**ASSIGNMENT I**

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR ASSAM-788010, INDIA



Computer Architecture And Organizations

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **SUBMITTED BY**: | | | | |
|  |  |  |  |  |  |  |
|  |  | **NAME**: Swagat Shubham Bhuyan | | | | |
| **SUBMITTED TO**: | |  |  |  |  |  |
| **SCHOLAR ID**: 18-1-5-059 | | | | |
|  |  |
| Dr. Malaya Dutta Borah | |
| **SEC**: A | | |  | |
|  |  |  |  |  |  |  |

**Experiment**:

1. Ripple Carry Adder.

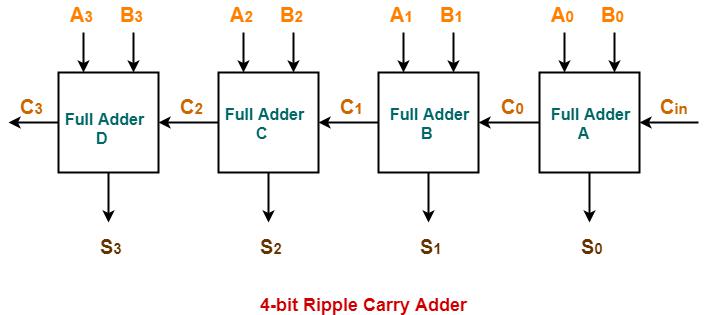
**Objectives**:

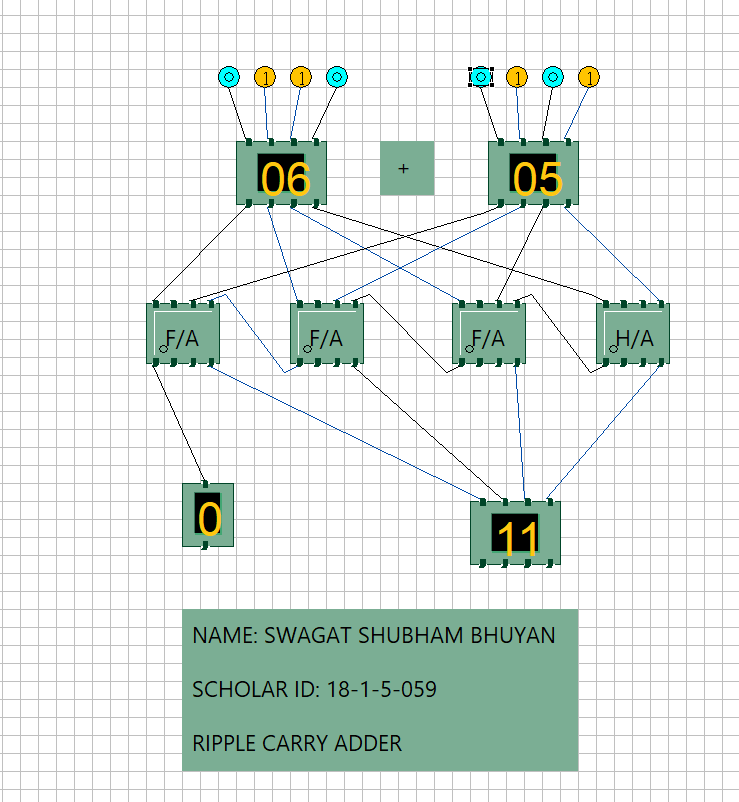
To show the addition of two 4-bit numbers using a 4-bit Ripple Carry Adder.

**Working Principle**:

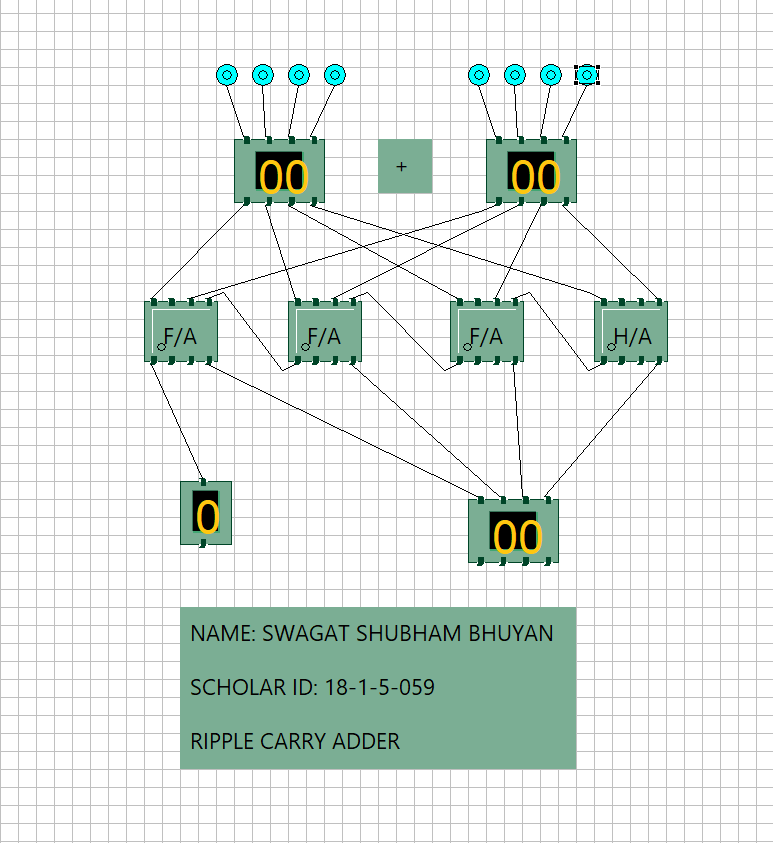
N full adders are cascaded i.e., connected in a serial manner to build an adder to add two N-bit numbers. This is called an N-bit Ripple Carry Adder. The first adder takes the last bits from both the numbers and 0 as carry-in as inputs and produces sum and carry as output. This carry acts as the carry-in for the next full adder which then produces the next sum and carry and so on.

**Circuit Diagram**:

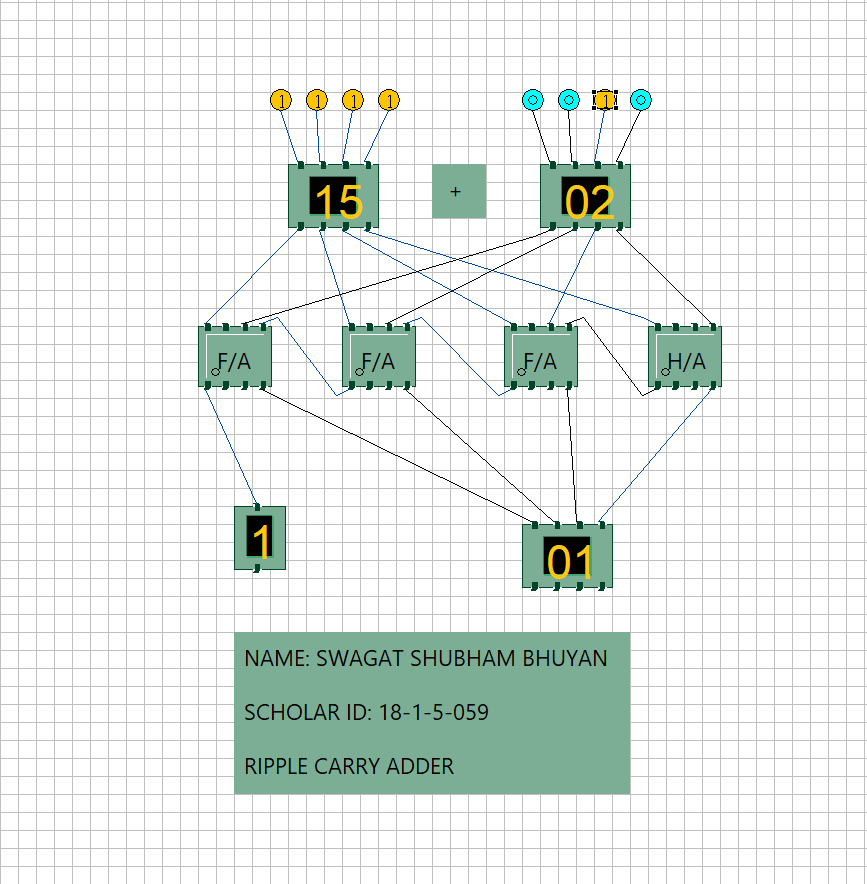


**OUTPUT**: 

**Fig 1. Output depicting the sum of 6 and 5**

.

**Fig 2. Output depicting the sum of 0 and 0**



**Fig 3. Output depicting the sum of 15 and 2**

**SIMULATOR USED**:

Simulator.jar in IITKGP website.

**Experiment:**

2. Carry-Look Ahead Adder

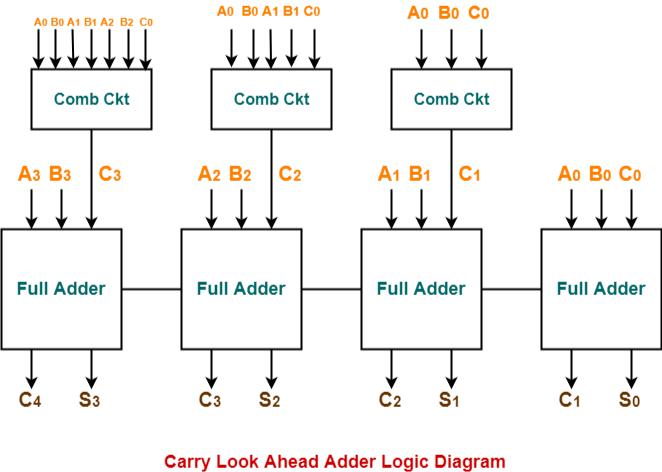
**Objectives**:

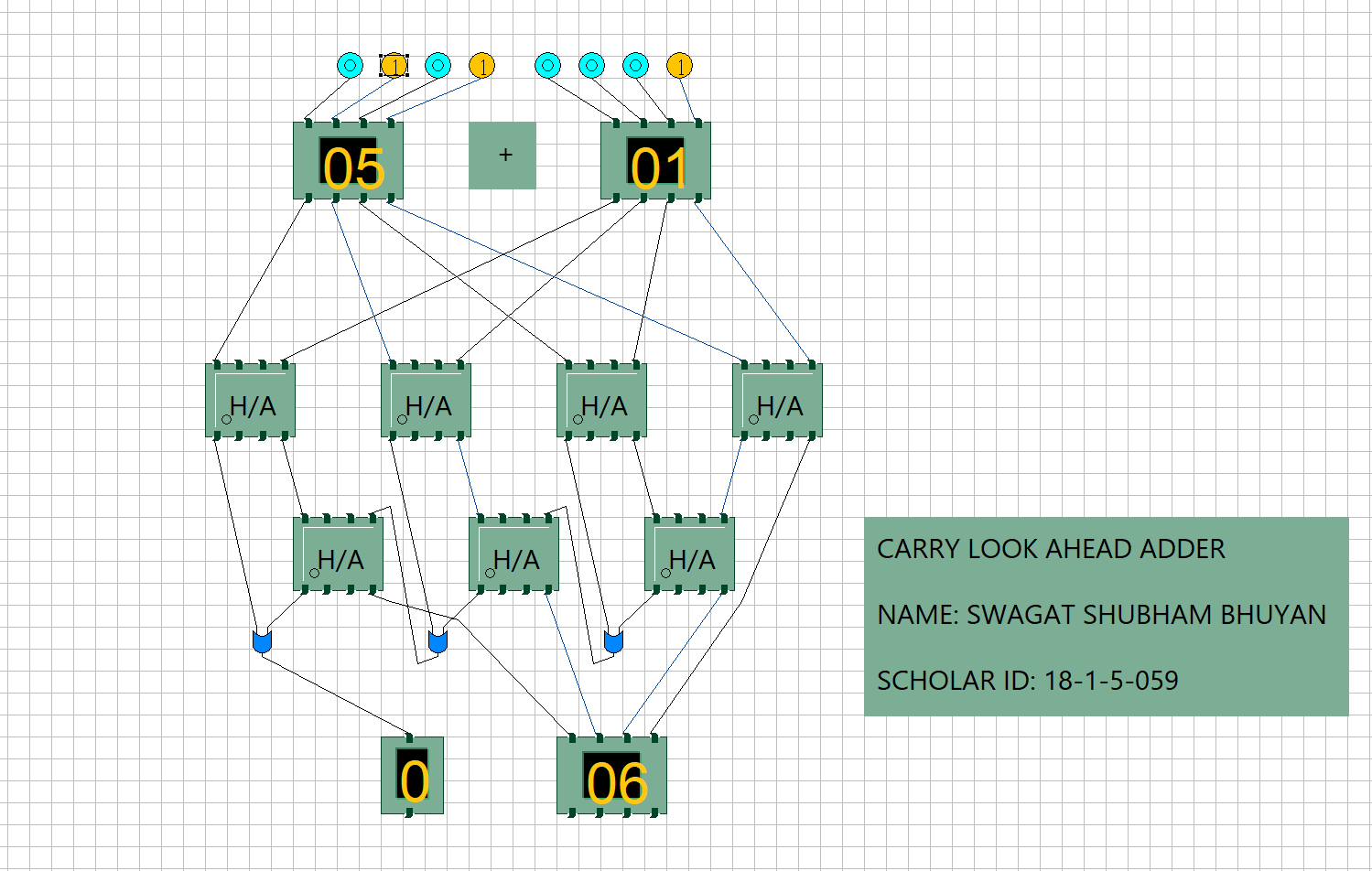
To show the addition of two 4-bit numbers using a 4-bit Carry-Look Ahead Adder.

