**How to use Multisim step by step?**

Step 1: Open Multisim. Begin by drawing your schematic in the Multisim environment.

Step 2: Place Components.

Step 3: Wire Components.

Step 4: Place a Simulation Source.

Step 5: Place Measurement Instruments.

Step 6: Run a Simulation.

**Digital Electronics Basics:**

An IC is a collection of electronic components – resistors, transistors, capacitors, etc. – all stuffed into a tiny chip, and connected together to achieve a common goal. Ex: op amps, 555 timers, voltage regulators, motor controllers etc.

**Logic families**:

A logic family of monolithic digital integrated circuit devices is a group of electronic logic gates, constructed using one of several different designs, usually with compatible logic levels and power supply characteristics within a family. A "logic family" may also refer to a set of techniques used to implement logic within VLSI integrated circuits such as central processors, memories, or other complex functions.

Some of the popularly used logic families are:

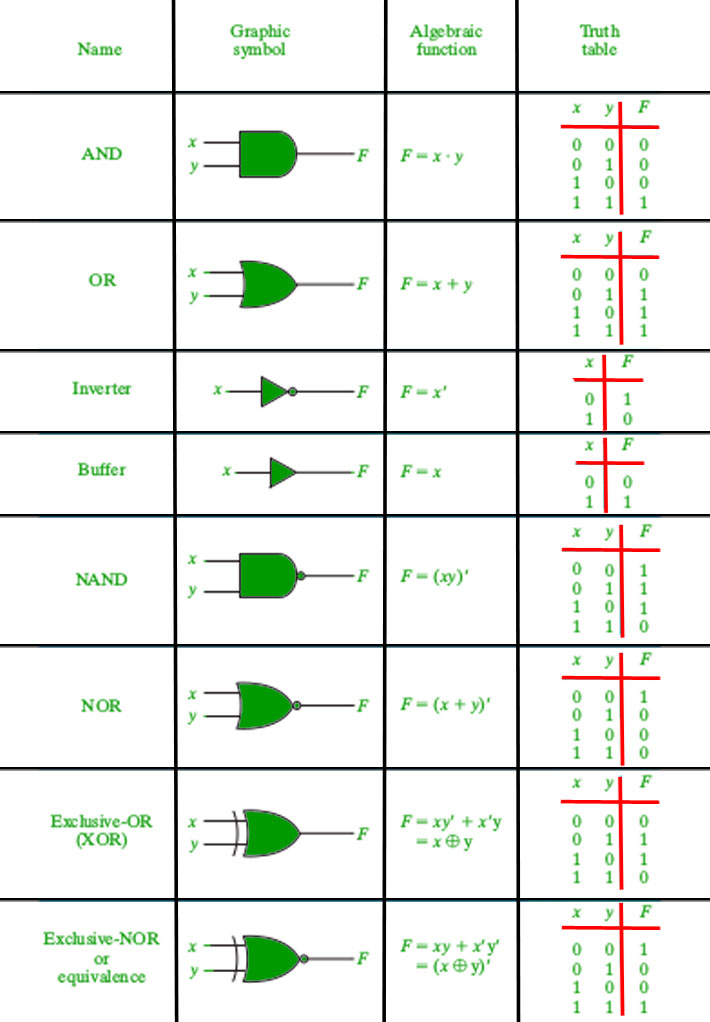
◦ **TTL** (Transistor-Transistor Logic): Made of bipolar transistors mainly used for logical operations.

◦ **CMOS** (Complementary Metal Oxide Semiconductor): Made from MOSFETs and requires less power to operate and wide range of supply voltages.

◦ **ECL** (Emitter Coupled Logic): Used for extremely high speed operation.

◦ **NMOS, PMOS**: Used for Very Large Scale Integrated Circuits (VLSI) Some of the IC’s used in this laboratory are IC 74XX series, IC 74151, IC7474, and IC7476 etc.

**Basic Gates**



**Z**

**EXPERIMENT 6**

**Realization of Adder and Subtractor using IC 74139/ 74155N (Demux/Decoder) and**

**Binary to Gray code conversion & vice versa using 74139/ 74155**

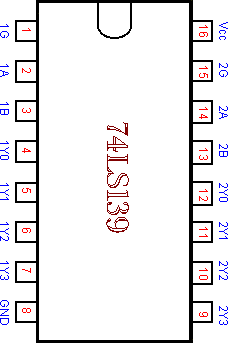
**Aim**: Realize (i) Adders & Subtractors using IC74139.

(ii) Binary to Gray code conversion & vice-versa (74139)

**Components Required:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.No** | **Group** | **Family** | **Component** |
| **1.** | **TTL** | **74STD** | **74LS139,74LS00** |
| **2.** | **Sources** | **DIGITAL\_SOURCES** | **INTERACTIVE\_DIGITAL\_CONSTANT** |
| **3.** | **Sources** | **POWER\_SOURCES** | **VCC, GROUND** |
| **4.** | **Indicators** | **PROBE** | **PROBE\_GRENN, PROBE\_RED** |

## IC 74139 DEMUX/ DECODER

** Pin Digram: Truth table:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Inputs** | | | **Outputs** | | | |
| **G** | **B** | **A** | **Y0** | **Y1** | **Y2** | **Y3** |
| **1** | **X** | **X** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **0** | **1** | **1** | **1** |
| **0** | **0** | **1** | **1** | **0** | **1** | **1** |
| **0** | **1** | **0** | **1** | **1** | **0** | **1** |
| **0** | **1** | **1** | **1** | **1** | **1** | **0** |

**Theory:**

A Demultiplexer is a circuit that receives information from a single line and directs it to one of 2n possible output lines. The selection of a specific output is controlled by the bit combination of n selection lines.

**Realization of Half ADDER/ Half Subtractor Using IC**

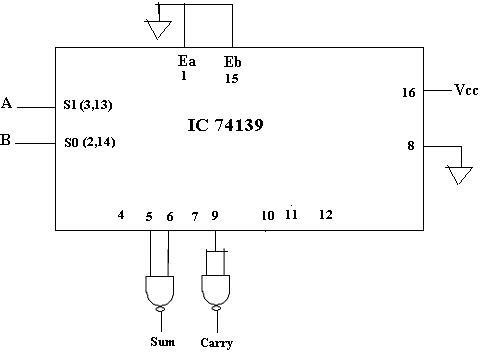
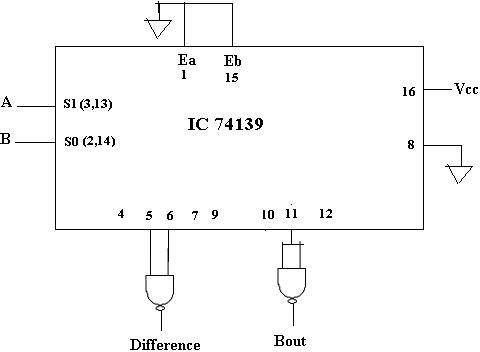
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **Sum** | **Carry** | **Diff** | **Bout** |
| **0** | **0** | **0** | **0** | **0** | **0** |
| **0** | **1** | **1** | **0** | **1** | **1** |
| **1** | **0** | **1** | **0** | **1** | **0** |
| **1** | **1** | **0** | **1** | **0** | **0** |

**Sum: Σ1,2**

**Carry: Σ3**

**Diff: Σ1,2**

**Bout: Σ1**

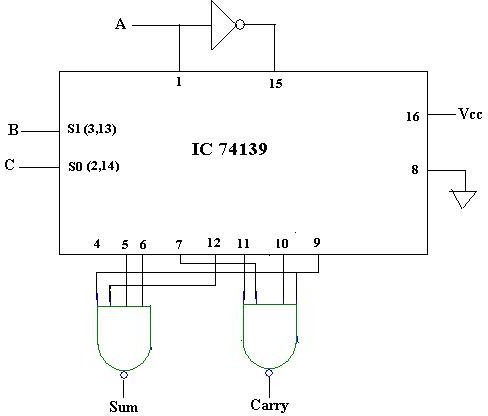
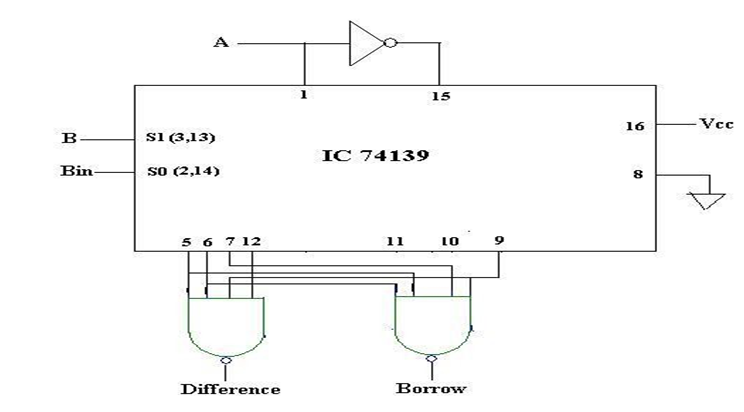
**Logic Diagram for Half Adder Logic Diagram for Half Subtractor**

**Components Required:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.No** | **Group** | **Family** | **Component** |
| **1.** | **TTL** | **74STD** | **74LS139,74LS20** |
| **2.** | **Sources** | **DIGITAL\_SOURCES** | **INTERACTIVE\_DIGITAL\_CONSTANT** |
| **3.** | **Sources** | **POWER\_SOURCES** | **VCC, GROUND** |
| **4.** | **Indicators** | **PROBE** | **PROBE\_GREEN, PROBE\_RED** |

