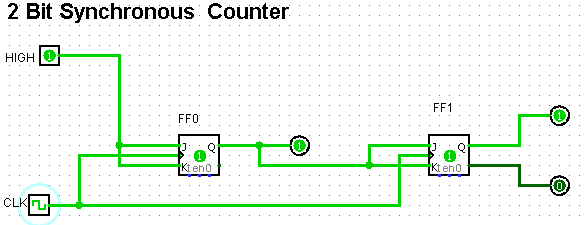
**Synchronous Counter:**

* + - **2 bit Synchronous Counter (2 bit ripple counter )**

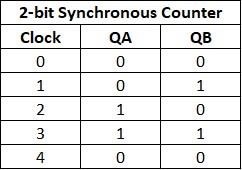




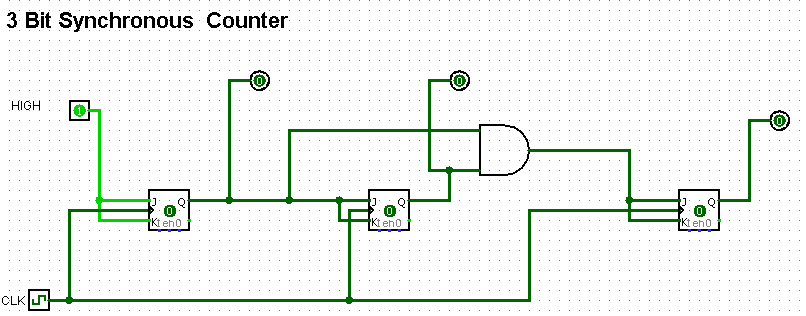


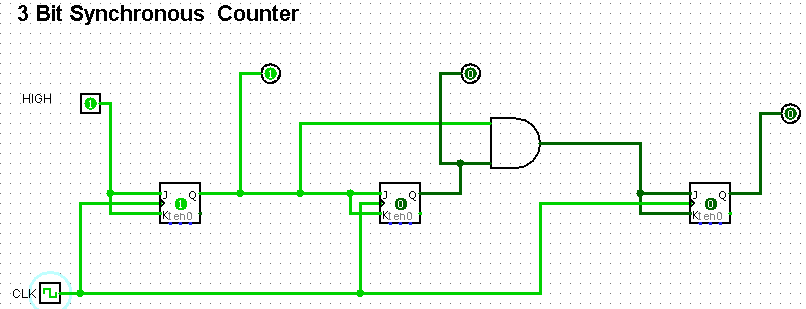


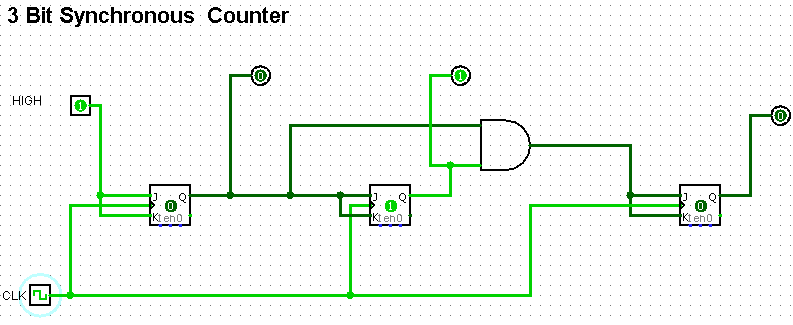
* **Truth Table:**

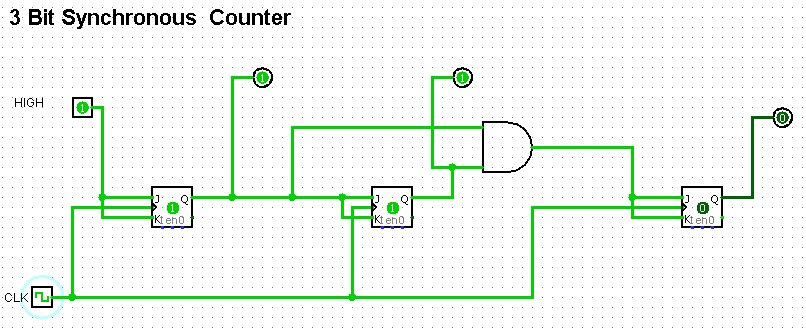


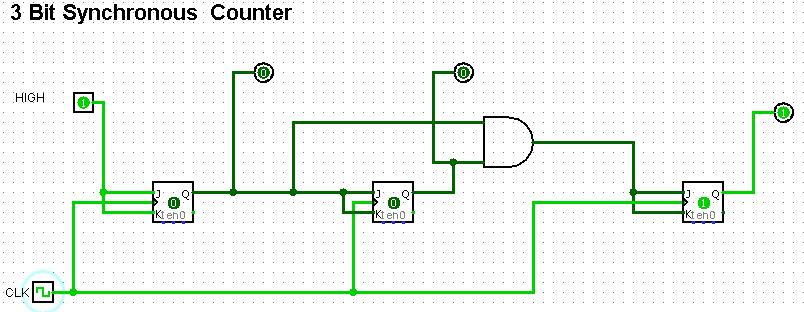
* **3-bit Synchronous Counter:**

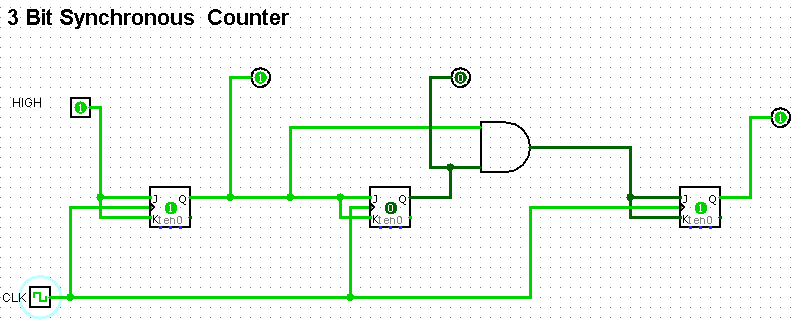


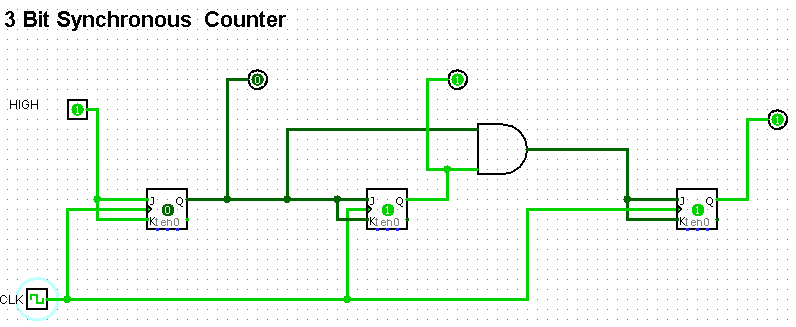


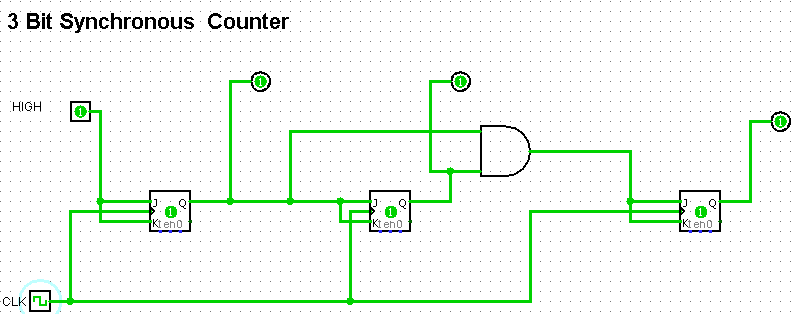




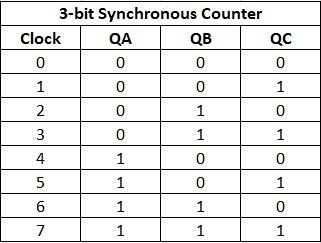








* **Truth Table :**



**Result:**

Truth Tables of Asynchronous Counter and Synchronous Counter are verified via Logisim.

**Conclusion:**

Circuit simulation and their TT verification can be achieved using open source “Logisim” software.

**Lab Quiz:**

Q1. What are asynchronous and synchronous counters?

**Asynchronous Counter :**

Asynchronous counters, or ripple counters, apply the clock pulse only to the first flip-flop, with each subsequent flip-flop triggered by the previous one’s output. This causes a "ripple" effect, where flip-flops toggle sequentially rather than simultaneously.