**Working Principle**:

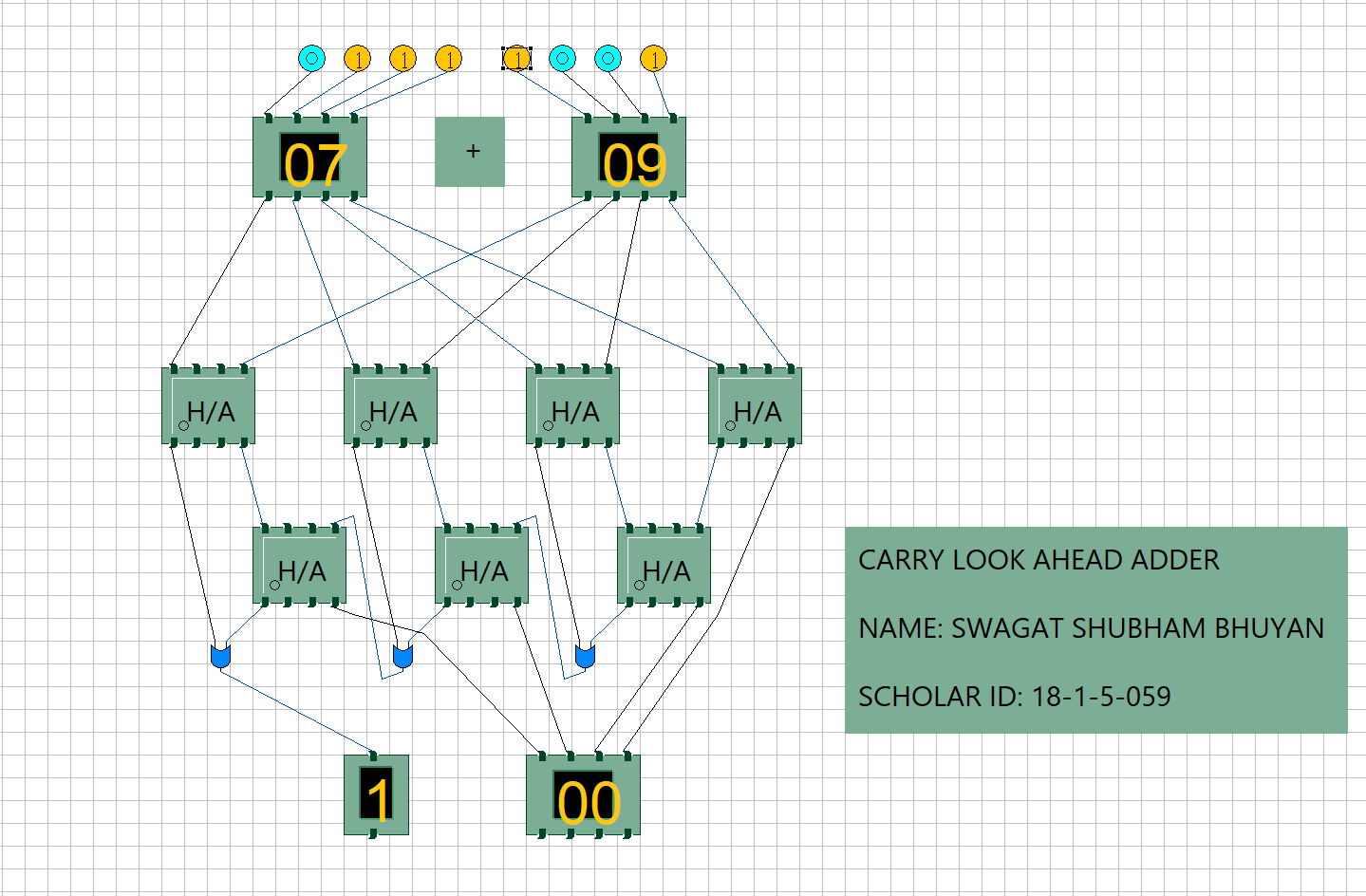
N full adders are cascaded i.e., connected in a parallel manner to build an adder to add twoN-bit numbers. This is called an N-bit Carry Look Ahead Adder. The Ripple Adder was very slow as it had to compute the carry then proceed onto the next adder. But in this adder, all the carry(s) are computed instantaneously given the carry-in and bits of the two numbers. Hence this adder is relatively faster than ripple adder.

**Circuit Diagram**:

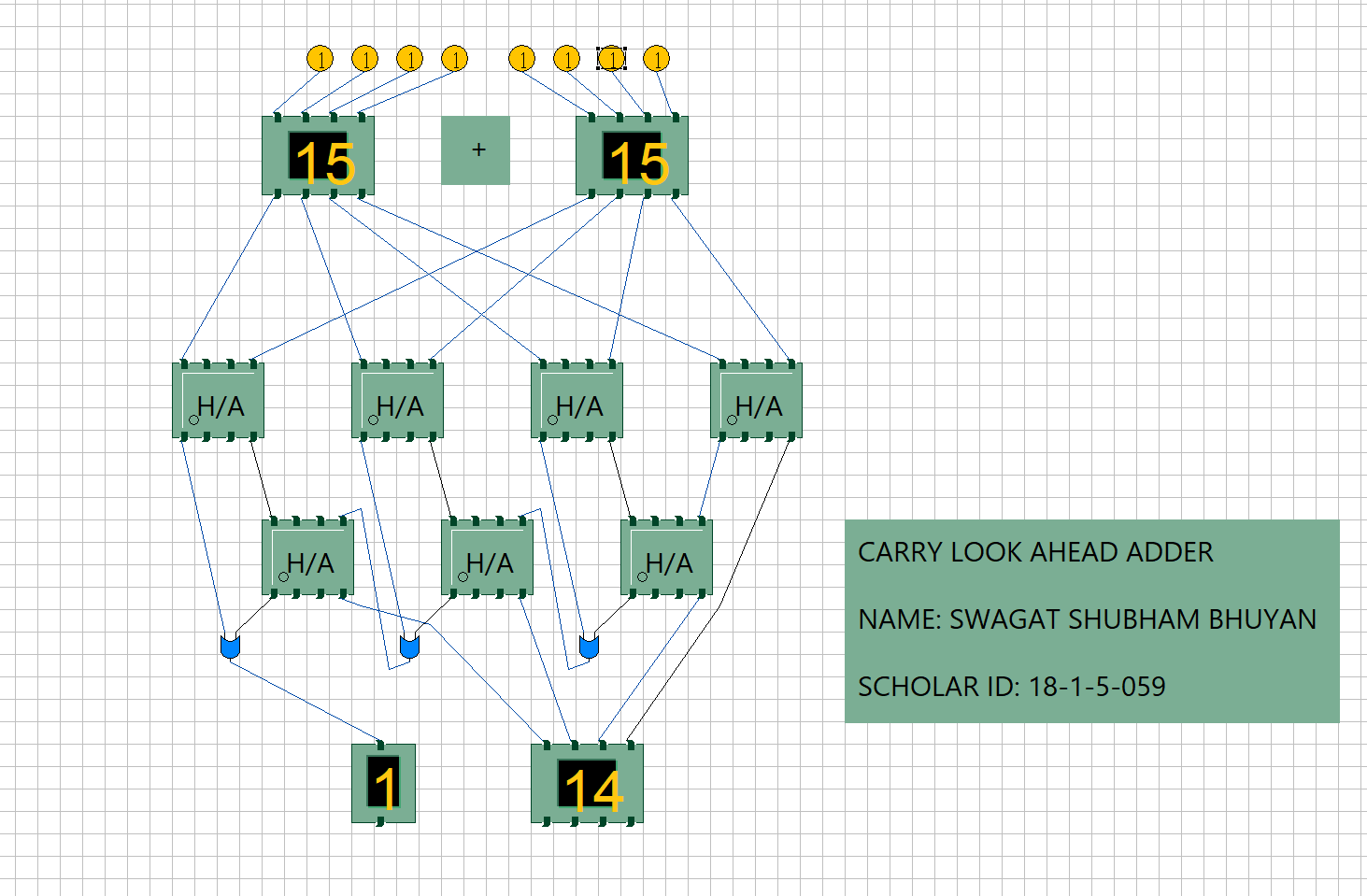




**Fig 4. Output depicting the sum of 5 and 1.**

****

**Fig 5. Output depicting the sum of 7 and 9.**

****

**Fig 6. Output depicting the sum of 15 and 15.**

**SIMULATOR USED**:

Simulator.jar in IITKGP website

**Experiment Name**:

3. Synthesis of flip-flops.

**Objectives**:

To show the synthesis of basic flip-flops like SR, JK, D and T.

**Working Principle**:

1. **Set-Reset Flip Flop**: It is one of the most basic sequential logic circuit. The reset input resets the flip-flopback to its original state with an output Q that will be either at a logic level “1” or logic “0” depending

upon this set/ reset condition.

|  |  |  |  |
| --- | --- | --- | --- |
| **S** | **R** | **Q** | **Qcomplement** |
| 0 | 0 | Q | Qcomplement |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | X | X |

**2.** **JK Flip-Flop:** SR Flip-flop fails when input are logic high. To overcome this condition, JK Flip-flop is used.It has an additional clock input circuitry that prevents the illegal combination. Hence a JK flip-flop toggles its input whenever both J and K are logic high.

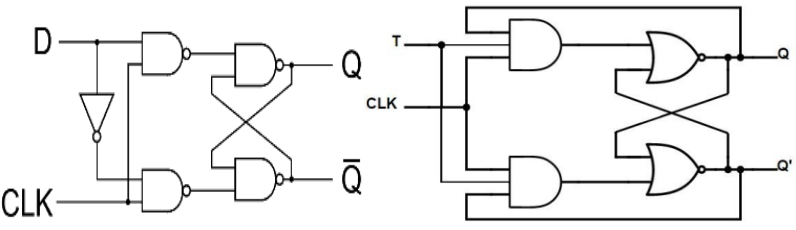
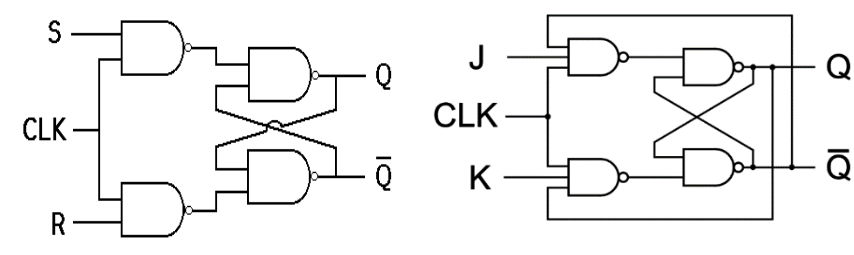
|  |  |  |  |
| --- | --- | --- | --- |
| **J** | **K** | **Q** | **Qcomplement** |
| 0 | 0 | Q | Qcomplement |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | Qcomplement | Q |

**3.** **Delay Flip-flop:** An inverter is connected between the SET and the RESET inputs to provide another typeof flip-flop called Delay Flip-flop. It ensured the inputs SET and RESET are never equal to each other at the same time.

|  |  |  |
| --- | --- | --- |
| **D** | **Q** | **Qcomplement** |
| 0 | 0 | 1 |
| 1 | 1 | 0 |