**Realization of Full Adder/ Full Subtractor Using IC**

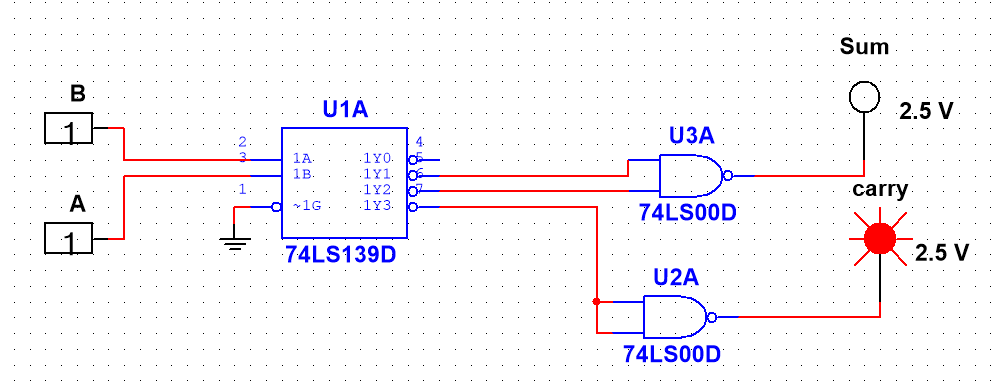
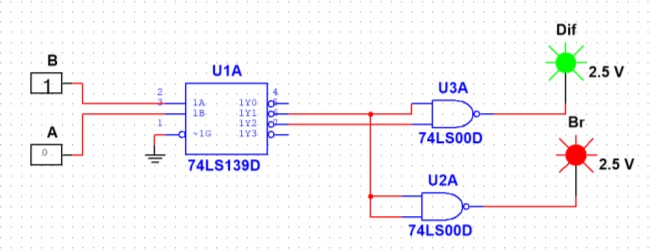
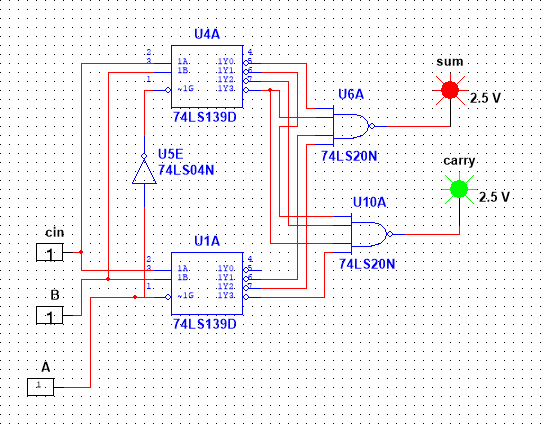
**Logic Diagram for Full Adder Logic Diagram for full Subractor using IC 74139**

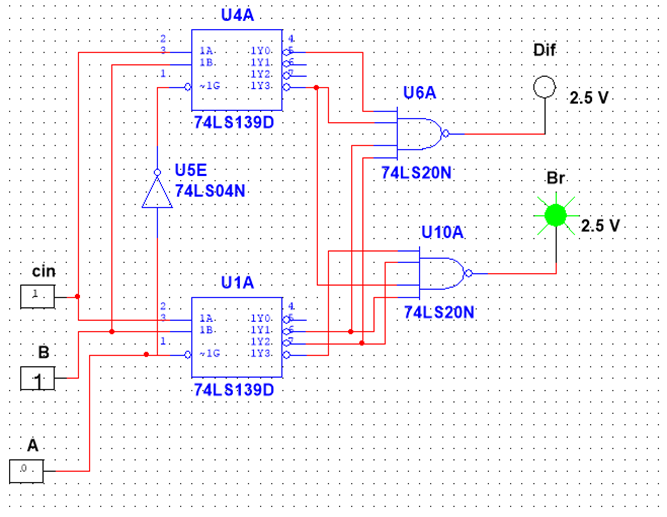
****

**Truth Table:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **Cin/Bin** | **Sum** | **Cout** | **Difference** | **Borrow out** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **1** | **0** | **1** | **1** |
| **0** | **1** | **0** | **1** | **0** | **1** | **1** |
| **0** | **1** | **1** | **0** | **1** | **0** | **1** |
| **1** | **0** | **0** | **1** | **0** | **1** | **0** |
| **1** | **0** | **1** | **0** | **1** | **0** | **0** |
| **1** | **1** | **0** | **0** | **1** | **0** | **0** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** |

**Multisim Simulation**

1. ******Half Adder 2.Half Subtractor**
2. **Full Adder Full Subtractor**

****

## Aim: Realize Binary to Gray code conversion and vice versa using IC74154(2-4 decoder).

## Component required:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.No** | **Group** | **Family** | **component** |
| **1.** | **TTL** | **74STD** | **74LS154,74LS30** |
| **2.** | **Sources** | **DIGITAL\_SOURCES** | **INTERACTIVE\_DIGITAL\_CONSTANT** |
| **3.** | **Sources** | **POWER\_SOURCES** | **VCC, GROUND** |
| **4.** | **Indicators** | **PROBE** | **PROBE\_GRENN, PROBE\_RED** |

**Theory:**

***Binary to gray code conversion*** is a very simple process. There are several steps to do this types of conversions. Steps given below elaborate on the idea on this type of conversion.

* The M.S.B. of the gray code will be exactly equal to the first bit of the given binary number.
* Now the second bit of the code will be exclusive-or of the first and second bit of the given binary number, i.e if both the bits are same the result will be 0 and if they are different the result will be 1.
* The third bit of gray code will be equal to the exclusive -or of the second and third bit of the given binary number. Thus the **Binary to gray code conversion** goes on. One example given below can make your idea clear on this type of conversion.

**Truth Table: Binary to Gray code conversion**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Decimal** | **Binary Inputs** | | | | **Gray Outputs** | | | |
| **B3** | **B2** | **B1** | **B0** | **G3** | **G2** | **G1** | **G0** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **1** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **1** |
| **2** | **0** | **0** | **1** | **0** | **0** | **0** | **1** | **0** |
| **3** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **0** |
| **4** | **0** | **1** | **0** | **0** | **0** | **1** | **1** | **0** |
| **5** | **0** | **1** | **0** | **1** | **0** | **1** | **1** | **1** |
| **6** | **0** | **1** | **1** | **0** | **0** | **1** | **0** | **1** |
| **7** | **0** | **1** | **1** | **1** | **0** | **1** | **0** | **0** |
| **8** | **1** | **0** | **0** | **0** | **1** | **1** | **0** | **0** |
| **9** | **1** | **0** | **0** | **1** | **1** | **1** | **0** | **1** |
| **10** | **1** | **0** | **1** | **0** | **1** | **1** | **1** | **1** |
| **11** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **0** |
| **12** | **1** | **1** | **0** | **0** | **1** | **0** | **1** | **0** |
| **13** | **1** | **1** | **0** | **1** | **1** | **0** | **1** | **1** |
| **14** | **1** | **1** | **1** | **0** | **1** | **0** | **0** | **1** |
| **15** | **1** | **1** | **1** | **1** | **1** | **0** | **0** | **0** |

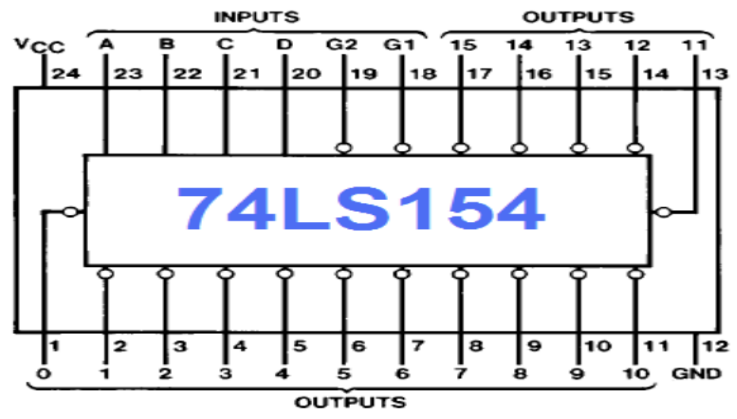
**Design Equations:**

**G3 = Σm (8,9,10,11,12,13,14,15) G2 = Σm (4,5,6,7,8,9,10,11)**

**G1 = Σm (2,3,4,5,10,11,12,13) G0 = Σm (1,2,5,6,9,10,13,15)**

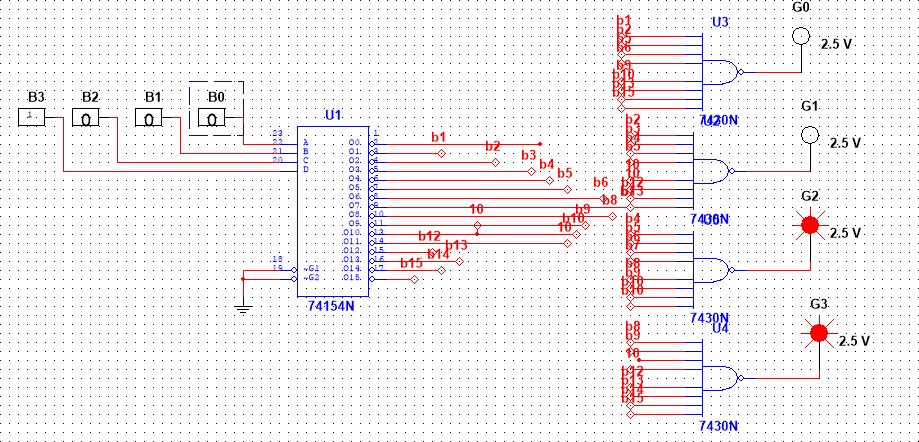
**74154 Pin diagram, Truth table and Logic symbol:**

4-line-to-16-line decoders utilizes TTL circuitry to decode four binary-coded inputs into one of sixteen mutually exclusive outputs when both the strobe inputs, G1 and G2, are low. The demultiplexing function is performed by using the 4 input lines to address the output line, passing data from one of the strobe inputs with the other strobe input low. When either strobe input is high, all outputs are high. These DE multiplexers are ideally suited for implementing high-performance memory decoders.

****

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Inputs** | | | | | | **Outputs** | | | | | | | | | | | | | | | |
| **G1** | **G2** | **D** | **C** | **B** | **A** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **0** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **0** | **1** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **1** | **0** | **0** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **1** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **1** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **1** | **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** |
| **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** |
| **0** | **0** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** |
| **0** | **0** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | **1** |
| **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** |
| **0** | **1** | **X** | **X** | **X** | **X** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **Inputs** | | | | | | **Outputs** | | | | | | | | | | | | | | | |
| **G1** | **G2** | **D** | **C** | **B** | **A** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **1** | **0** | **X** | **X** | **X** | **X** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **X** | **X** | **X** | **X** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |

**Multisim Binary to Gray Circuit Simulation**

****

***Gray code to binary conversion***

Following steps can make your idea clear on this type of conversions.

