Switching Circuits and Logic Design Laboratory
CS29002

# Binary to BCD – Sequential Double Dabble

*Experiment - VI*

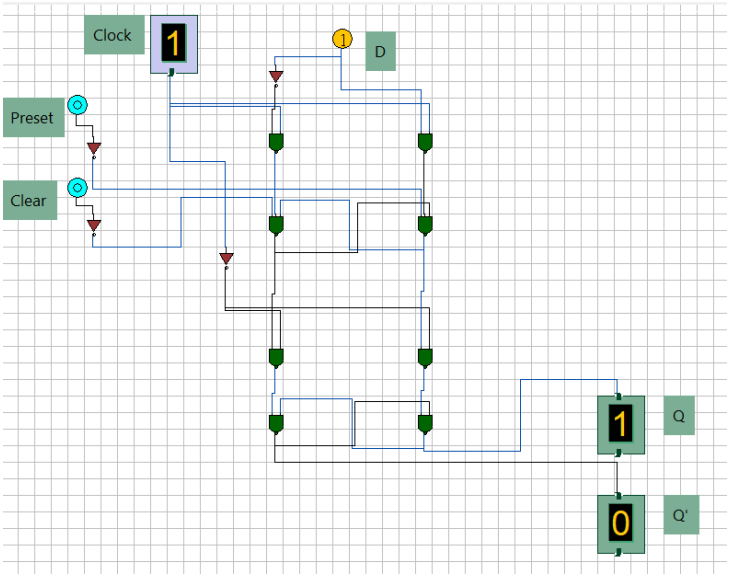

***GROUP 13***

| | |
|---|---|
| Chandra Pavan Sai Chowdary | 19CS10023 |
| Gurajala Joel Moses | 19CS10033 |
| Ravi Sri Ram Chowdary | 19CS30038 |
| Simhadri Praveena | 19CS30044 |
| Sirusolla Sri Bharath | 19CS30045 |

## Contents

# PART 1: D- Flip Flop

## Circuit Construction:




## Explanation:

- D-FF with preset and clear inputs to working on the negative edge of the clock is designed

### Circuit Description:

The operation of negative edge triggered Master Slave D flip flop is explained below:

When clock signal changes from low to high, the master flip flop stores the data from the D input. When clock signal goes high to low, the slave flip-flop will send the data got as its input as an output. And since the output is coming in the negative edge (From 1 to 0) it is a negative edge DFF)

### Preset and Clear:

D flip flop has another two inputs namely PRESET and CLEAR. A HIGH signal to CLEAR pin will make the Q output to reset that is 0. Similarly a HIGH signal to PRESET pin will make the Q output to set that is 1. Hence the name itself explains the description of the pins.

| Clock | INPUT | | | OUTPUT | |
|---|---|---|---|---|---|
| | PRESET | CLEAR | D | Q | Q’ |
| X | HIGH | LOW | X | 1 | 0 |
| X | LOW | HIGH | X | 0 | 1 |
| X | HIGH | HIGH | X | 1 | 1 |
| HIGH | LOW | LOW | 0 | 0 | 1 |
| HIGH | LOW | LOW | 1 | 1 | 0 |

- The clear bit makes the output Q to be 0 irrespective of D and Clock
- And similarly the Preset bit when made 1 makes the output 1 irrespective of the D and the clock.

## Simulation and Testing:

### Normal Operation:

- To simulate, give a value to D, and then set the values of the clear and the preset to LOW
- Start the clock and wait for some time, such that one rising edge followed by a negative edge comes from the clock.
- Immediately after the negative edge, the value of Q comes to be D

### Clear and Preset:

#### *Clear:*

- To simulate, set the value of the clear to HIGH
- Start the clock and wait for some time, such that one rising edge followed by a negative edge comes from the clock.
- We observe that irrespective of the values of the D and the state of the clock, the output is 0