**4.** **Toggle Flip-flop:** It is like JK Flip-flop. These are basically a single input version of JK FF. This modified formis obtained by connecting both inputs J and K together. This FF has only one input along with the clock input. It toggles the input as its output.

|  |  |  |
| --- | --- | --- |
| **T** | **Q** | **Qcomplement** |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

******Circuit Diagram:**

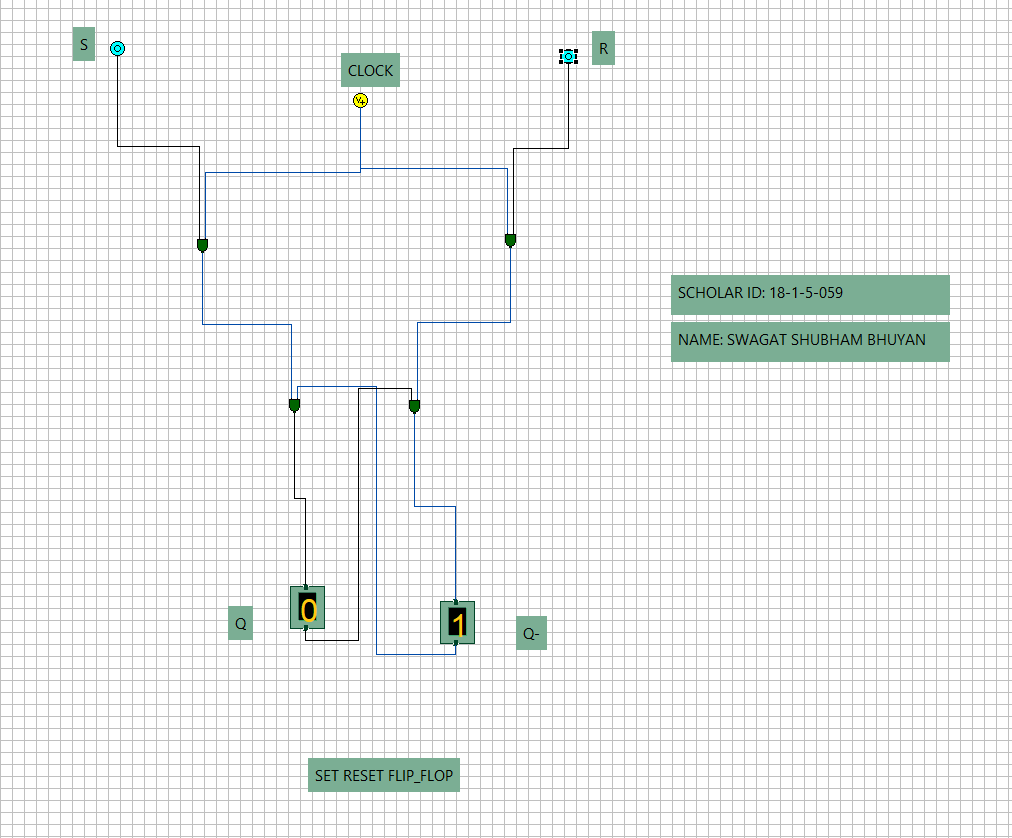
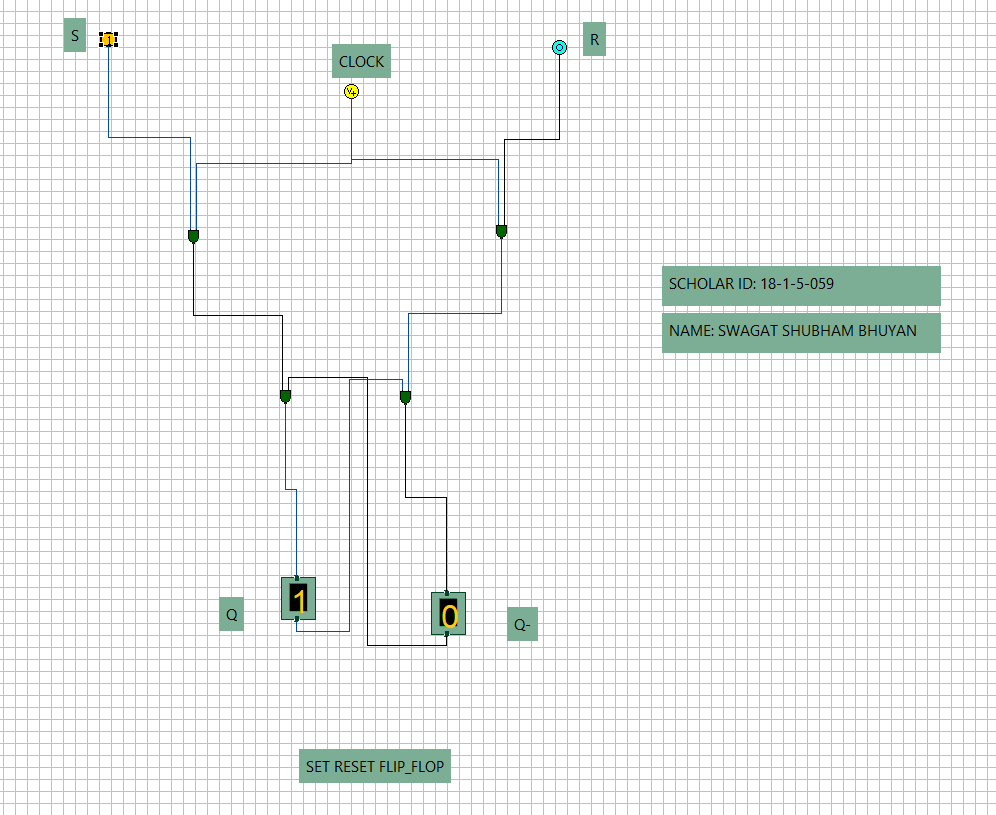
T Flip Flop

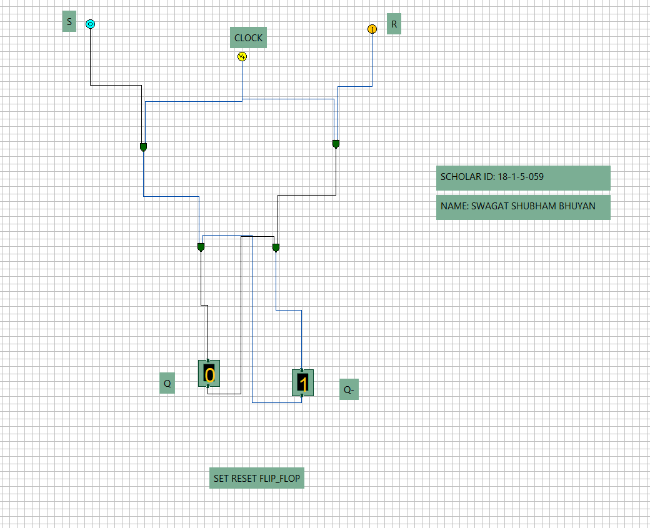
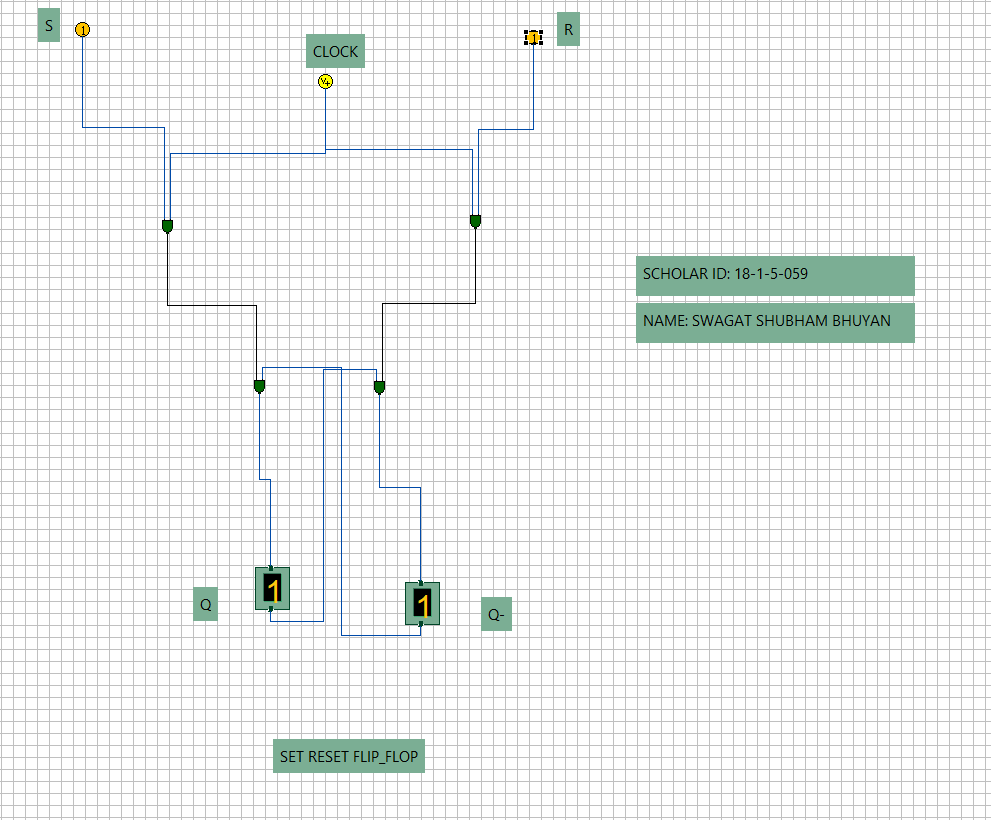
D Flip Flop

J K Flip Flop

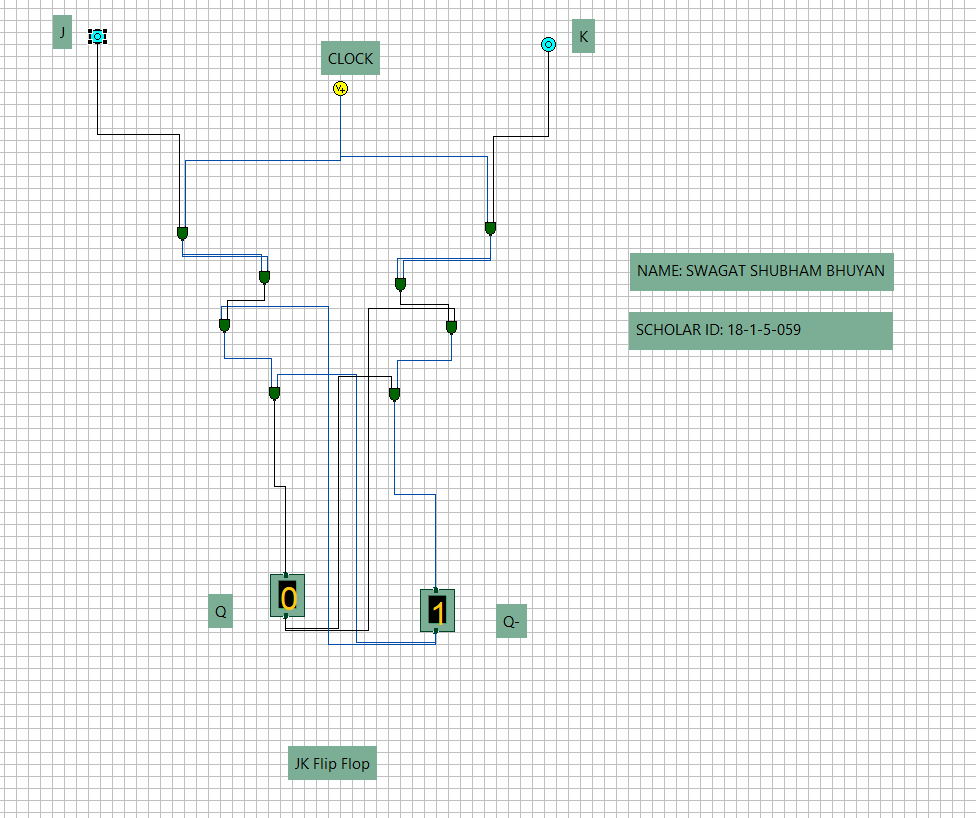
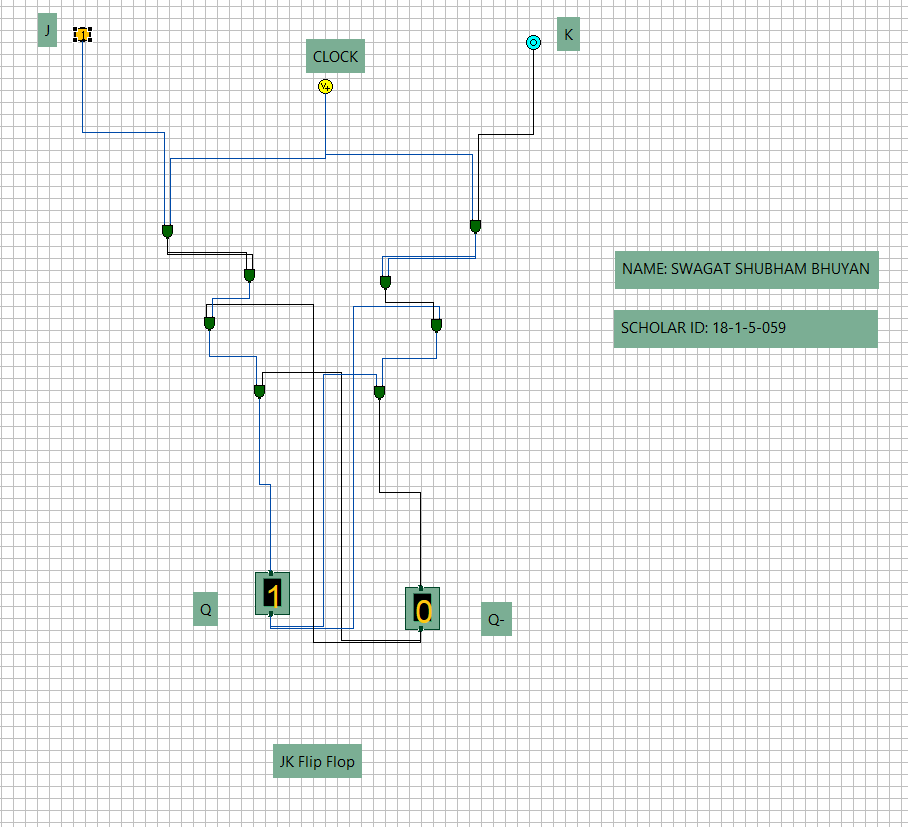
S R Flip Flop