* The M.S.B of the binary number will be equal to the M.S.B of the given gray code.
* Now if the second gray bit is 0 the second binary bit will be same as the previous or the first bit. If the gray bit is 1 the second binary bit will alter. If it was 1 it will be 0 and if it was 0 it will be 1.
* This step is continued for all the bits to do **Gray code to binary conversion**

**Truth Table: Gray to Binary code conversion**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Decimal** | **Gray Inputs** | | | | **Binary Outputs** | | | |
| **G3** | **G2** | **G1** | **G0** | **B3** | **B2** | **B1** | **B0** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **1** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **1** |
| **2** | **0** | **0** | **1** | **0** | **0** | **0** | **1** | **1** |
| **3** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **0** |
| **4** | **0** | **1** | **0** | **0** | **0** | **1** | **1** | **1** |
| **5** | **0** | **1** | **0** | **1** | **0** | **1** | **1** | **0** |
| **6** | **0** | **1** | **1** | **0** | **0** | **1** | **0** | **0** |
| **7** | **0** | **1** | **1** | **1** | **0** | **1** | **0** | **1** |
| **8** | **1** | **0** | **0** | **0** | **1** | **1** | **1** | **1** |
| **9** | **1** | **0** | **0** | **1** | **1** | **1** | **1** | **0** |
| **10** | **1** | **0** | **1** | **0** | **1** | **1** | **0** | **0** |
| **11** | **1** | **0** | **1** | **1** | **1** | **1** | **0** | **1** |
| **12** | **1** | **1** | **0** | **0** | **1** | **0** | **0** | **0** |
| **13** | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **1** |
| **14** | **1** | **1** | **1** | **0** | **1** | **0** | **1** | **1** |
| **15** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **0** |

**Design Equations:**

**B3 = Σm (8,9,10,11,12,13,14,15) B2 = Σm (4,5,6,7,8,9,10,11)**

**B1 = Σm (2,3,4,5,8,9,14.15) B0 = Σm (1,2,4,7,8,11,13,14)**

**Multisim Gray to Binary Ciruit Diagram**

**Note:** The multi sim circuit can designed similar to Binary to Gray with relevant expressions.

**Procedure:**

* 1. Place the components as tabulated in component list.
  2. Rig up the connections as shown in the circuit diagram.
  3. Simulate the design.
  4. Verify with respect to the truth table given.

**Result:**

Realised Adder and subtractor using IC74139

Realised Binary to Gray and Vice versa using IC 74154

**EXPERIMENT 7**

**SR, Master-Slave JK, D & T flip-flops using NAND Gates using Pspice/Multisim.**

**Aim:** To realize SR, Master-Slave JK, D and T Flip-Flops using NAND Gates

**Components:**

**SR Flip Flop:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No.** | **Group** | **Family** | **Components** | **Quantity** |
| **1.** | **TTL** | **74LS** | **IC7400** | **4** |
| **2.** | **Indicators** | **PROBE** | **PROBE\_BLUE,PROBE\_RED** | **2** |
| **3.** |  |  | **Multisim Software** |  |

**Master-Slave JK Flip Flop:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No.** | **Group** | **Family** | **Components** | **Quantity** |
| **1.** | **TTL** | **74LS** | **IC7400** | **4** |
| **2.** | **TTL** | **74LS** | **IC7410** | **5** |
| **3.** | **Indicators** | **PROBE** | **PROBE\_BLUE,PROBE\_RED** | **2** |
| **4.** |  |  | **Multisim Software** |  |

**D Flip Flop:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No.** | **Group** | **Family** | **Components** | **Quantity** |
| **1.** | **TTL** | **74LS** | **IC7400** | **4** |
| **2.** | **TTL** | **74LS** | **IC7410** | **6** |
| **3.** | **Indicators** | **PROBE** | **PROBE\_BLUE,PROBE\_RED** | **2** |
| **4.** |  |  | **Multisim Software** |  |

**T Flip Flop:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No.** | **Group** | **Family** | **Components** | **Quantity** |
| **1.** | **TTL** | **74LS** | **IC7400** | **4** |
| **2.** | **TTL** | **74LS** | **IC7410** | **5** |
| **3.** | **Indicators** | **PROBE** | **PROBE\_BLUE,PROBE\_RED** | **2** |
| **4.** |  |  | **Multisim Software** |  |

**Theory:**

Basically Flip-Flops are the bistable multivibrators that stores logic 1 and logic 0.Shift registers, memory, and counters are built by using Flip – Flops. Any complex sequential machines are build using Flip – Flops. Sequential circuit (machine) output depends on the present state and input applied at that instant.

JK flip Flop is the most widely used of all the flip-flop designs and is considered to be a universal flip-flop circuit. The sequential operation of the JK flip flop is exactly the same as for the previous SR flip-flop with the same “Set” and “Reset” inputs. The difference this time is that the “JK flip flop” has no invalid or forbidden input states of the SR Latch even when S and R are both at logic “1”.

The **JK flip flop** is basically a gated SR flip-flop with the addition of a clock input circuitry that prevents the illegal or invalid output condition that can occur when both inputs S and R are equal to logic level “1”. Due to this additional clocked input, a JK flip-flop has four possible input combinations, “logic 1”, “logic 0”, “no change” and “toggle”.

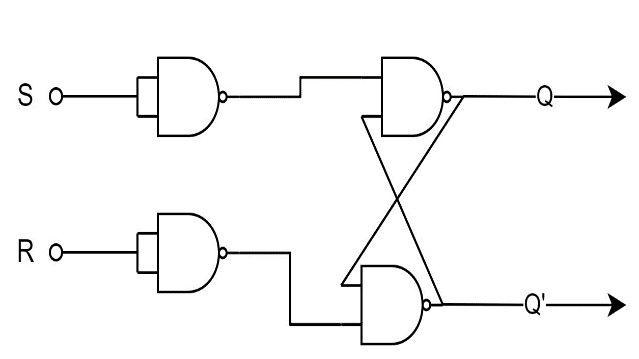
A **D flip flop** has a single data input. This type of FF is obtained from the SR FF by connecting the R input through an inverter, and the S input is connected directly to data input. The modified clocked SR flip-flop is known as D-flip-flop and is shown below. From the truth table of SR flip-flop we see that the output of the SR flip-flop is in unpredictable state when the inputs are same and high.

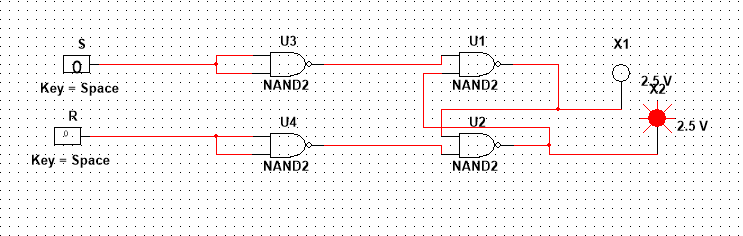
**T flip-flop** is known as toggle flip-flop. The T flip-flop is modification of the J-K flip-flop. Both the JK inputs of the JK flip – flop are held at logic 1 and the clock signal continuous to change.

**Logic Diagram:**

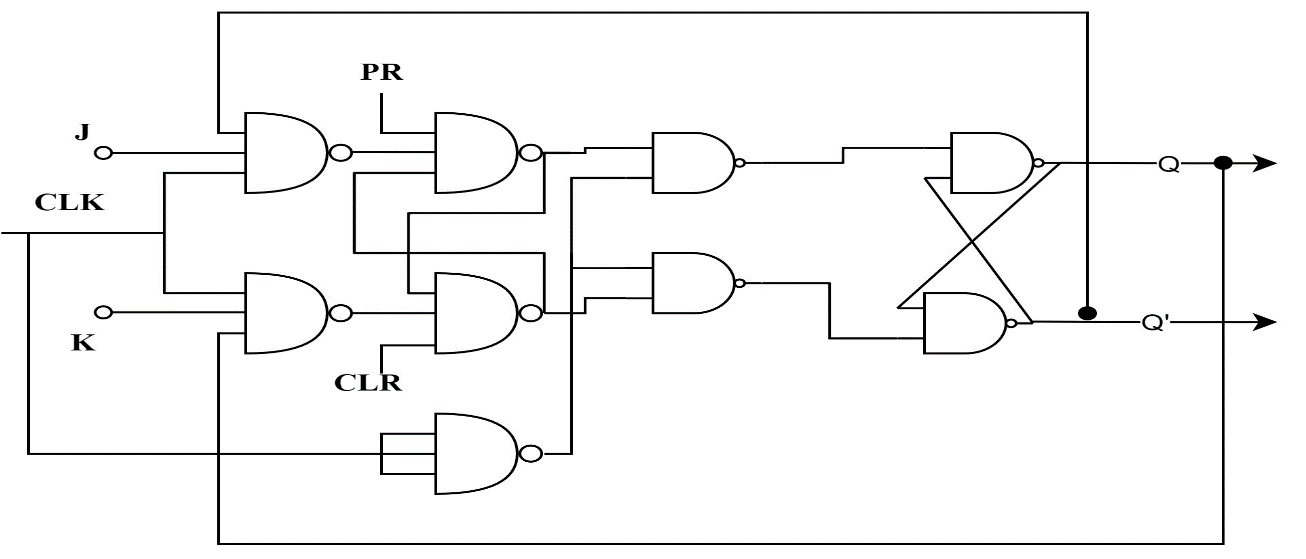
**SR Flip Flop Truth Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Inputs** | | **Outputs** | | **Comments** |
| **S** | **R** | **Q** | **Q\_bar** |  |
| **0** | **0** | **0** | **0** | **Previous State** |
| **0** | **0** | **1** | **1** | **Previous State** |
| **0** | **1** | **0** | **0** | **Previous State** |
| **0** | **1** | **1** | **0** | **Set** |
| **1** | **0** | **0** | **1** | **Reset** |
| **1** | **0** | **1** | **1** | **Previous State** |
| **1** | **1** | **0** | **-** | **Indeterminate** |
| **1** | **1** | **1** | **-** | **Indeterminate** |

