**LAB-13**

**Frequency divider, Synchronous Counter and Asynchronous Counter with Reset Operations**

]

**Equipment/Components**

**Hardware:** Explorer Board

IC type Dual D flip flop

IC Type 74LS76 JK flip flop

IC Type 74LS04 NOT gate

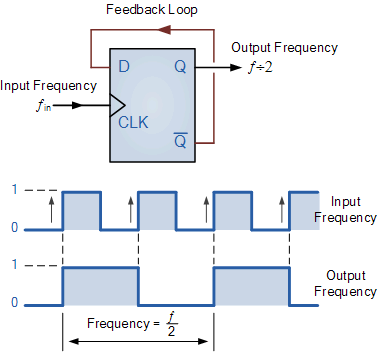
IC type Dual T flip flop

**Software:** Circuit Maker, Waveform

**Description**

**Frequency Divider**

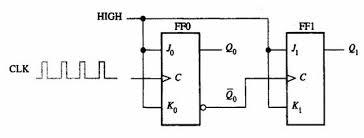
A useful feature of the D-type Flip-Flop is a binary divider, can be used for **Frequency Division**. Here the inverted output terminal Q (NOT-Q) is connected directly back to the data input terminal D giving the device “feedback” as shown below.



**Figure 1**

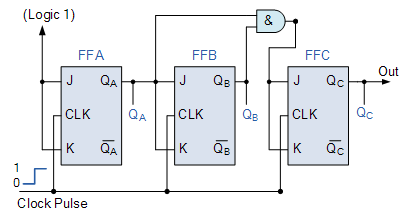
**Counters**

Flip flops can be used to form the basis for counters and registers. The counters are of two main types: **synchronous** and **asynchronous**. These flip flop and logic gates can be combined to produce different types of asynchronous counters. The Fig. 1 and Fig.2 illustrates the asynchronous and synchronous counter operation using JK flip-flop.



**Figure 1. Asynchronous Counter Operation using JK flip-flop**

Please see datasheet for details of the flip-flop.



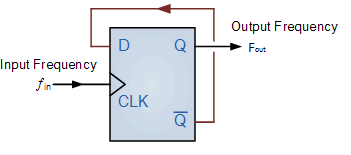
**Figure 2. Synchronous Counter Operation using JK flip-flop**

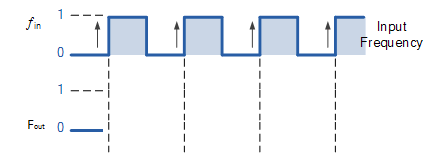
**Objective**

* To understand the concept and working of a synchronous and asynchronous counters (2, 3 and 4 bits) and to use IC 74LS76 JK flip flop for hardware implementation.
* To understand the concept and working of an asynchronous decade counters (BCD decade counter)

**Task # 1**

1. Consider the configuration of D flip flop shown below and draw the output (Fout) on page with respect to given clock pulse (Fin).

****

****

1. Find the relationship between input (Fin) and output frequency (Fout). Also, comment on relationship.
2. Dry run the given circuit and fill the table below for given input.

|  |  |
| --- | --- |
| **fin** | **fout** |
| 1KHz | **500hz** |
| 500Hz | **250hz** |
| 200Hz | **100hz** |
|  |  |

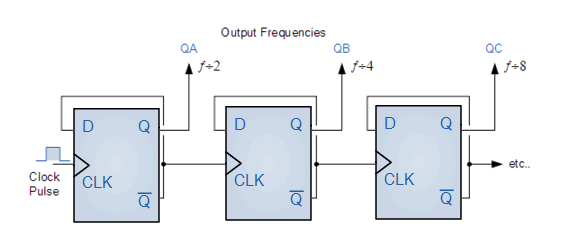
1. Design the circuit on Circuit Maker and fill the table below

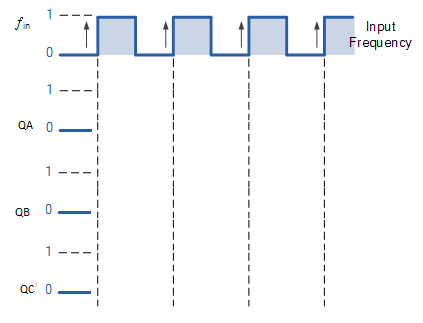
|  |
| --- |
|  |

|  |
| --- |
|  |

**Task # 2**

1. Consider the configuration of D flip flop shown below and draw the outputs (QA, QB and QC) **on page** with respect to given clock pulse (Fin).