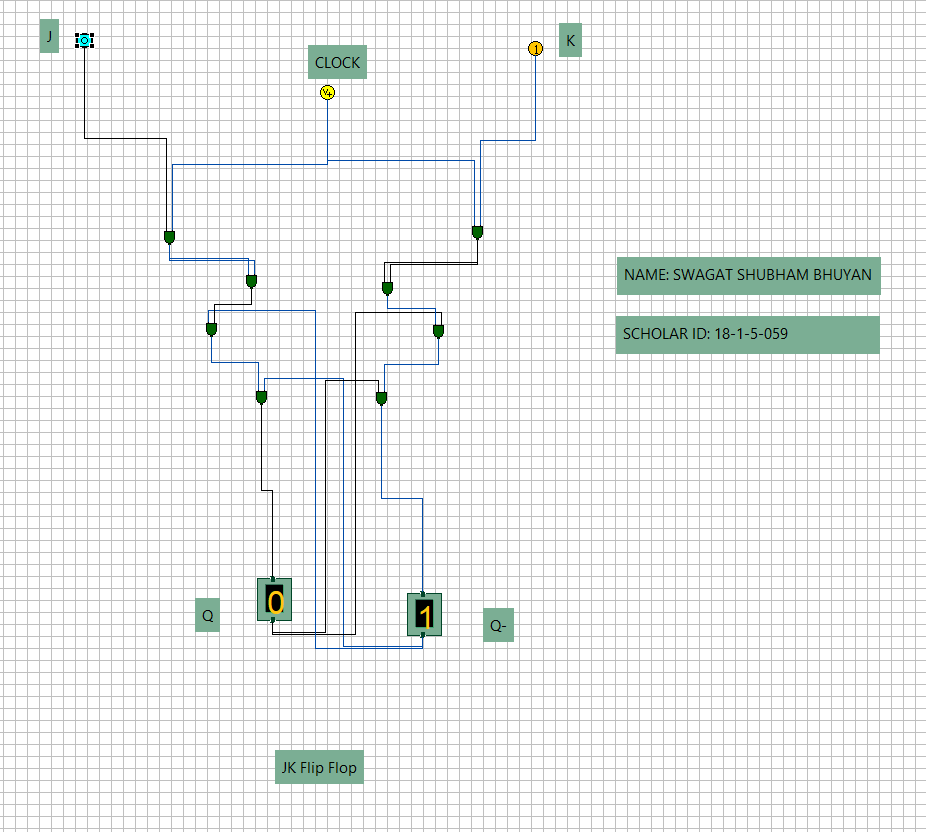
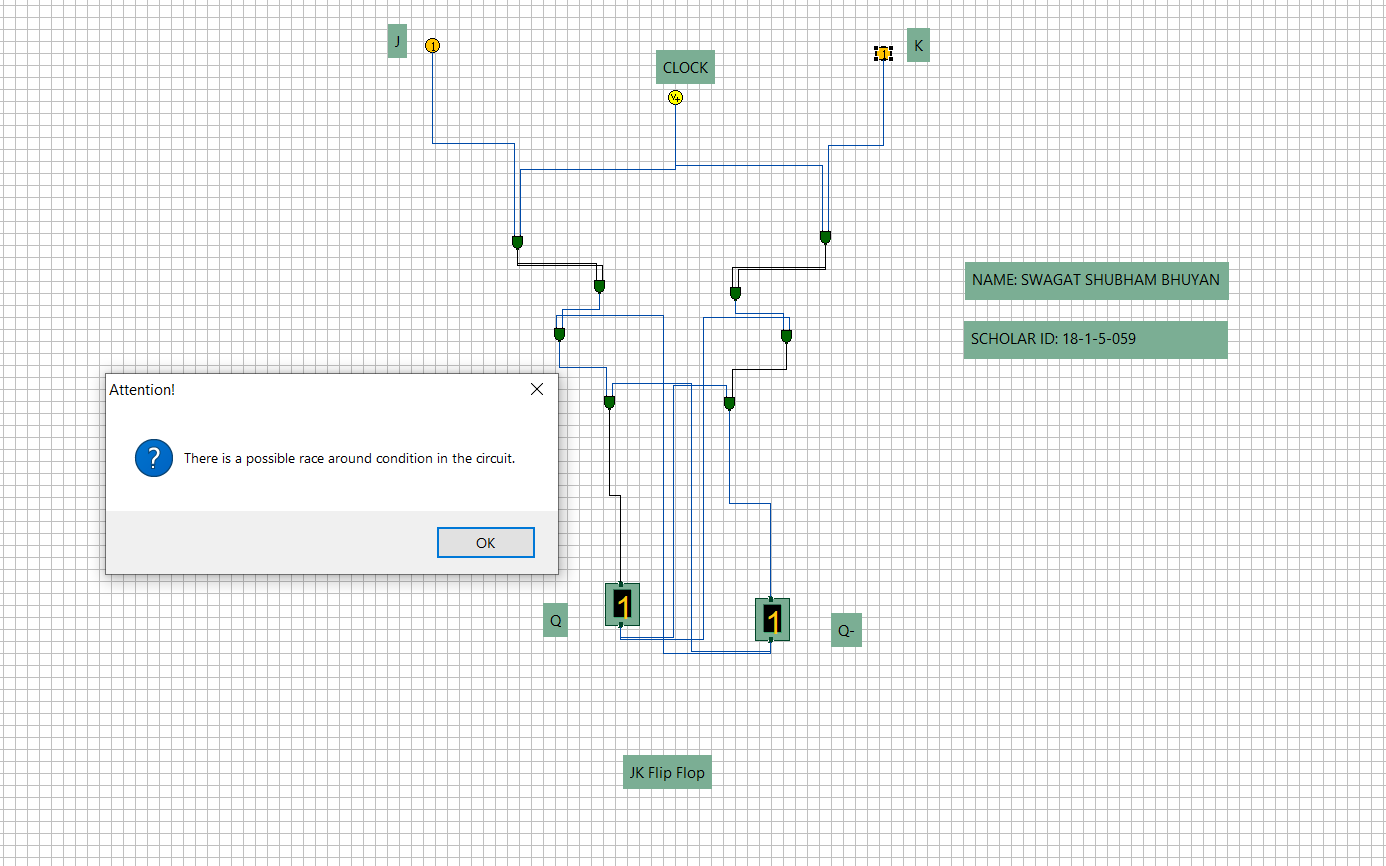
**Output:**

** **

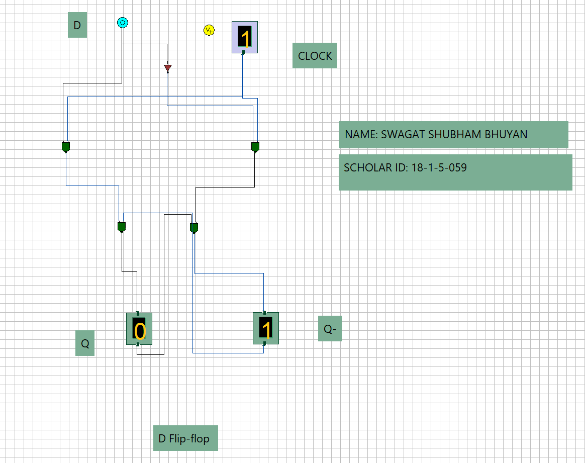
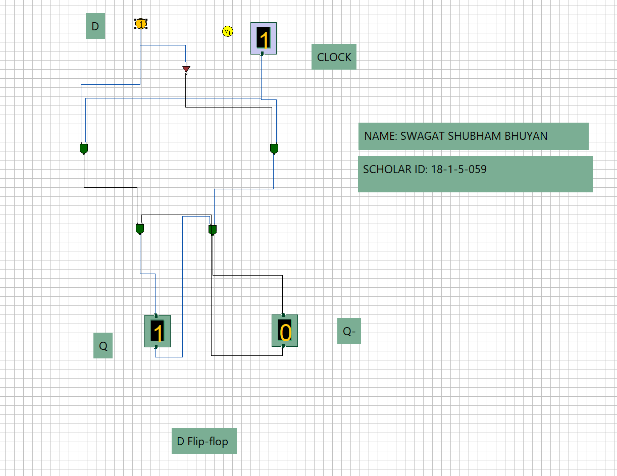
** **