****



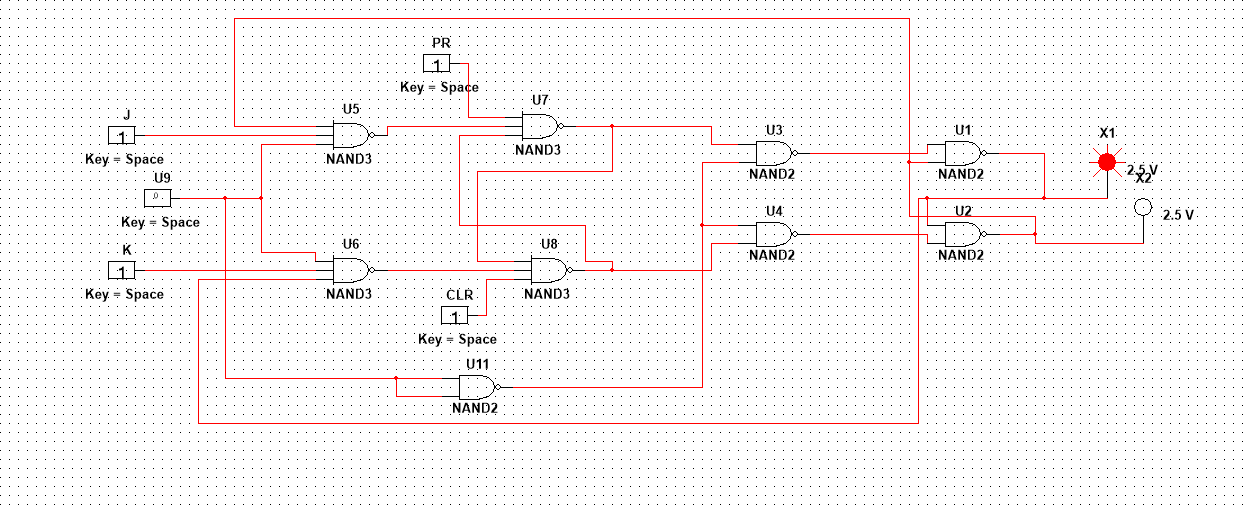
**Multisim Circuit Diagram:**

**Master Slave JK Flip Flop**

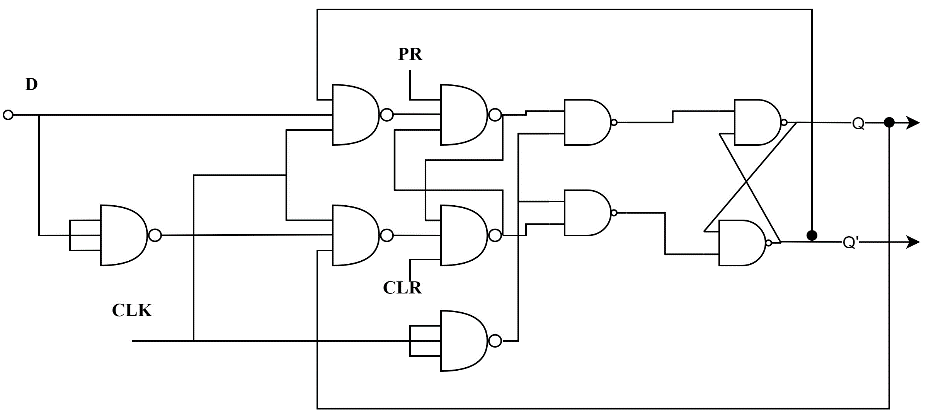
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PR** | **CLR** | **Inputs** | | | **Outputs** | | **Comments** |
| **J** | **K** | **Clock** | **Qn+1** | **Q\_barn+1** |
| **0** | **1** | **X** | **X** | **X** | **1** | **0** | **Set** |
| **1** | **0** | **X** | **X** | **X** | **0** | **1** | **Reset** |
| **1** | **1** | **0** | **0** |  | **Qn** | **Q\_barn** | **No Change** |
| **1** | **1** | **0** | **1** |  | **0** | **1** | **Reset** |
| **1** | **1** | **1** | **0** |  | **1** | **0** | **Set** |
| **1** | **1** | **1** | **1** |  | **Q\_barn** | **Qn** | **Toggle** |

**Truth Table**

**Multisim Circuit Diagram:**

****

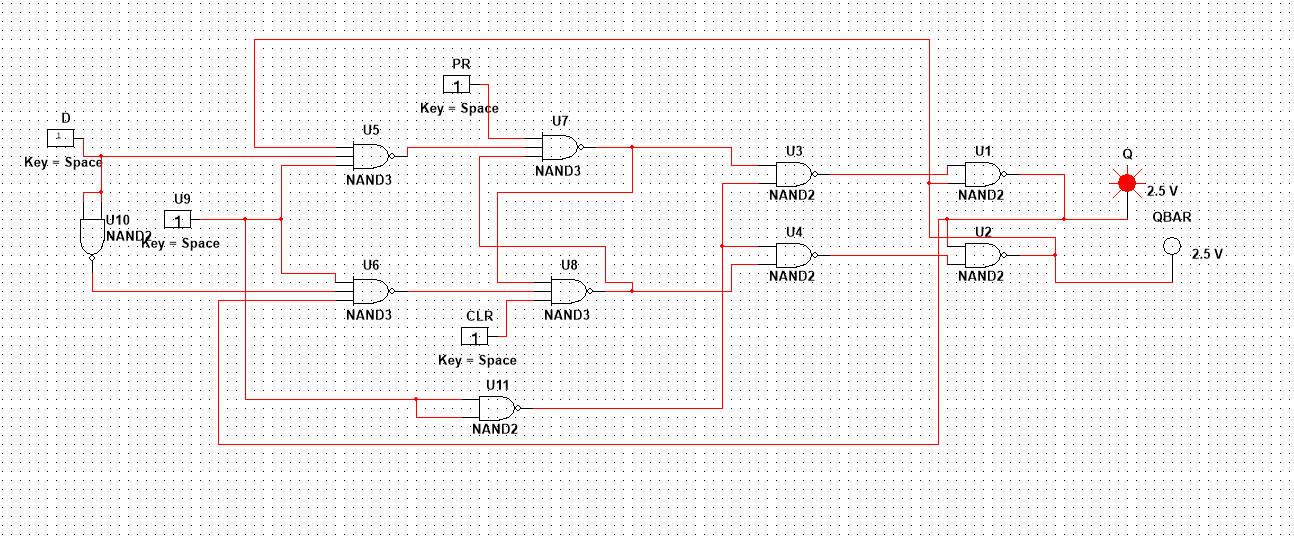
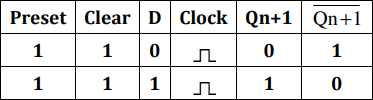
**D Flip-Flop:**

****

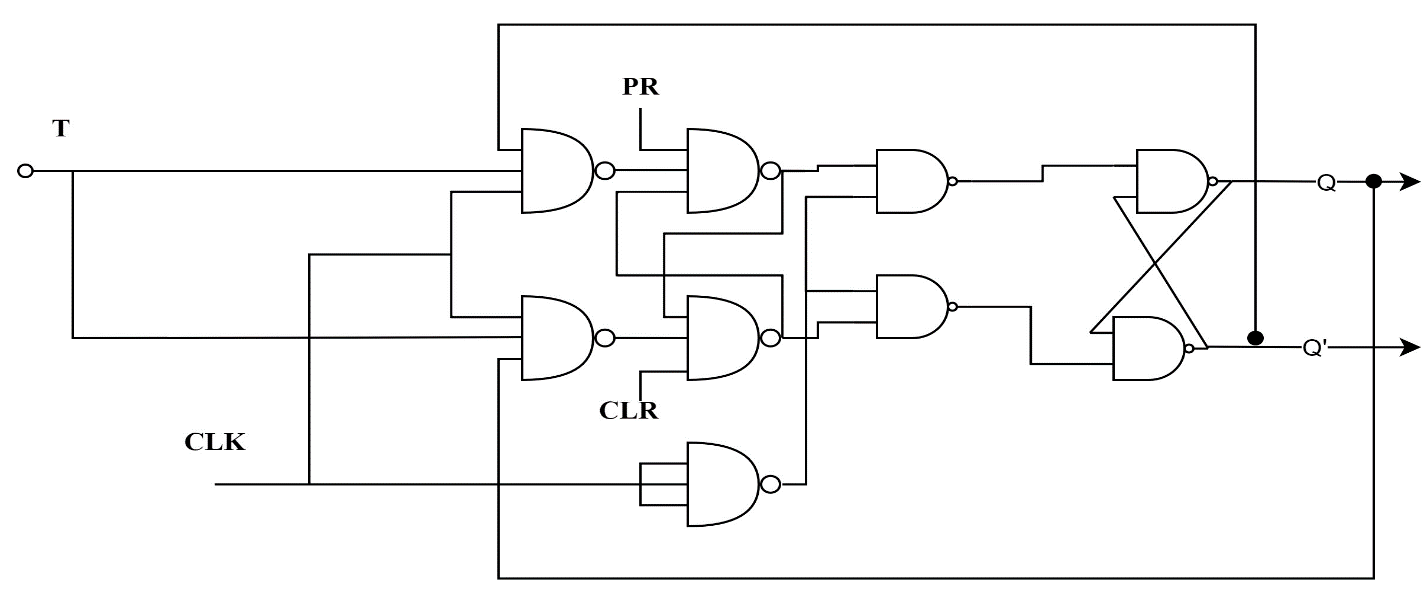
**Truth Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PR** | **CLR** | **Inputs** | | **Outputs** | |
| **D** | **Clock** | **Qn+1** | **Q\_barn+1** |
| **1** | **1** | **0** |  | **0** | **1** |
| **1** | **1** | **1** |  | **1** | **0** |