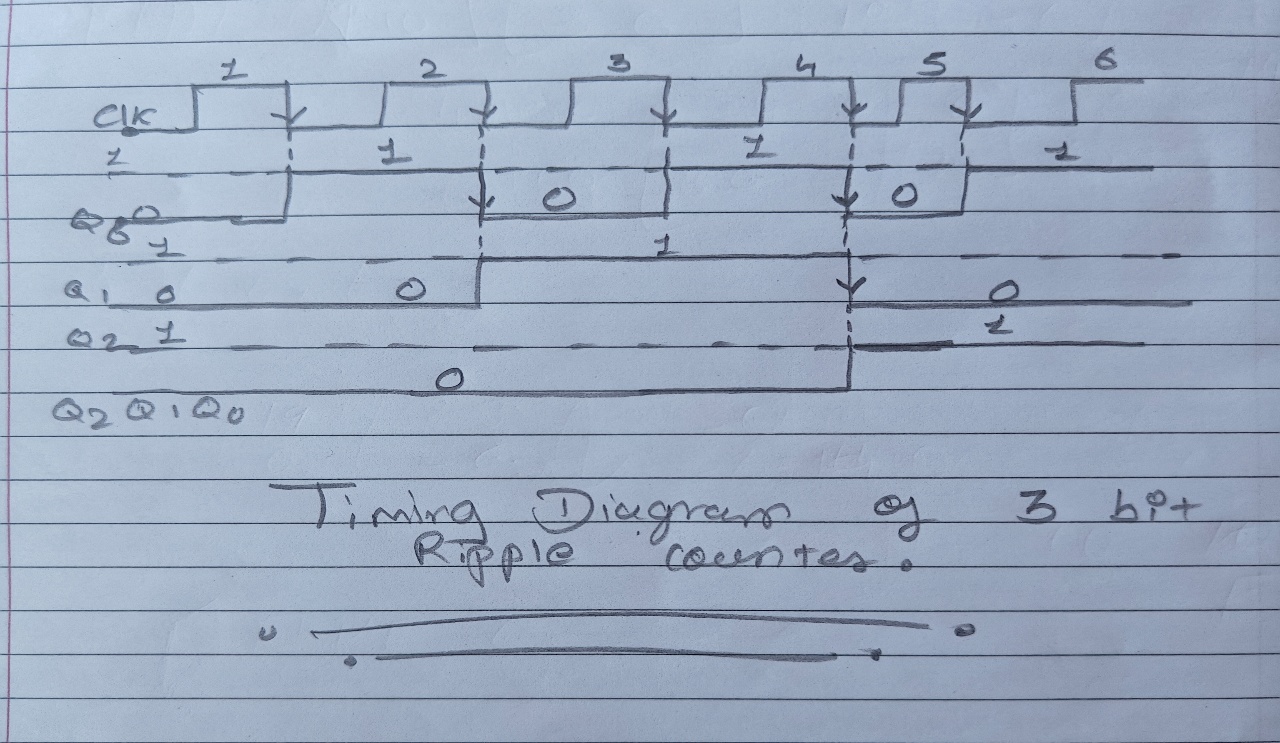
In a 3-bit asynchronous counter using J-K flip-flops, the design can be described as follows: The J and K inputs of all three flip-flops (FF0, FF1, and FF2) are connected to HIGH (logic 1), configuring them to toggle on each clock transition.

**Synchronous Counter :**

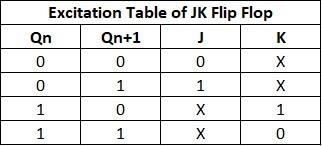
Synchronous counters are designed in such a way that the clock pulses are applied to the CP inputs of all the flip-flops. The common pulse triggers all the flip-flops simultaneously, rather than one at a time in succesion.

In the 3-bit synchronous counter, we have used three j-k flip-flops. As in the diagram, The J and K inputs of FF0 are connected to HIGH. The inputs J and K of FF1 are connected to the output of FF0, and the J and K inputs of FF2 are connected to the output of an AND gate, which is fed by the outputs of FF0 and FF1.

**Q2. Draw timing diagram of a 3-bit ripple counter.**

****

**Q3. Write down excitation table of JK FF**



**Q4. What is the significance of “don’t care” in the design of a synchronous counter? Answer:**

* In the design of a synchronous counter, a "don't care" condition refers to a specific input combination that will never occur during normal operation, meaning the output state for that combination is irrelevant and can be designed to be either logic high or low without affecting the counter's functionality; essentially, it allows for flexibility in the circuit design by not needing to define a specific output for unused states.