**Fig 7. Set-Reset Flip-Flop OUTPUTs**

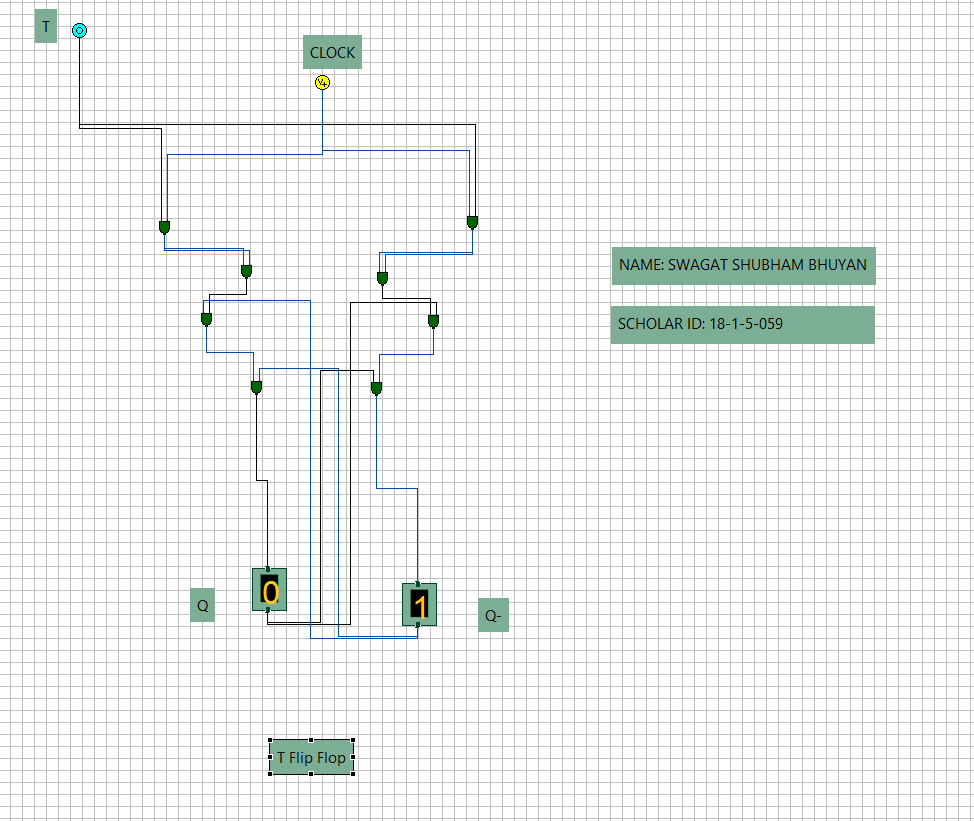
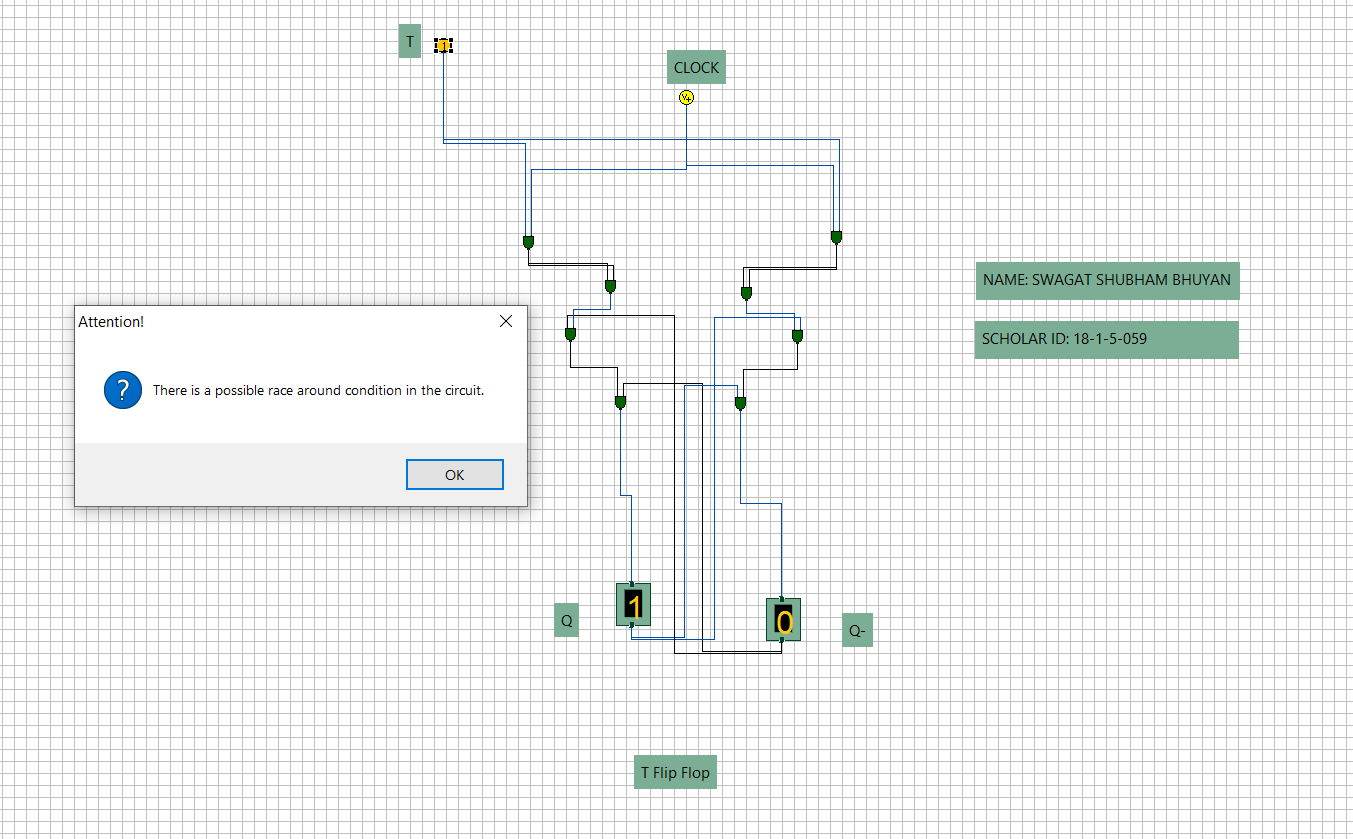
** **

** **

**Fig 8. J K Flip-Flop OUTPUTs**

** **

**Fig 9. D Flip-Flop OUTPUTs**

** **

**Fig 10. T Flip-Flop OUTPUTs**

**SIMULATOR USED**:

Simulator.jar in IITKGP website.

**Experiment Name**:

4. Synthesis of Registers and Counters

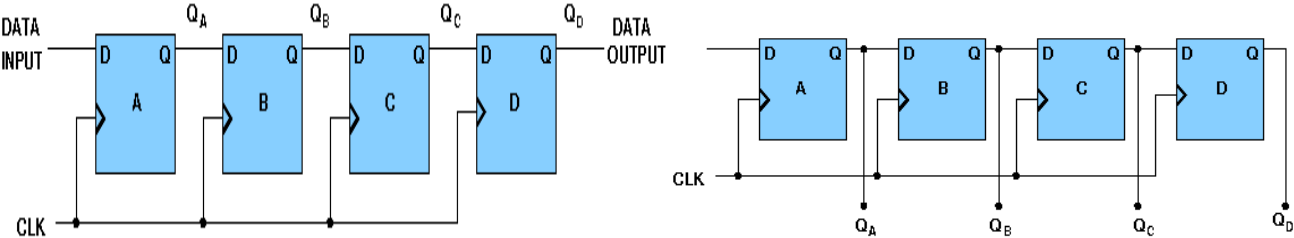
**Objectives**:

To show the synthesis of basic registers and basic counters.

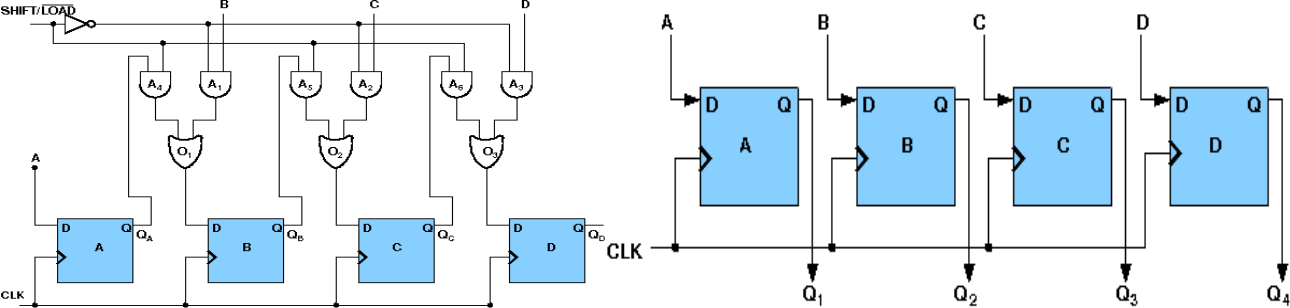
**Working Principle**:

1. **REGISTERS**: Register is a group of flip-flops. Its basic function is to hold information within a digital system soas to make it available to the logic units during the computing process.
2. **COUNTERS**: A register that goes through a prescribed sequence of states upon the application of input pulseis called a counter.

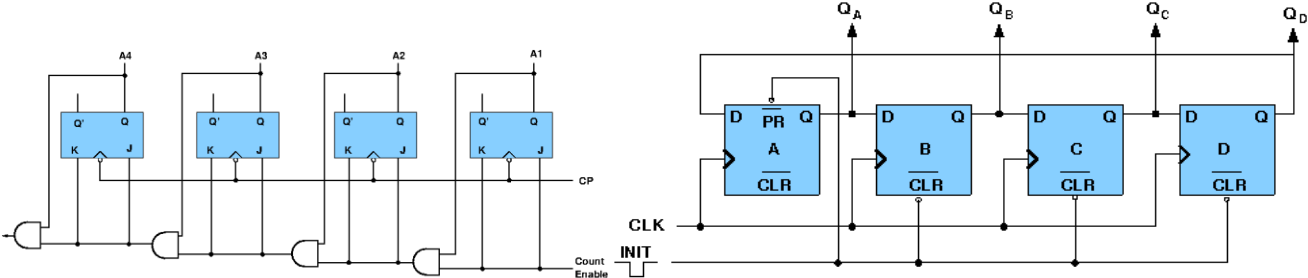
**Circuit Diagram**:



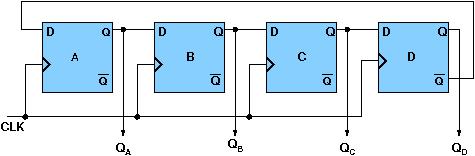
**Serial-In Serial-Out Shift Register** **Serial-In Parallel-Out Shift Register**

****

**Parallel-In Serial-Out Shift Register** **Parallel-In Parallel-Out Register**

****

**Synchronous Binary Counter** **Ring Binary Counter**

****

**Johnson Counter**

**REGISTER OUTPUTs:**

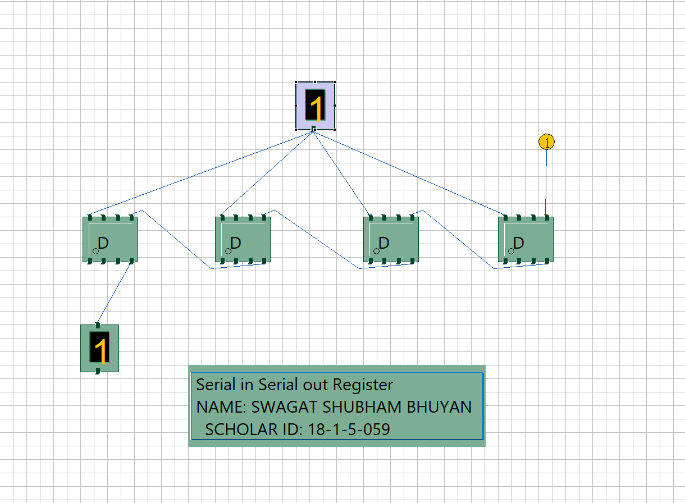
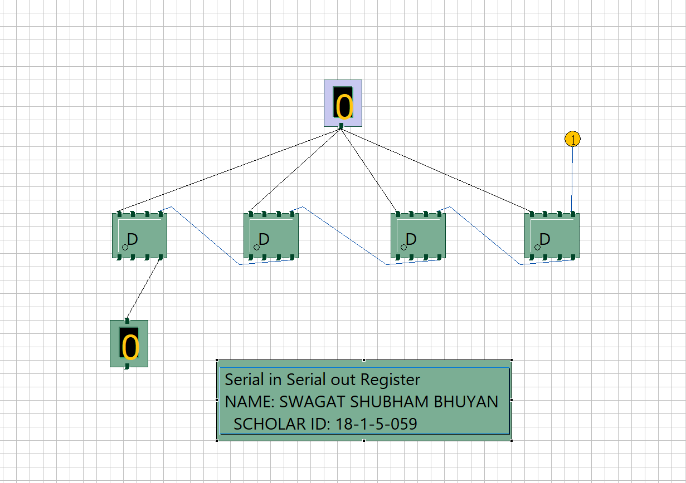
****