**Multisim Circuit Diagram:**



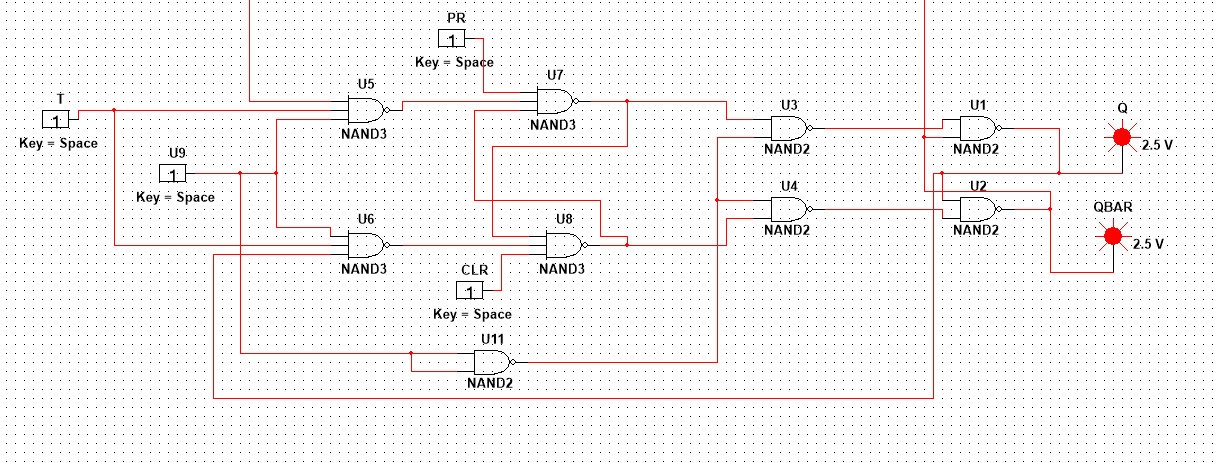
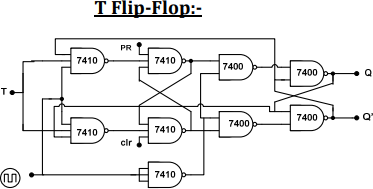
**T-Flip Flop:**

****

**Truth Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PR** | **CLR** | **Inputs** | | **Outputs** | |
| **T** | **Clock** | **Qn+1** | **Q\_barn+1** |
| **1** | **1** | **0** |  | **Qn** | **Q\_barn** |
| **1** | **1** | **1** |  | **Q\_barn** | **Qn** |

**Multisim Circuit Diagram:**



**Result:**

Realized SR Flip flop, Master-Slave JK Flip-Flop, D-Flip flop and T-Flip flop verified using NAND gates

**EXPERIMENT 8**

**Design and realize the Synchronous counters (up/down decade/binary) using PSpice/Multisim**

**Aim:** To Design Synchronous Counters using IC7490 (Decade) and IC74192 (Up/down binary)

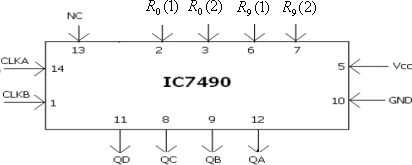
**Components:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No.** | **Group** | **Family** | **Components** | **Quantity** |
| **1.** | **TTL** | **74LS** | **IC7490** | **4** |
| **2.** | **TTL** | **74LS** | **IC74192** |  |
| **3.** | **Indicators** | **PROBE** | **PROBE\_BLUE,PROBE\_RED** | **2** |
| **4.** |  |  | **Multisim Software** |  |

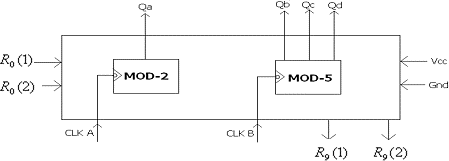
**Decade Counter using IC7490**

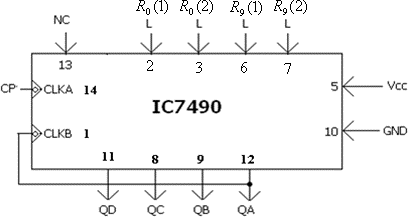
**Theory**:

Counter is a sequential circuit. A digital circuit which is used for a counting pulses is known counter. Counter is the widest application of flip-flops. It is a group of flip-flops with a clock signal applied. If the "clock" pulses are applied to all the flip-flops in a counter simultaneously, then such a counter is called as synchronous counter.

**Pin Diagram:**

**Internal Diagram:**



**IC 7490 as BCD/Decade Counter**

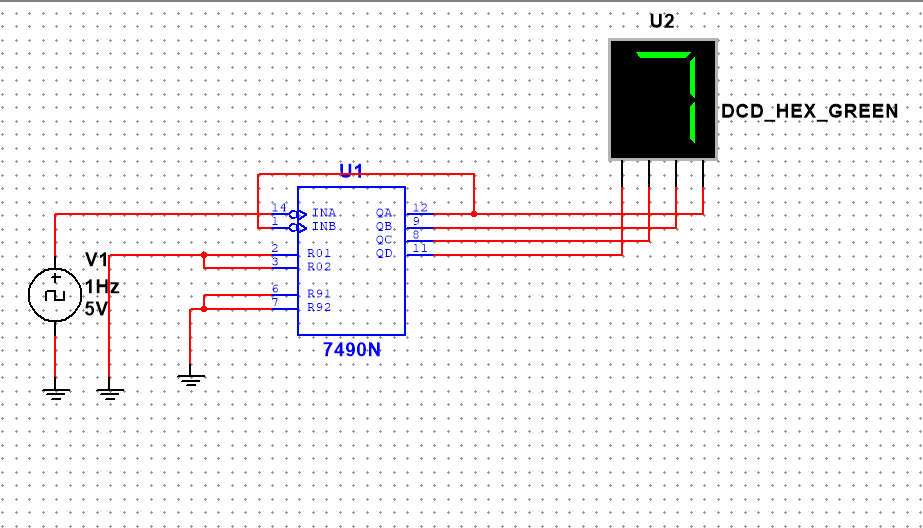
**Components:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.No** | **Group** | **Family** | **Component** |
| **1.** | **TTL** | **74STD** | **7490N** |
| **2.** | **Sources** | **SIGNAL VOLTAGE SOURCES** | **CLOCK VOLTAGE** |
| **3.** | **Sources** | **POWER SOURCES** | **GROUND** |
| **4.** | **Indicators** | **HEX\_DISPLAY** | **DCD\_HEX** |

**Conditional Table:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **R0(1)** | **R0(2)** | **R9(1)** | **R9(2)** | **Qa** | **Qb** | **Qc** | **Qd** |
| **H** | **H** | **L** | **X** | **L** | **L** | **L** | **L** |
| **H** | **H** | **X** | **L** | **L** | **L** | **L** | **L** |
| **X** | **L** | **H** | **H** | **1** | **0** | **0** | **1** |
| **L** | **X** | **L** | **X** | **MOD-2 COUNTER** | | | |
| **X** | **L** | **X** | **L** | **MOD-5 COUNTER** | | | |

**Multisim Circuit Diagram:**

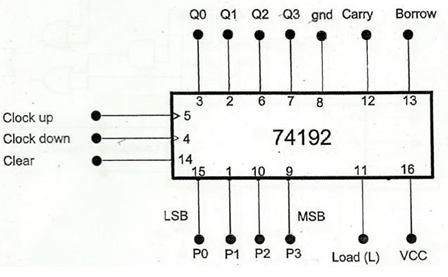


**Up/Down Binary Counter using IC74192:**

**Theory:**

The 74192 is a Presettable Synchronous 4-Bit Up/Down Decade Counter. Presetting the counter to the number on the preset data inputs (Input A - Input D) is accomplished by a LOW asynchronous parallel load input (Load). The counter is incremented on the low-to-high transition of the UP input (and a high level on the Clock- DOWN) and decremented on the low to high transition of the DOWN input (and a high level on the UP input). A high level on the CLR input overrides any other input to clear the counter to its zero state. The Terminal Count up (CO) goes low half a clock period before the zero count is reached and returns to a high level at the zero count.

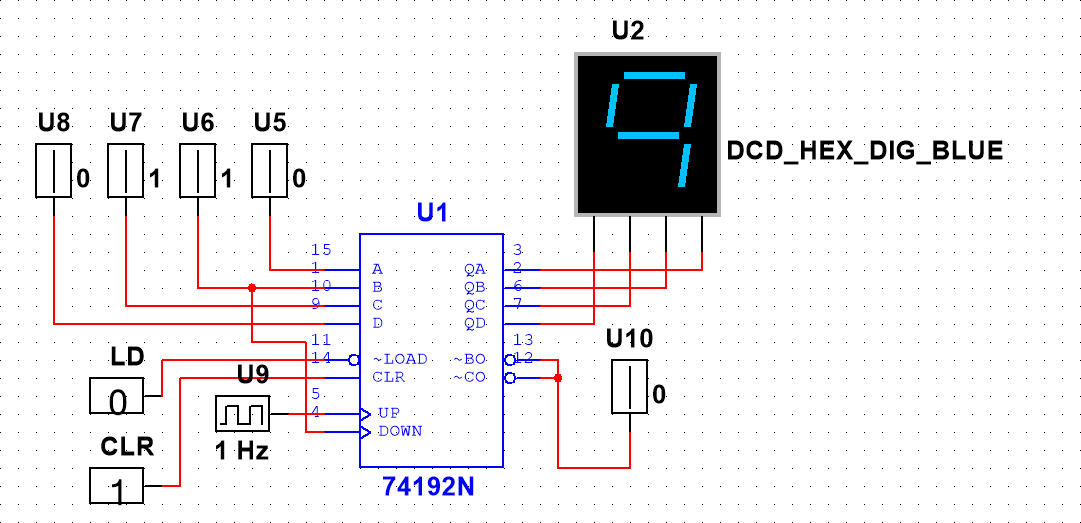
**Pin Diagram:**



**Function Table:**

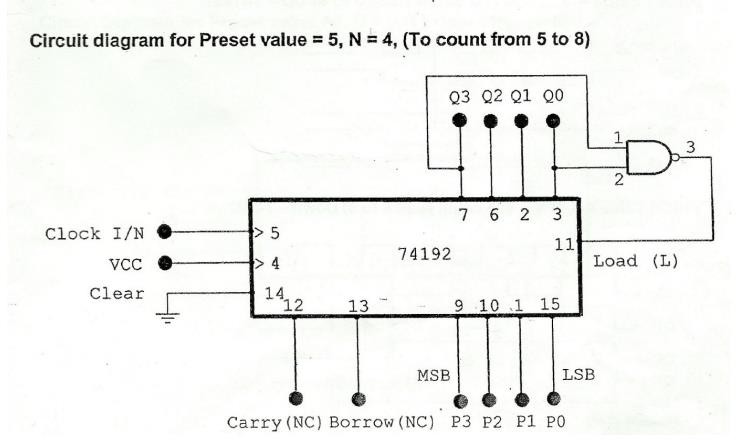
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LOAD** | **CLEAR** | **CLK\_UP** | **CLK\_DOWN** | **MODE** |
| **X** | **1** | **X** | **X** | **Reset to Zero** |
| **1** | **0** |  | **1** | **Up Count** |
| **1** | **0** | **1** |  | **Down Count** |
| **0** | **0** | **X** | **X** | **Preset** |
| **1** | **0** | **1** | **1** | **Stop Count** |

**Multisim Circuit Diagram:**



**NOTE:**

For down count connect **clock** to **pin 4** and **pin 5** to **logic ‘1’**, Same circuit can be configured to work as **mod-4 up counter** according to the below circuit diagram.