****

1. Find the relationship between input (Fin) and output frequency (Fout). Also, comment on relationship.
2. Dry run the circuit and note the output frequencies

|  |  |  |  |
| --- | --- | --- | --- |
| **fin** | **QA** | **QB** | **QC** |
| 1KHz | **500hz** | 250hz | **125hz** |
| 500Hz | **250hz** | 125hz | **62.5hz** |
| 200Hz | **100hz** | 50hz | **25hz** |
|  |  |  |  |

1. Design the circuit on Circuit Maker and note the frequencies

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| **fin** | **QA** | **QB** | **QC** |
| 1KHz | **500hz** | 250hz | **125hz** |
| 500Hz | **250hz** | 125hz | **62.5hz** |
| 200Hz | **100hz** | 50hz | **25hz** |
|  |  |  |  |

1. Implement the circuit on explorer board and note the frequencies

|  |
| --- |
| **Hardware Implementation** (Wave generated by explorer board) |

|  |  |  |  |
| --- | --- | --- | --- |
| **fin** | **QA** | **QB** | **QC** |
| 1KHz | **500hz** | 250hz | **125hz** |
| 500Hz | **250hz** | 125hz | **62.5hz** |
| 200Hz | **100hz** | 50hz | **25hz** |
|  |  |  |  |

1. Compare the result of software simulation and hardware implementation

**Task # 3**

1. Design and implement on Circuit Maker a 3-bit asynchronous binary up counter using JK flip-flop as 2-bit counter is illustrated in Fig. 1. Dry run, check and understand it’s working.

**Note 1.** Make sure flip flop should be working in **toggle mode** with **active high clock**.

**Note 2.** You can use two-way buttons in place of clock for testing purpose.

|  |
| --- |
| Circuit Maker Implementation. |

1. Fill the table shown below for the given clock pulse. Draw the wave form (on page) for, and. (paste Circuit Maker wave form here)

|  |
| --- |
| Timing diagram using Circuit Maker |

|  |  |  |  |
| --- | --- | --- | --- |
| Clock Pulse | Q2 | Q1 | Q0 |
| Initially | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 |
| 5 | 1 | 0 | 0 |
| 6 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 |
| 8 | 1 | 1 | 1 |
| 9 | 0 | 0 | 0 |

1. What is the range (maximum number which can count) of the designed counter?

0-7

1. Find the clock pulse for recycle. (Where, **recycle** refers to the transition of counter from its final state to initial/original state.)

At 9

**Task # 4**

1. Design and implement on **Circuit Maker** a 4-bit asynchronous binary up counter using JK flip-flop. Dry run, check and understand it’s working.

|  |
| --- |
| Circuit Maker Implementation. |

1. Fill the table shown below for the clock pulse.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clock Pulse | Q3 | Q2 | Q1 | Q0 |
| Initially | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 |
| 14 | 1 | 1 | 1 | 0 |
| 15 | 1 | 1 | 1 | 1 |

1. What is the range (maximum number which can count) of the designed counter?

0-15

1. Find the clock pulse for recycle. (Where, **recycle** refers to the transition of counter from its final state to initial/original state.)

Initial

1. Fill the table shown below for the clock pulse.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clock Pulse | Q3 | Q2 | Q1 | Q0 |
| Initially | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 |

1. What is the range (maximum number which can count) of the designed counter?

0-13

1. Find the clock pulse for recycle. (Where, **recycle** refers to the transition of counter from its final state to initial/original state.)

Initial