Fig 11. Serial-In Serial-Out Shift Register

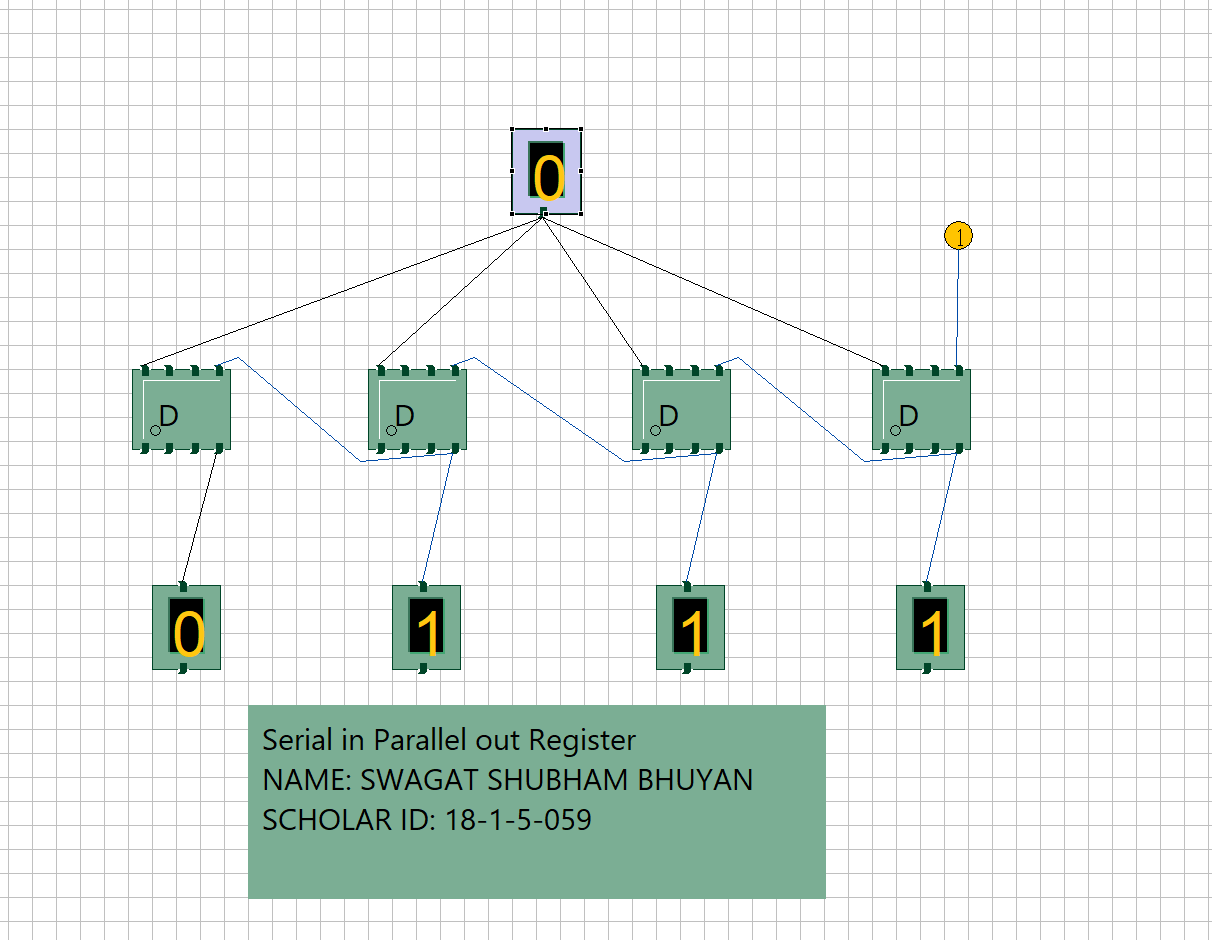
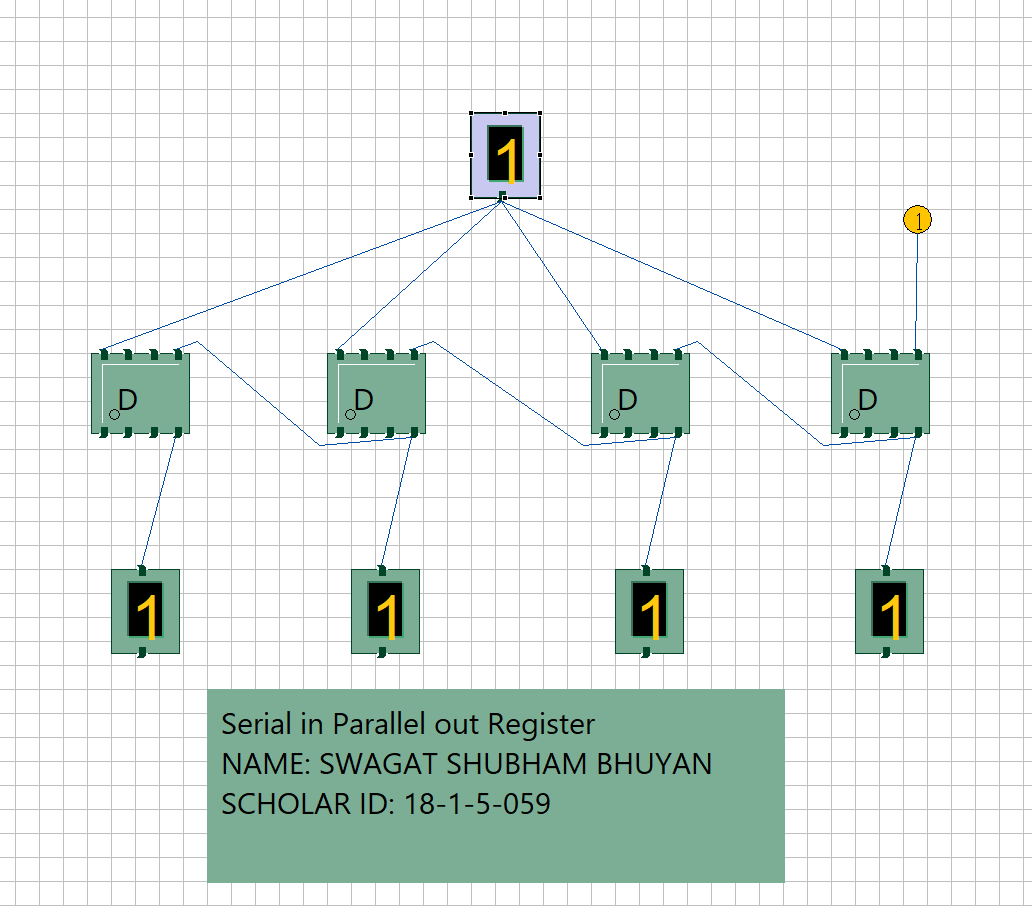
 

Fig 12. Serial-In Parallel-Out Shift Register

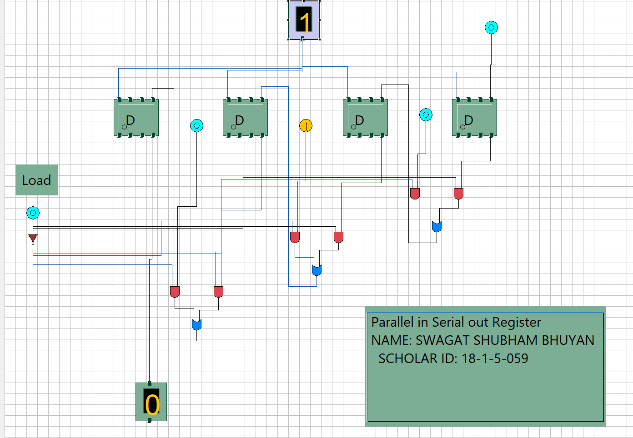
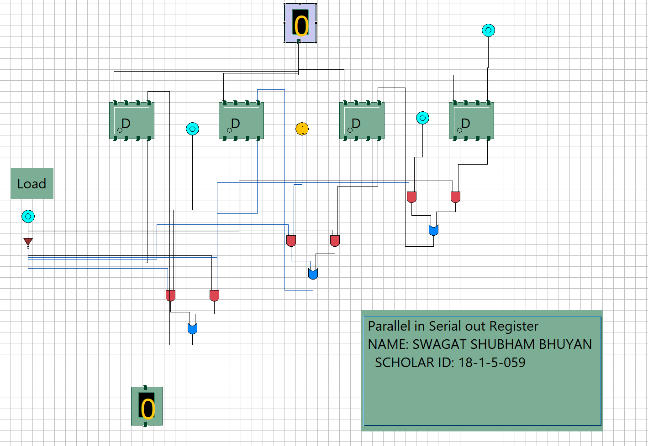
 

Fig 13. Parallel-In Serial-Out Shift Register

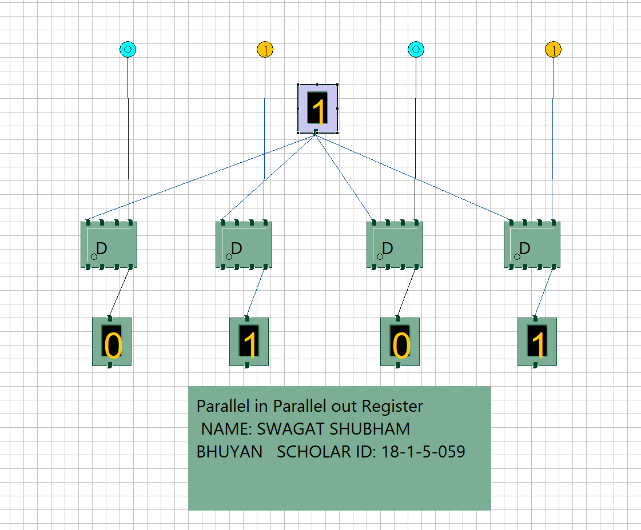
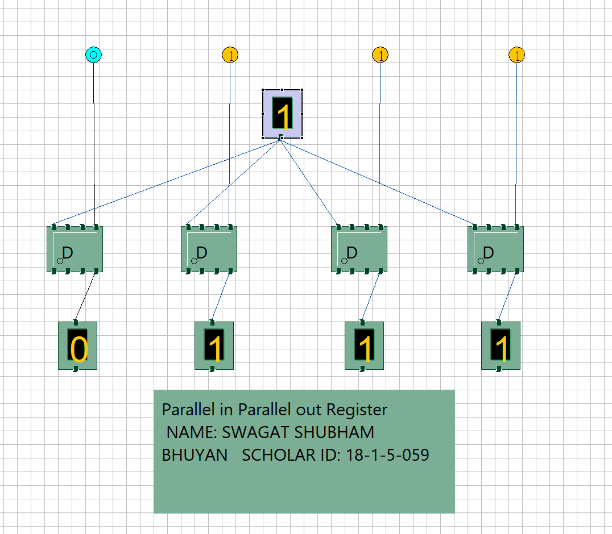
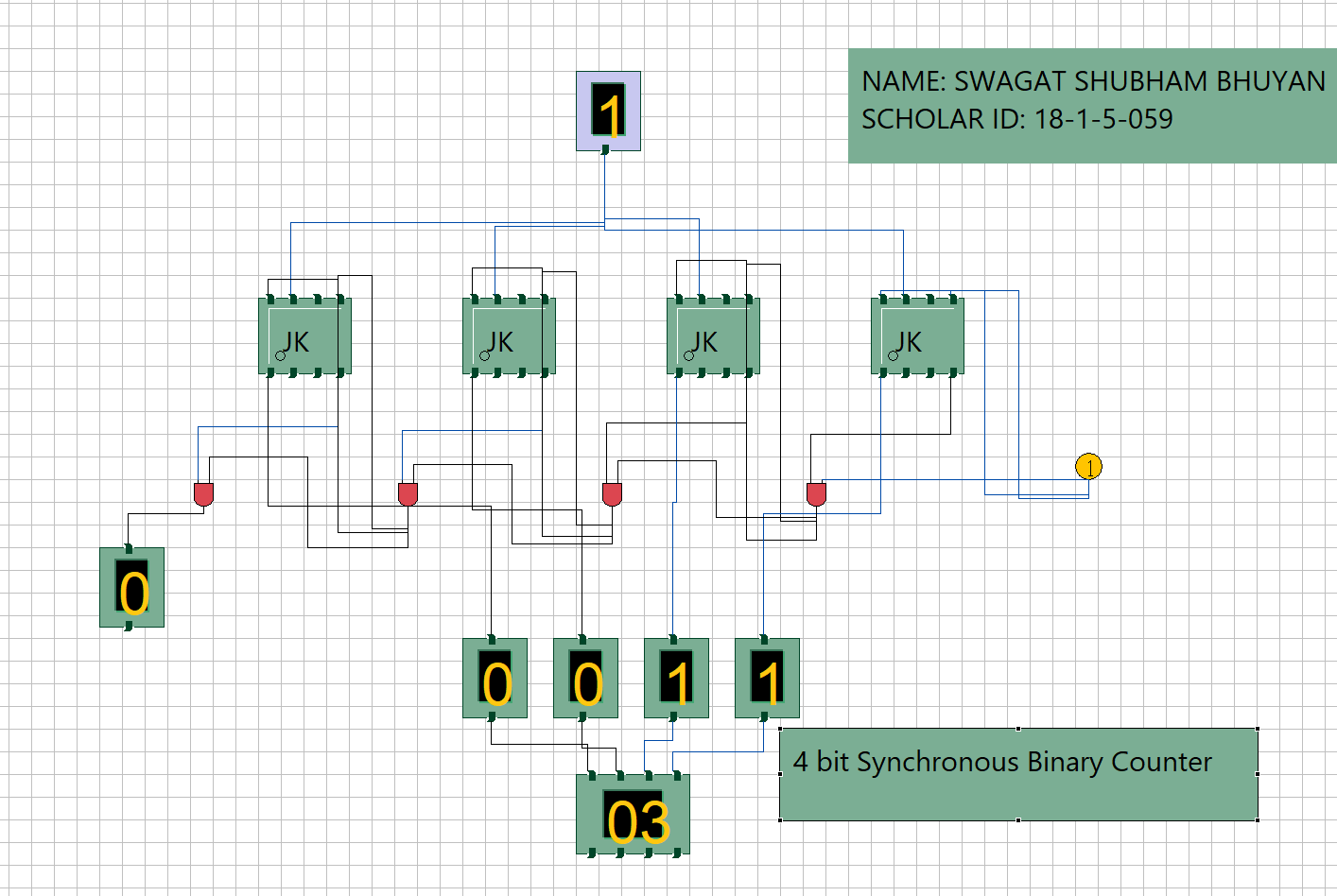
 