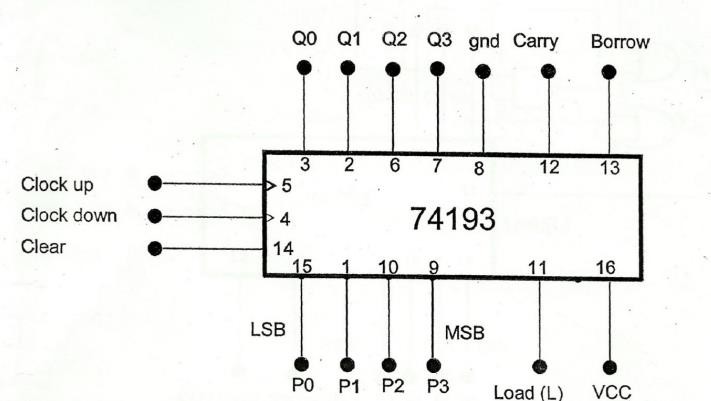
**Up/down binary counter using IC74193**

**Theory:**

The 74193 is a Presettable Synchronous 4-Bit Up/Down Counter( same as 74192 the only difference is it is mod-15 where as 74192 is of mod-10). Presetting the counter to the number on the preset data inputs (Input A - Input D) is accomplished by a LOW asynchronous parallel load input (Load). The counter is incremented on the low-to-high transition of the UP input (and a high level on the Clock- DOWN) and decremented on the low to high transition of the DOWN input (and a high level on the UP input). A high level on the CLR input overrides any other input to clear the counter to its zero state. The Terminal Count up (CO) goes low half a clock period before the zero count is reached and returns to a high level at the zero count.

**Pin Diagram: Function Table:**

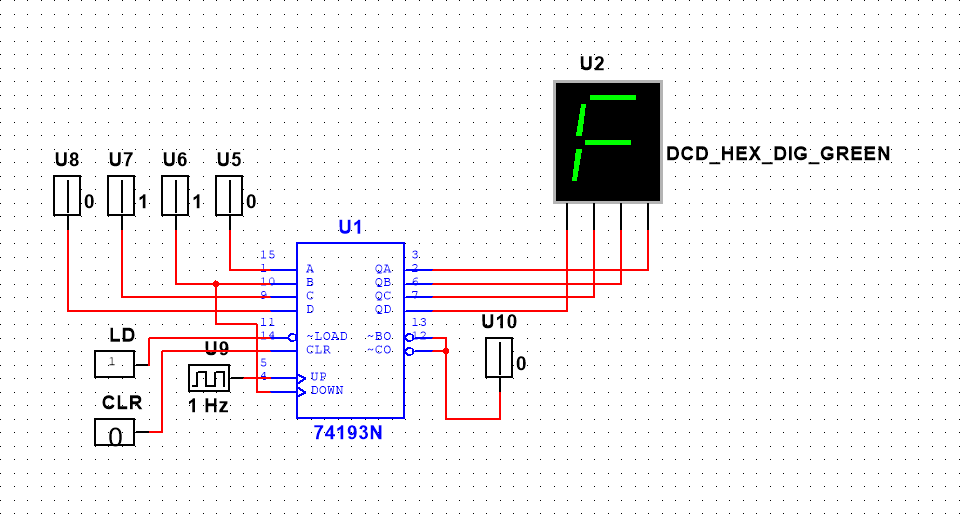
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LOAD** | **CLEAR** | **CLK\_UP** | **CLK\_DOWN** | **MODE** |
| **X** | **1** | **X** | **X** | **Reset to Zero** |
| **1** | **0** |  | **1** | **Up Count** |
| **1** | **0** | **1** |  | **Down Count** |
| **0** | **0** | **X** | **X** | **Preset** |
| **1** | **0** | **1** | **1** | **Stop Count** |



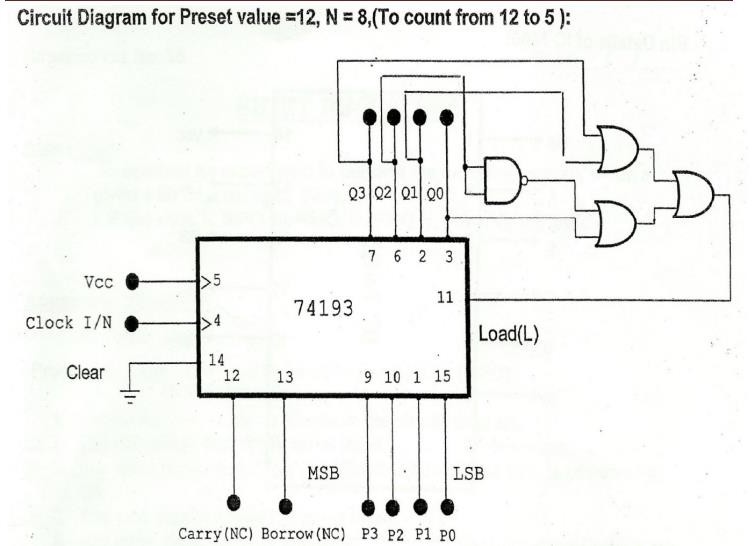
**Components:**

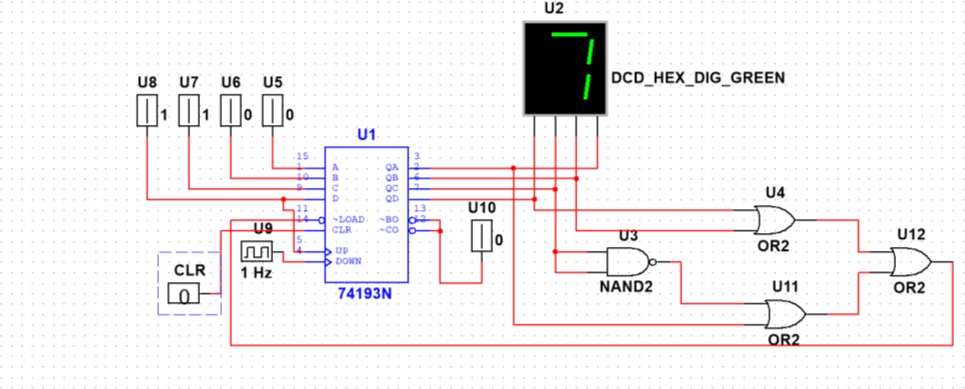
|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.No** | **Group** | **Family** | **Component** |
| **1.** | **TTL** | **74STD** | **74192,74193** |
| **2.** | **Sources** | **DIGITAL SOURCES** | **INTERACTIVE\_DIGITAL\_CONSTANT** |
| **3.** | **Sources** | **DIGITAL SOURCES** | **DIGITAL\_CLOCK** |
| **4.** | **Indicators** | **PROBE** | **DCD\_HEX\_DIG\_GREEN** |

**Multisim Circuit Diagram:**



**Note:** For down count connect **clock** to **pin 4** and **pin 5** to **logic ‘1’**



**Multisim Circuit Diagram:**

**Procedure:**

1. Place the components as tabulated in component list.
2. Rig up the connections as shown in the circuit diagram.
3. Simulate the design.
4. Verify with respect to the truth table given.

**EXPERIMENT 9**

**Realize the shift registers using IC 7474/7495**

**Aim:** To realize the shift registers and their modes (i) SISO (ii) PIPO (iii) PISO (iv) SIPO using IC 7474/7495 using PSpice/Multisim.

##### **Learning Objectives:**

* 1. To illustrate the operation of shift registers
  2. To study different shift register configurations

**Components:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | **Group** | **Family** | **Component** | **Quantity** |
| **1.** | **TTL** | **74STD** | **74194** | **1** |
| **2.** | **Sources** | **DIGITAL SOURCES** | **INTERACTIVE\_DIGITAL\_CONSTANT** | **9** |
| **3.** | **Sources** | **DIGITAL SOURCES** | **DIGITAL\_CLOCK** | **1** |
| **4.** | **Indicators** | **PROBE** | **PROBE\_BLUE,PROBE\_RED** | **4** |

**Theory:**

• Shift registers are a type of sequential logic circuit, mainly for storage of digital data. They are a group of flip- flops connected in a chain so that the output from one flipflop becomes the input of the next flip-flop. All the flip-flops are driven by a common clock, and all are set or reset simultaneously.

• Different types of shift register are Serial in serial out shift register, Serial in parallel out shift register, Parallel in serial out shift register, Parallel in parallel out shift register, Bidirectional shift register shift register.