Fig 14. Parallel-In Parallel-Out Shift Register

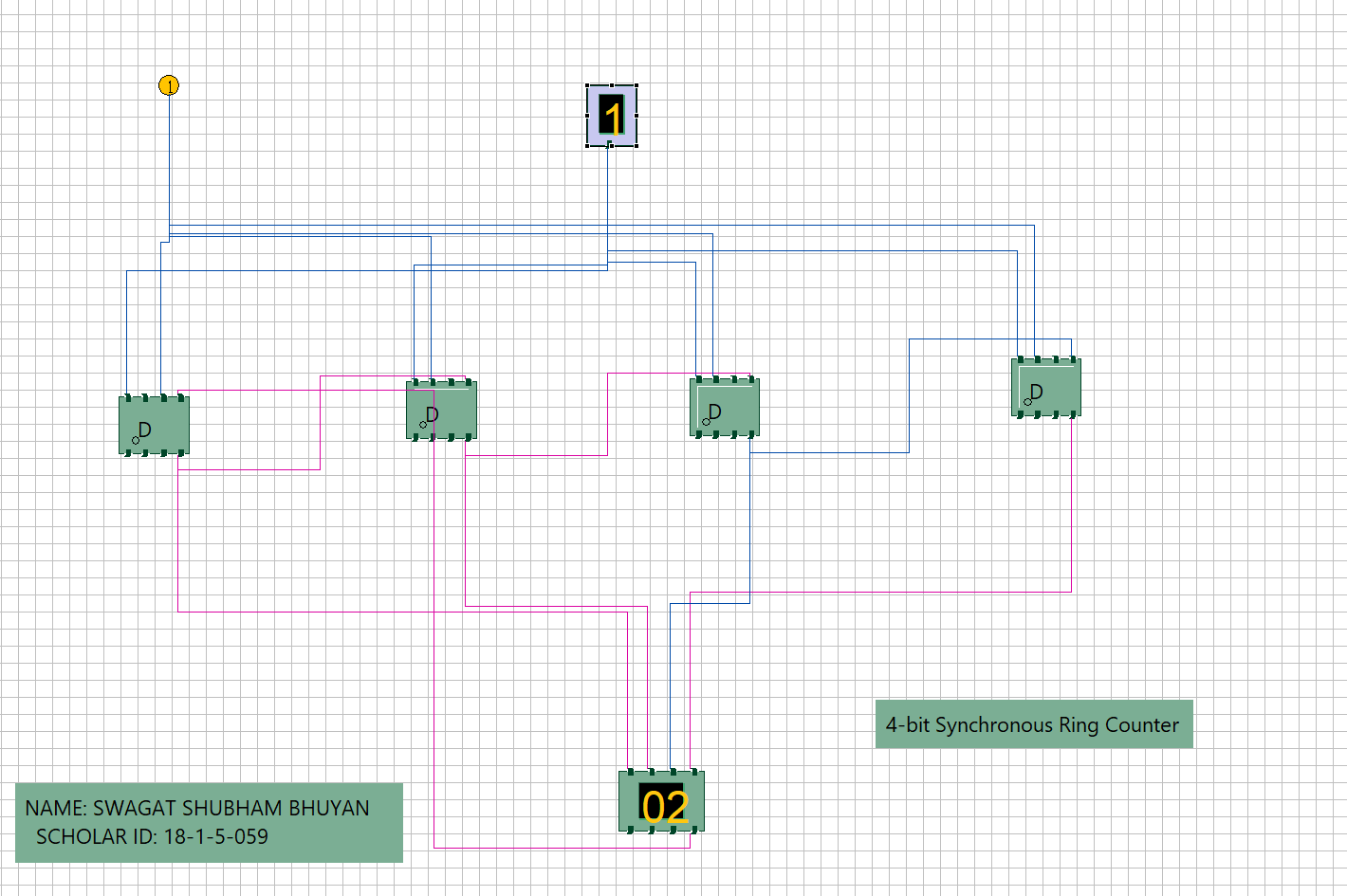
**SIMULATOR USED**:

Simulator.jar in IITKGP website.

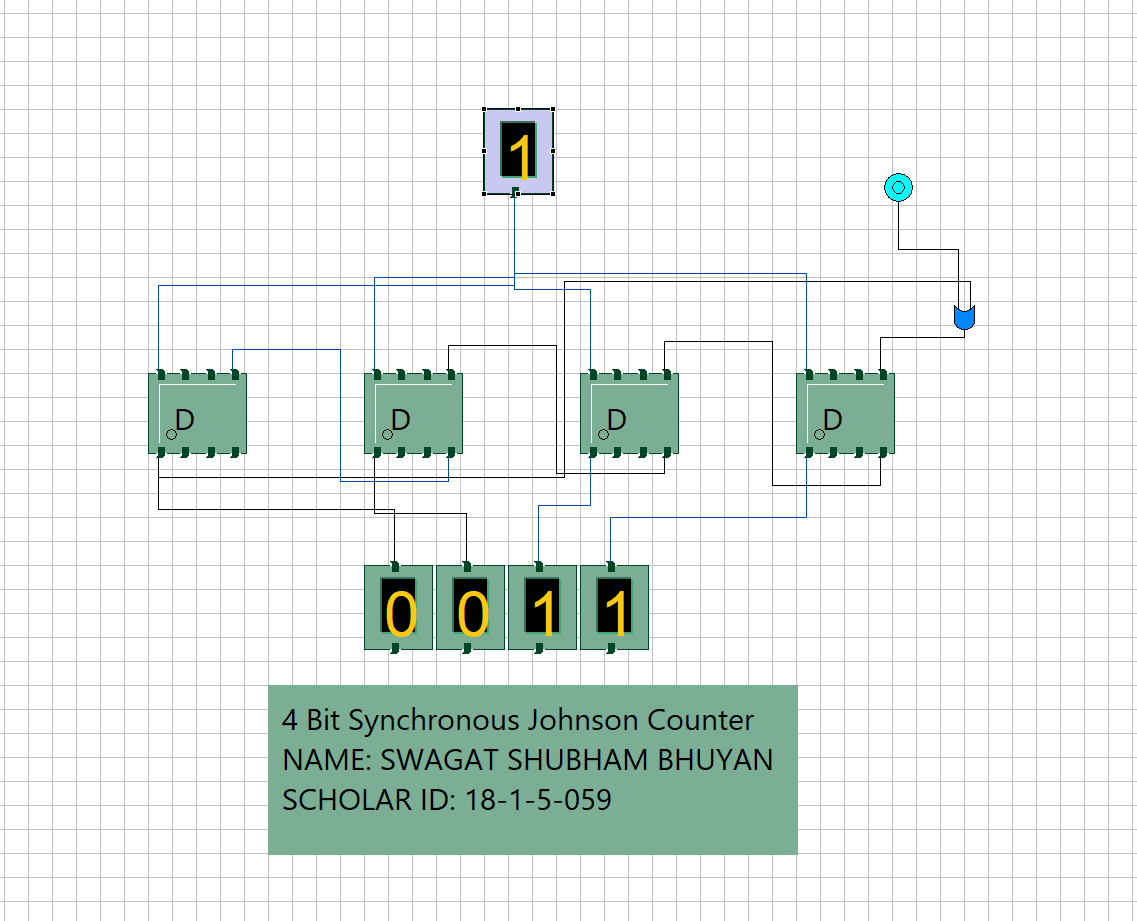
**COUNTER OUTPUTs:**

****

**Fig 15. 4bit Synchronous Binary Counter**

****

**Fig 16. 4bit Synchronous Ring Counter**

****

**Fig 17. 4bit Synchronous Johnson Counter**

**SIMULATOR USED**:

Simulator.jar in IITKGP website.

**Experiment**:

5. Combinational Multiplier.

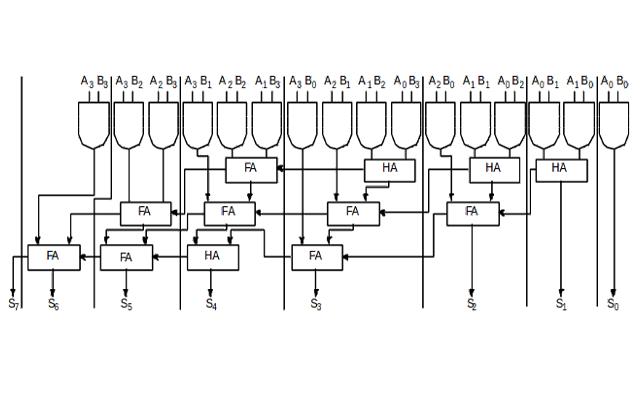
**Objectives**:

To show the multiplication of two 4-bit numbers using a Combinational Multiplier.

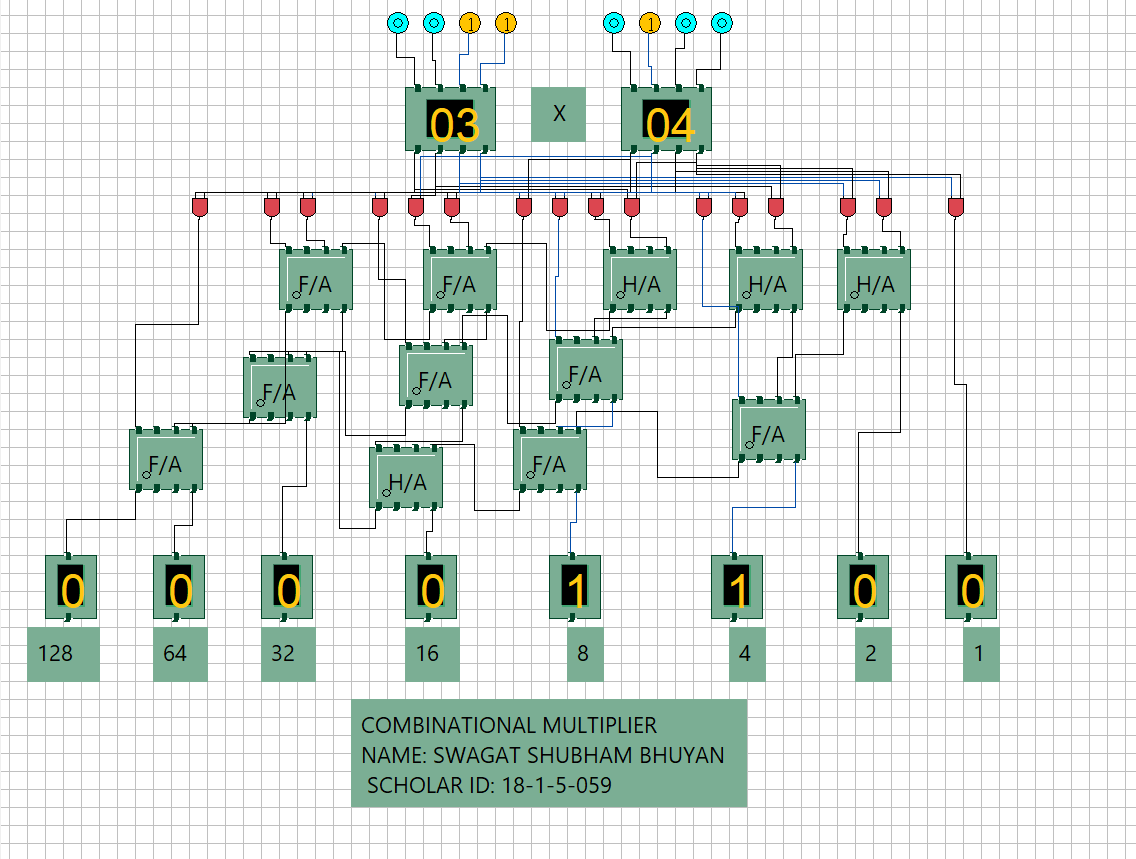
**Working Principle**:

Combinational Multipliers do multiplication of two unsigned binary numbers. Each bit of themultiplier is multiplied against the multiplicand, the product is aligned according to the position of the bit within the multiplier, and the resulting products are then summed to form the final result. Main advantage of binary multiplication is that the generation of intermediate products are simple: if the multiplier bit is a 1, the product is an appropriately shifted copy of the multiplicand; if the multiplier bit is a 0, the product is simply 0. If two n-bit numbers are multiplied then the output will be less than or equals to 2n bits.

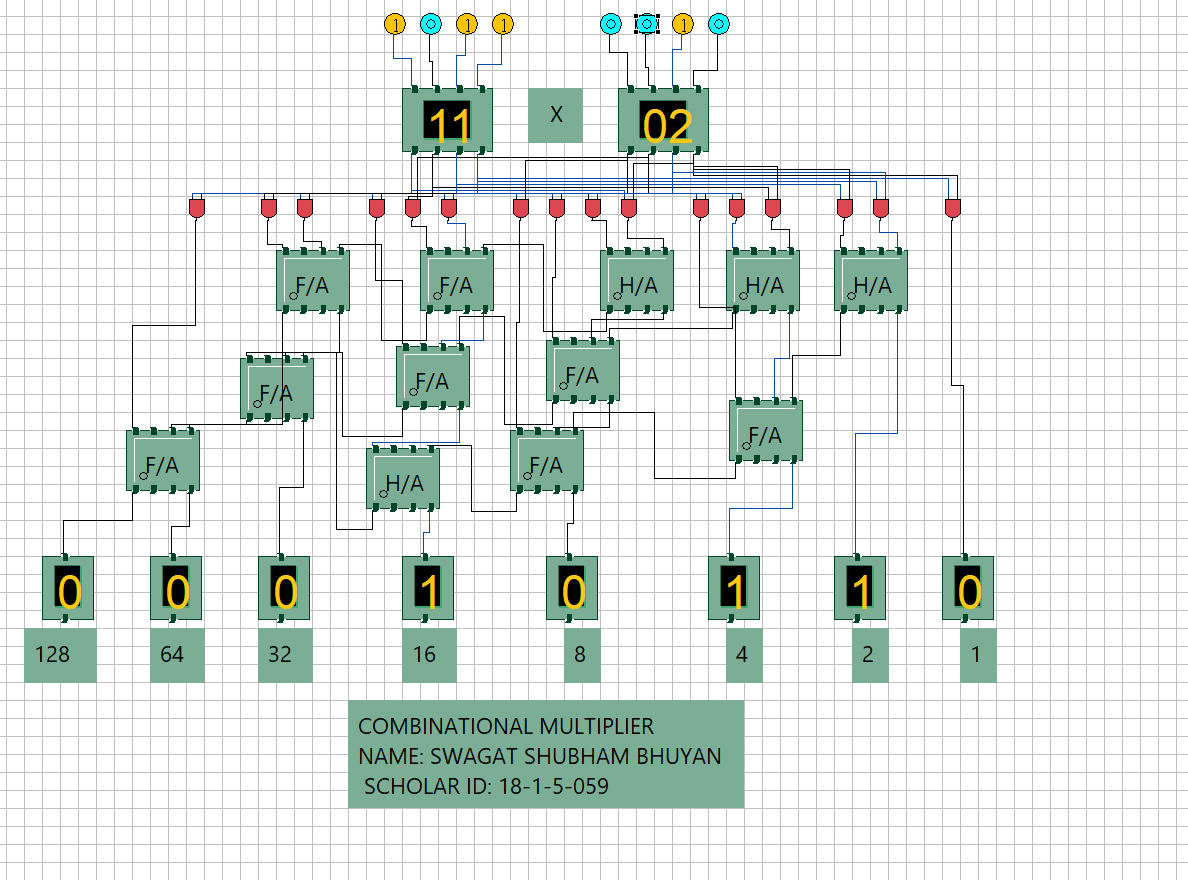
**Circuit Diagram**:



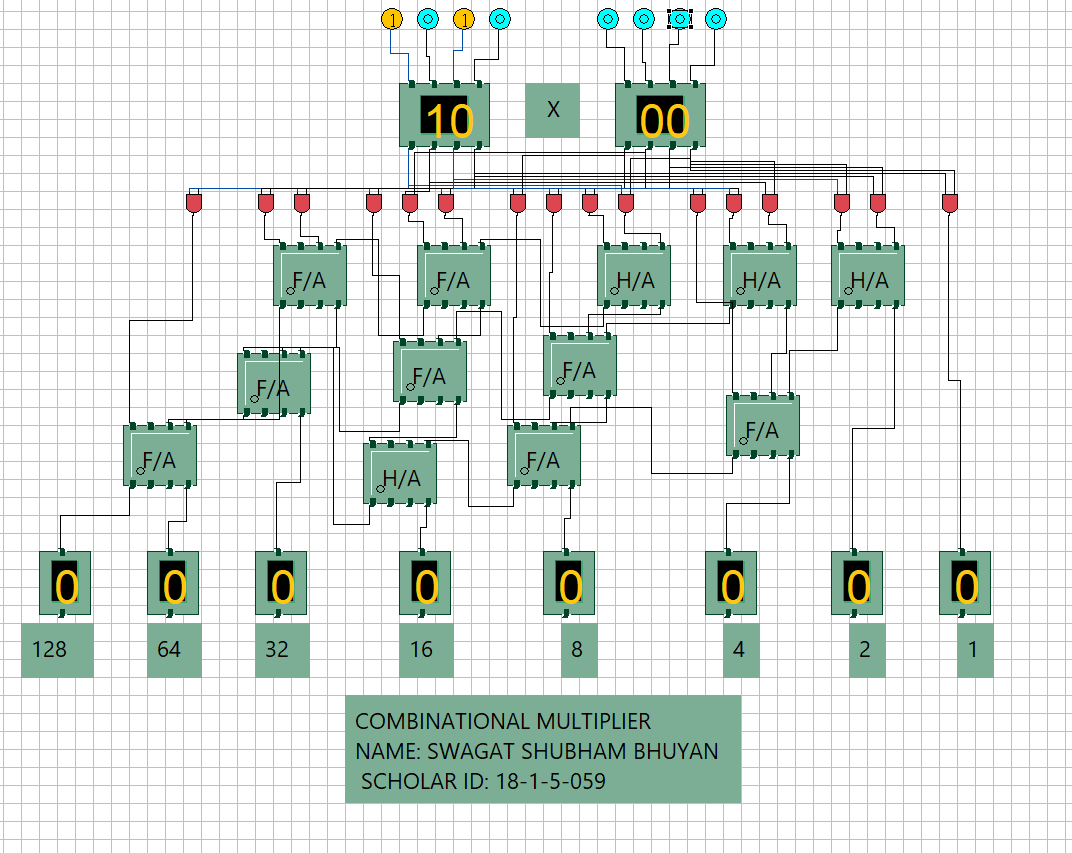
Output:



**Fig 18. Output depicting the product of 3 and 4.**



**Fig 19. Output depicting the product of 11 and 2.**



**Fig 20. Output depicting the product of 10 and 0.**

**SIMULATOR USED**:

Simulator.jar in IITKGP website.

**Experiment:**

6. Booth’s Multiplier.

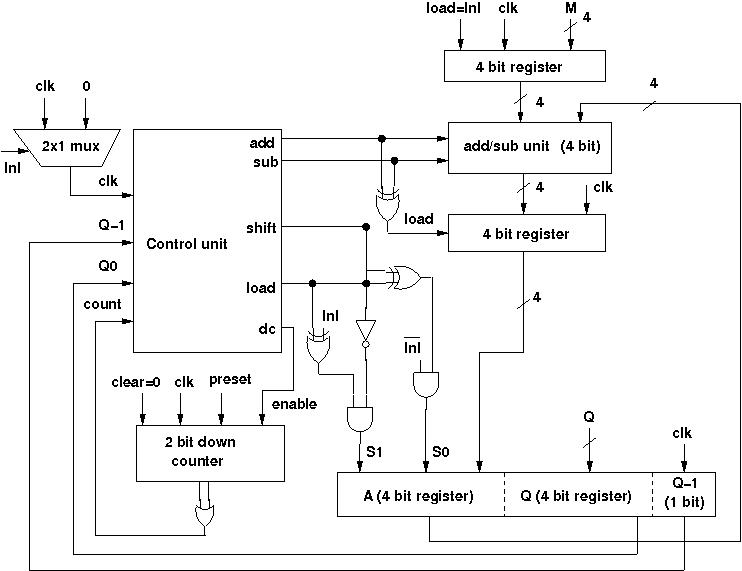
**Objectives**:

To show the multiplication of two 4-bit numbers using a Booth’s Multiplier.

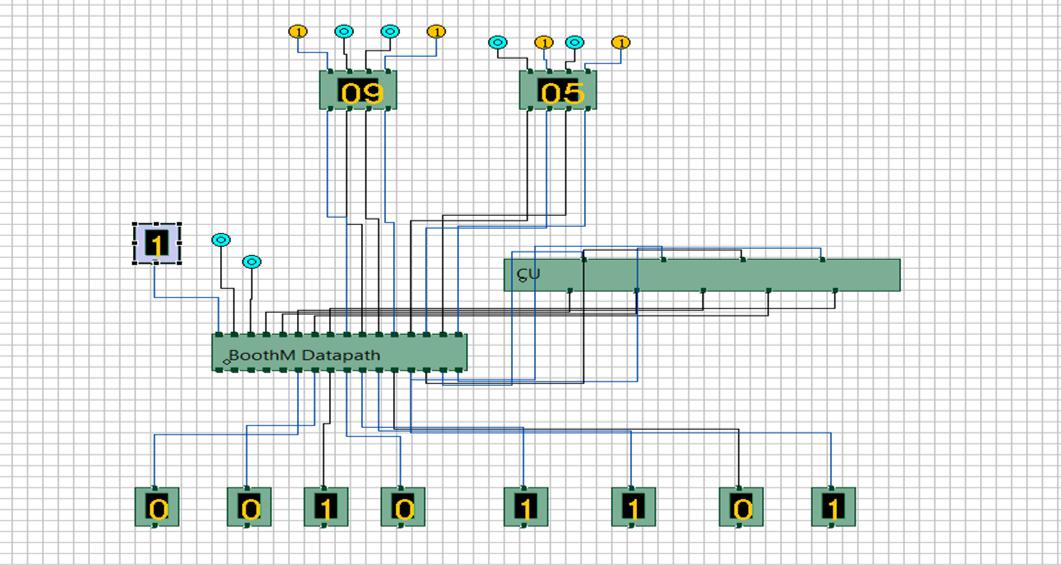
**Working Principle**:

Booth's multiplication algorithm is an algorithm which multiplies 2 signed integers in 2'scomplement. The algorithm is depicted in the following figure with a brief description. This approach uses fewer additions and subtractions than more straightforward algorithms. The multiplicand and multiplier are placed in the m and Q registers respectively. A 1 bit register is placed logically to the right of the LSB (least significant bit) Q0 of Q register. This is denoted by Q-1. A and Q-1 are initially set to 0. Control logic checks the two bits Q0 and Q-1. If the two bits are same (00 or 11) then all of the bits of A, Q, Q-1 are shifted 1 bit to the right. If they are not the same and if the combination is 10 then the multiplicand is subtracted from A and if the combination is 01 then the multiplicand is added with A. In both the cases results are stored in A, and after the addition or subtraction operation, A, Q, Q-1 are right shifted. The shifting is the arithmetic right shift operation where the left most bit namely, An-1 is not only shifted into An-2 but also remains in An-1. This is to preserve the sign of the number in A and Q. The result of the multiplication will appear in the A and Q.

**Circuit Diagram**:



**OUTPUT**:

****

**Fig 21. Output depicting the product of 9 and 5.**

**SIMULATOR USED**:

Simulator.jar in IITKGP website