**Multisim Procedure:**

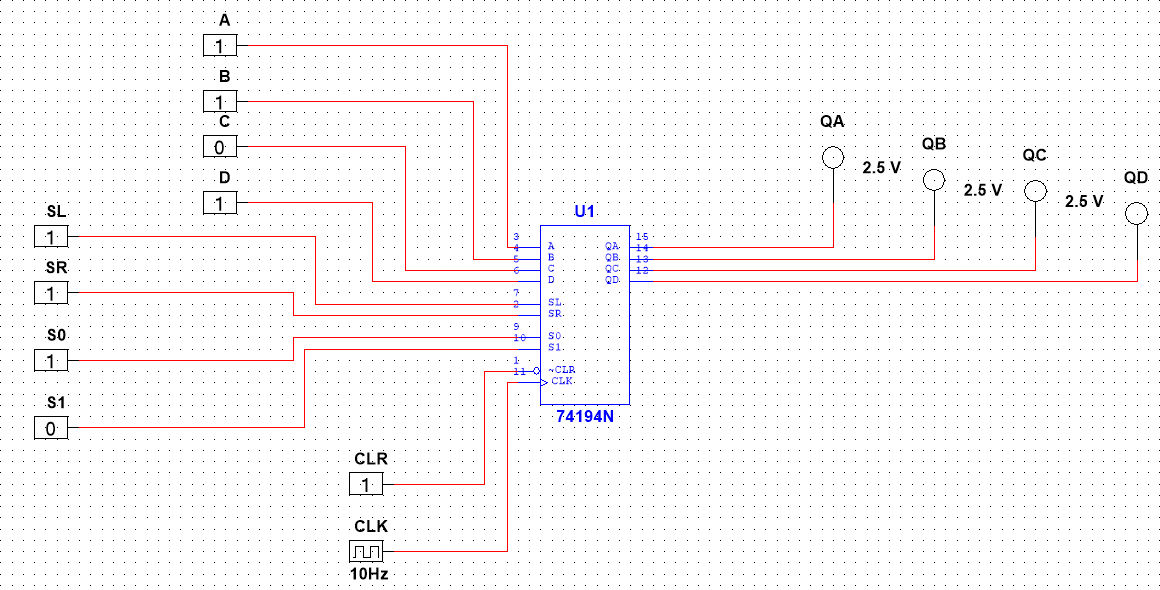
1. To Place Components click on Place/Components. On the Select Component window, click on Group-> All groups. Type the components needed for the circuit. Click OK to place the component on the schematic.

2. Place component 74194 IC from the component list.

3. Place the component interactive\_digital\_constant & Digital\_clock from the component list for inputs.

4. Select Probe from component list and place at outputs.

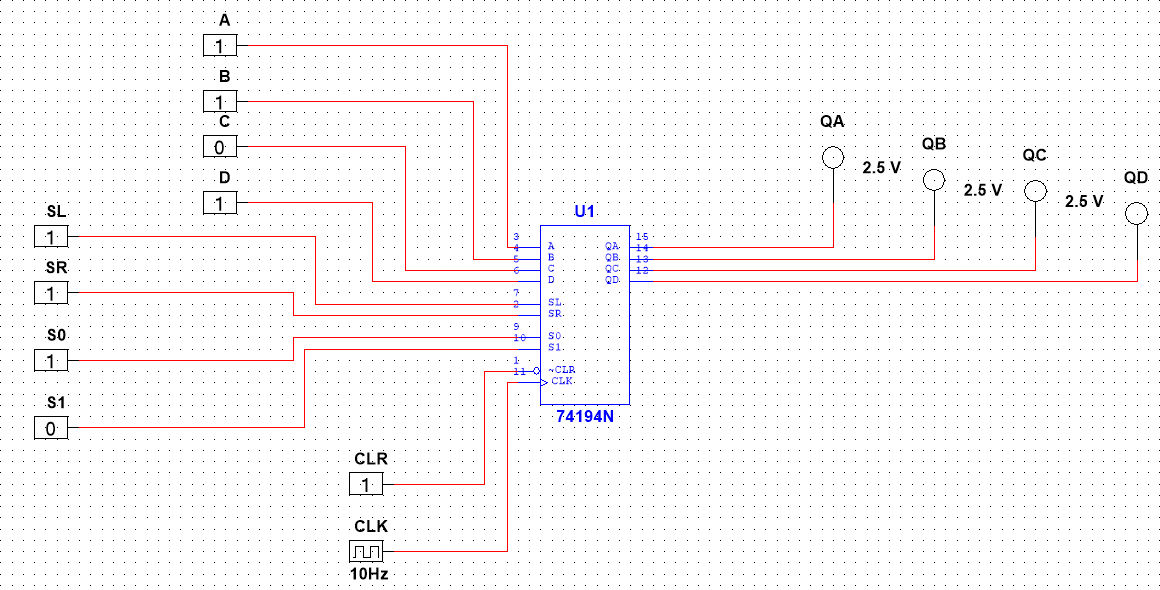
5. Once the components are placed, wire the circuit by clicking on Place/Wire drag and place the wire. Components can also be connected by clicking the mouse over the terminal edge of one component and dragging to the edge of another component.

**Multisim simulation: Serial In Serial Out [SISO]:**

**Truth Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CLR** | **SL** | **SR(Serial Input)** | **S0** | **S1** | **QA Serial Right Shift (Serial Out)** |
| **1** | **1** | **1** | **1** | **0** | **1** |
| **1** | **1** | **0** | **1** | **0** | **0** |
| **1** | **1** | **1** | **1** | **0** | **1** |
| **1** | **1** | **0** | **1** | **0** | **0** |

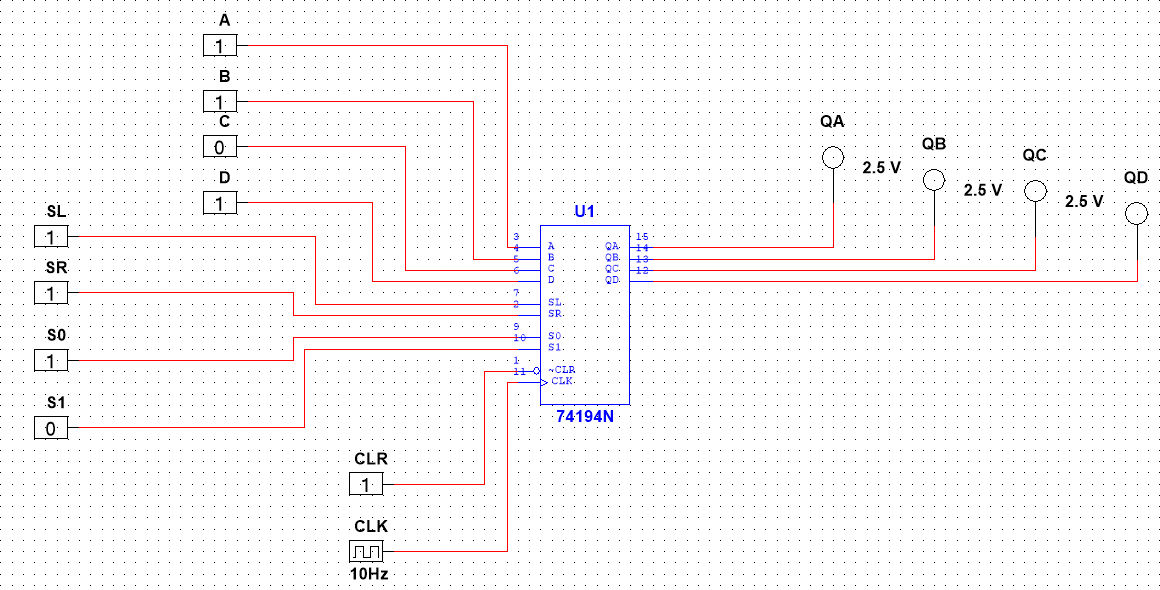
**Parallel In Parallel Out [PIPO]:**

****

**Truth Table:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CLR** | **SL** | **SR(Serial Input)** | **S0** | **S1** | **A** | **B** | **C** | **D** | **QA** | **QB** | **QC** | **QD** |
| **1** | **X** | **X** | **1** | **1** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** |
| **1** | **X** | **X** | **1** | **1** | **1** | **1** | **1** | **0** | **1** | **1** | **1** | **0** |

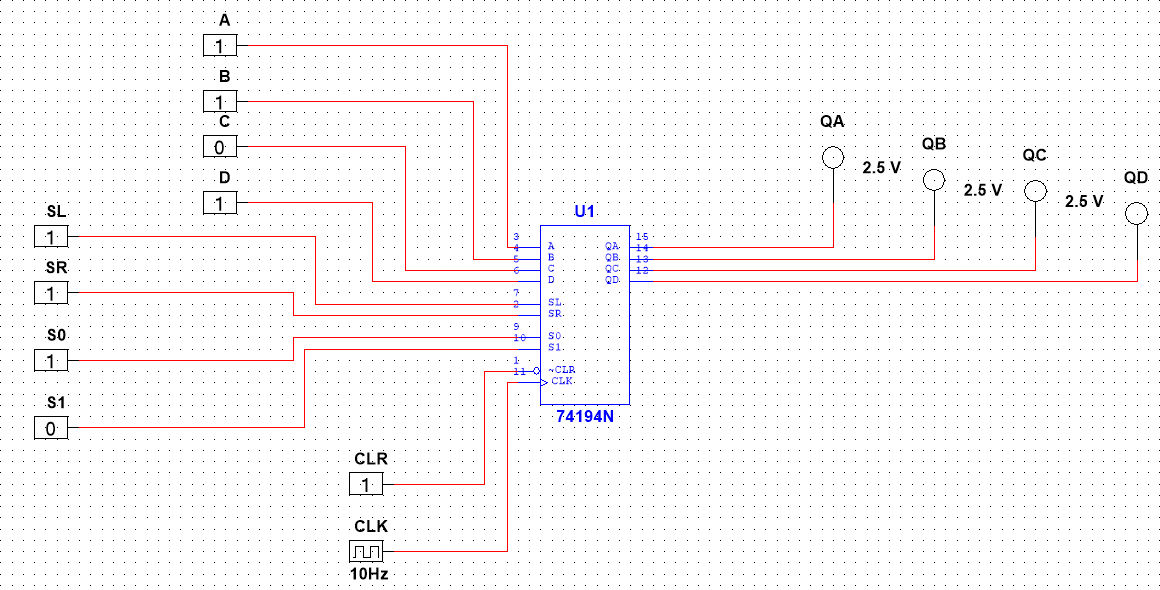
**Parallel In Serial Out [PIPO]:**

****

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CLR** | **SL** | **SR(Serial Input)** | **S0** | **S1** | **A** | **B** | **C** | **D** | **QA** | **QB** | **QC** | **QD** |
| **1** | **X** | **X** | **1** | **1** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** |
| **1** | **1** | **0** | **1** | **0** |  |  |  |  |  |  |  | **0** |
| **1** | **1** | **0** | **1** | **0** |  |  |  |  |  |  |  | **1** |
| **1** | **1** | **0** | **1** | **0** |  |  |  |  |  |  |  | **0** |
| **1** | **1** | **0** | **1** | **0** |  |  |  |  |  |  |  | **1** |

**Truth Table:**

**Serial In Parallel Out [SIPO]:**

****

**Truth Table:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CLR** | **SL** | **SR(Serial Input)** | **S0** | **S1** | **QA** | **QB** | **QC** | **QD** |
| **1** | **1** | **1** | **1** | **0** | **1** |  |  |  |
| **1** | **1** | **0** | **1** | **0** | **0** |  |  |  |
| **1** | **1** | **1** | **1** | **0** | **1** |  |  |  |
| **1** | **1** | **0** | **1** | **0** | **0** |  |  |  |
| **1** | **X** | **X** | **1** | **1** | **0** | **1** | **0** | **1** |

**Result:**

All the shift register operations are verified.

**EXPERIMENT NO. 10**

**Design Pseudo Random Sequence generator using 7495**

**Aim:** To design Pseudo Random Sequence generator using IC 7495 using Pspice/Multisim.

**Learning Objectives:**

To learn about generation of given sequence using Flip flops.

**Components Required:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | **Group** | **Family** | **Component** | **Quantity** |
| **1.** | **TTL** | **74STD** | **IC74194** | **1** |
| **2.** | **TTL** | **74LS** | **IC7486** | **1** |
| **3.** | **Sources** | **DIGITAL SOURCES** | **INTERACTIVE\_DIGITAL\_CONSTANT** | **6** |
| **4.** | **Sources** | **DIGITAL SOURCES** | **DIGITAL\_CLOCK** | **1** |
| **5.** | **Indicators** | **PROBE** | **PROBE\_BLUE,PROBE\_RED** | **4** |

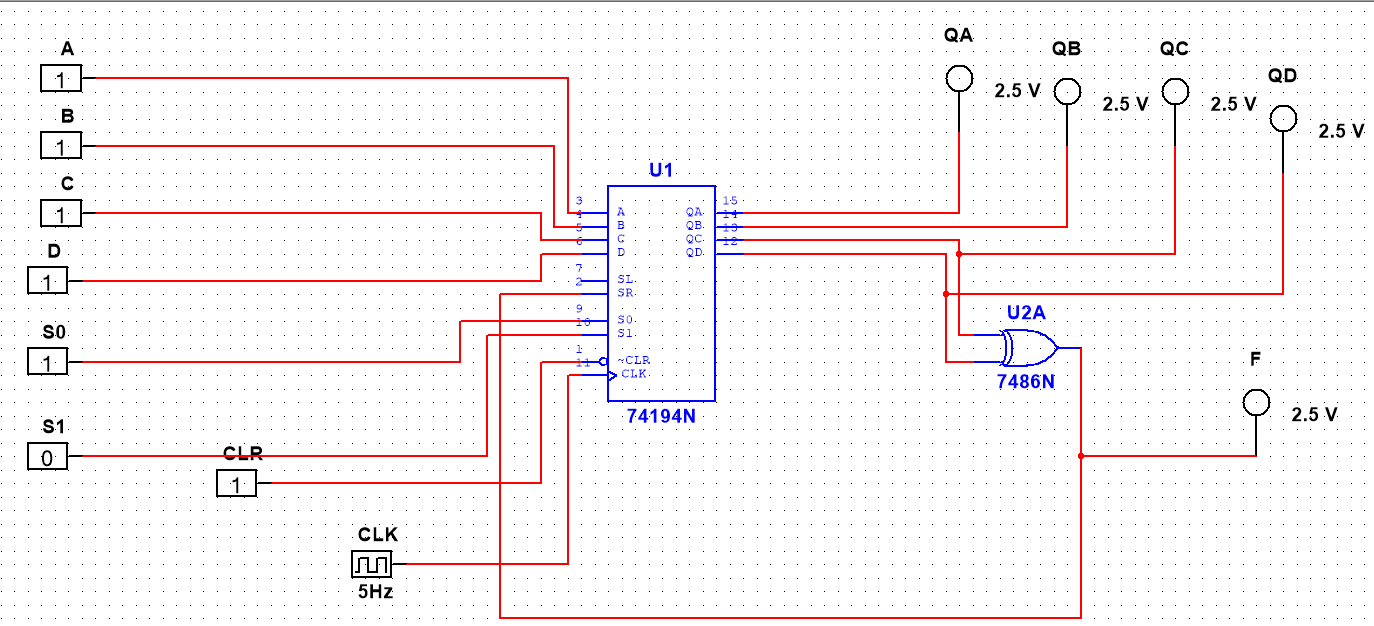
**Theory:**

The sequence generators are nothing but a set of digital circuits which are designed to result in a specific bit sequence at their output. There are several ways in which these circuits can be designed including those which are based on multiplexers and flip-flops. Here in this article we deal with the designing of sequence generator using D flip-flops (please note that even JK flip-flops can be made use of).As an example, let us consider that we intend to design a circuit which moves through the states 0-1-3-2 before repeating the same pattern.

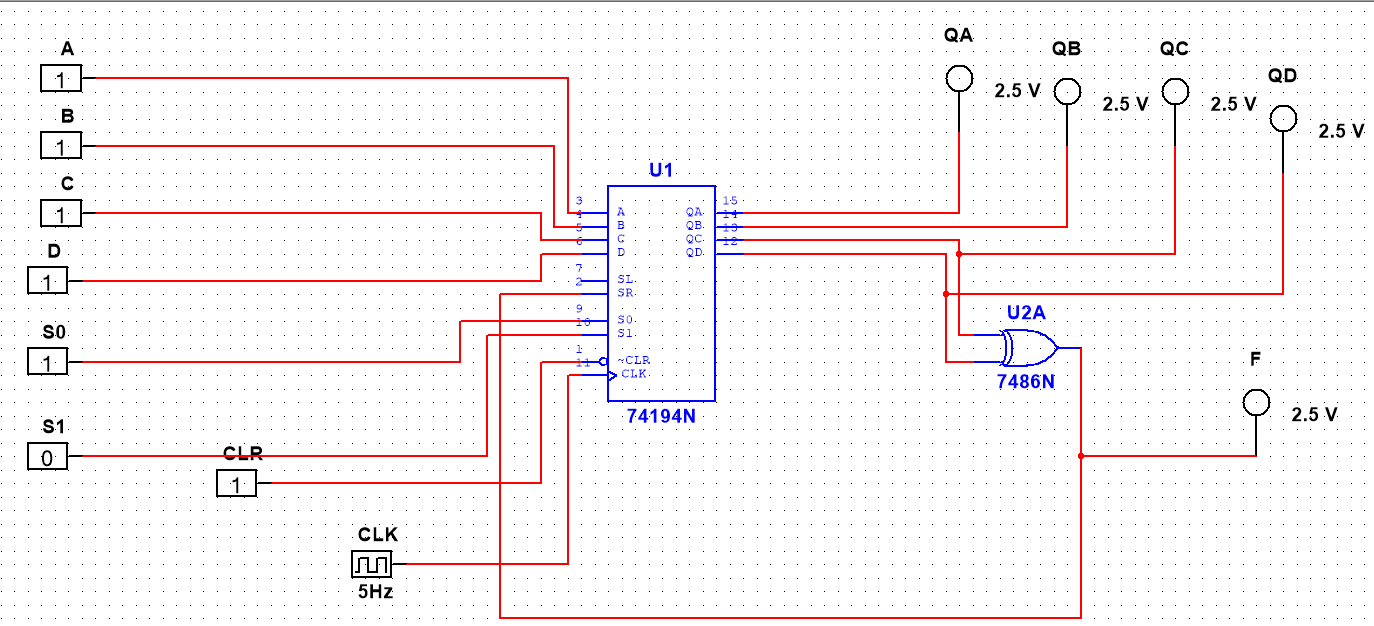
**Design:**

**Multisim simulation:**

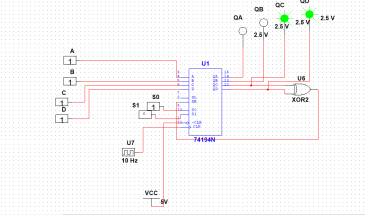
**Parallel Loading S1 =1, S0 = 1**

****

**Right Shifting S1 = 0, S0 = 1**

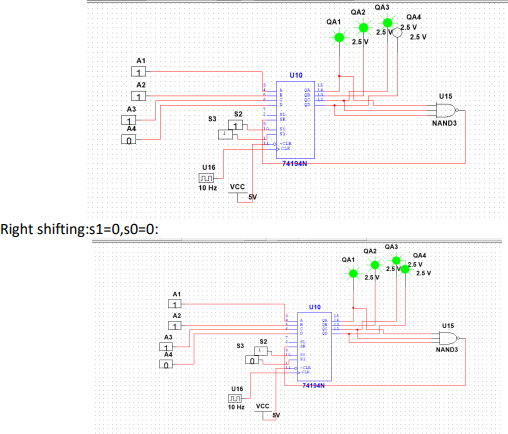
****

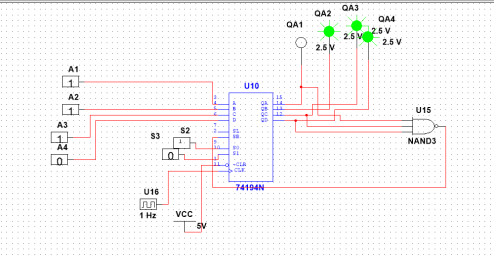
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **QA** | **QB** | **QC** | **QD** | **F** |
| **1** | **1** | **1** | **1** | **0** |
| **0** | **1** | **1** | **1** | **0** |
| **0** | **0** | **1** | **1** | **0** |
| **0** | **0** | **0** | **1** | **1** |
| **1** | **0** | **0** | **0** | **0** |
| **0** | **1** | **0** | **0** | **0** |
| **0** | **0** | **1** | **0** | **1** |
| **1** | **0** | **0** | **1** | **1** |
| **1** | **1** | **0** | **0** | **0** |
| **0** | **1** | **1** | **0** | **1** |
| **1** | **0** | **1** | **1** | **0** |
| **0** | **1** | **0** | **1** | **1** |
| **1** | **0** | **1** | **0** | **1** |
| **1** | **1** | **0** | **1** | **1** |
| **1** | **1** | **1** | **0** | **1** |



**Truth Table:**

**Multisim Procedure:**





**Result:**

Pseudo random sequence generator are designed and verified.