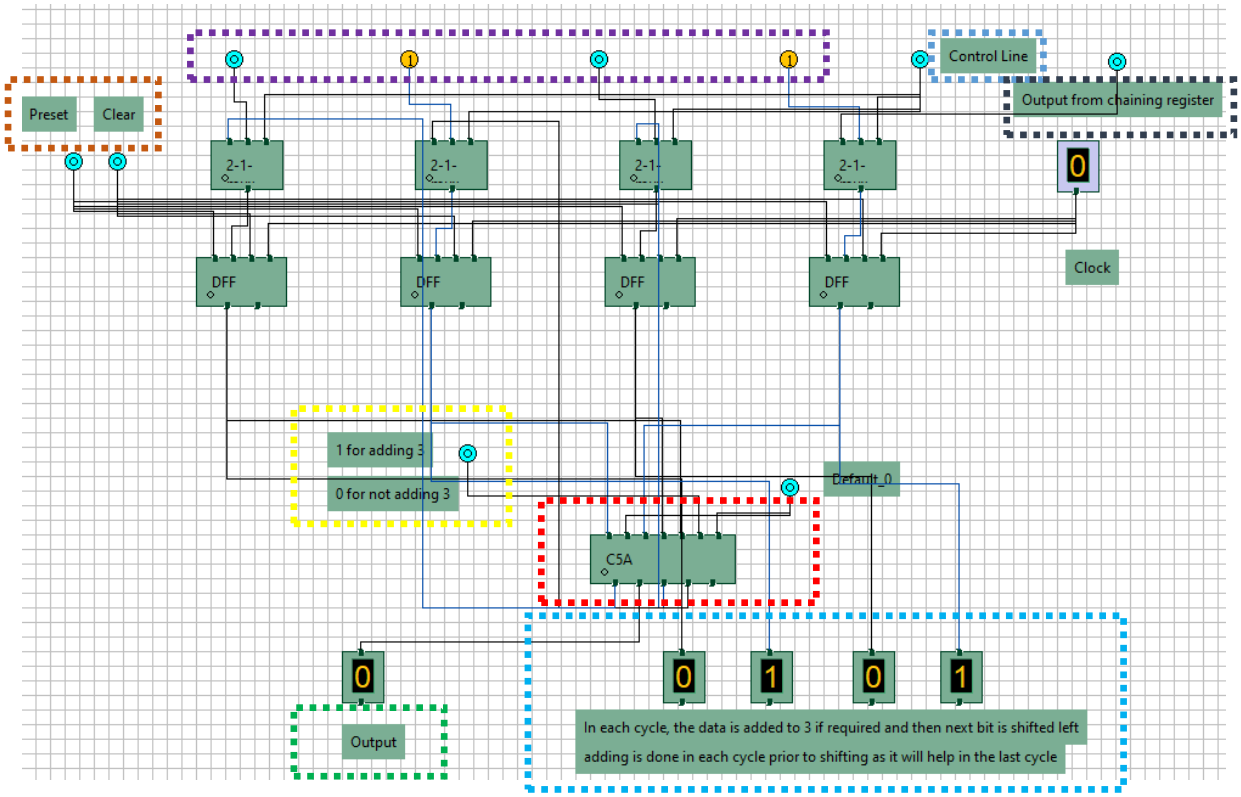
### Preset:

- To simulate, set the value of the preset to HIGH
- Start the clock and wait for some time, such that one rising edge followed by a negative edge comes from the clock.
- We observe that irrespective of the values of the D and the state of the clock, the output is 1

## Part 2: PISO Shift Register with additional features

In Parallel in Serial out (PISO) shift registers, the data is loaded onto the register in parallel format while it is retrieved from it serially. PISO shift register which has a control-line and combinational circuit (AND and OR gates) in addition to the basic register components (flip-flops) fed with clock, preset and clear pins.

### Circuit Construction:




### Detailed Explanation of Circuit:

- ***Control***: It is the control line that switches the behaviour of the register whether to shift the bits coming from their adjacent DFFs or the parallel input given through the MUX. This 'Control' controls the MUX and if 0, gives the parallel input to the registers and if 1, send the output of the previous DFF conditionally added with 3 if specified.
- ***Preset and Clear*** are the inputs to the DFFs
- ***Parallel Input*** is the 4 bit input sent as data into the DFF using the MUX if the control is zero
- ***Output*** is the serial output obtained from the register after doing the operations. Serial data means data comes bit by bit with the cycles of the clock
- ***The display*** of the data just before adding: the action done here is the before the data goes into the adder the display is shown (and hence can be detailed as output in the component) Say for example in a previous cycle (obviously considering when the adding condition is 1), 1010 is the data ended at the DFFs (i.e. sent as output by each of them) and one comes as input now, then 3 is added to this before shifting i.e. it becomes 1101 and then 1 s shifted.
- ***C5A*** is the adder that adds 3 (if it greater than 5 and ***adding conditioner is 1***) to the previous output from the DFFs and send it as input into the MUX at the top where parallel input / serial shift is done based on the control
- ***Output from chaining register*** is for chaining more registers and chaining makes them to behave like a big (more bits) single register

## Simulation and Testing:

### Initialisation:

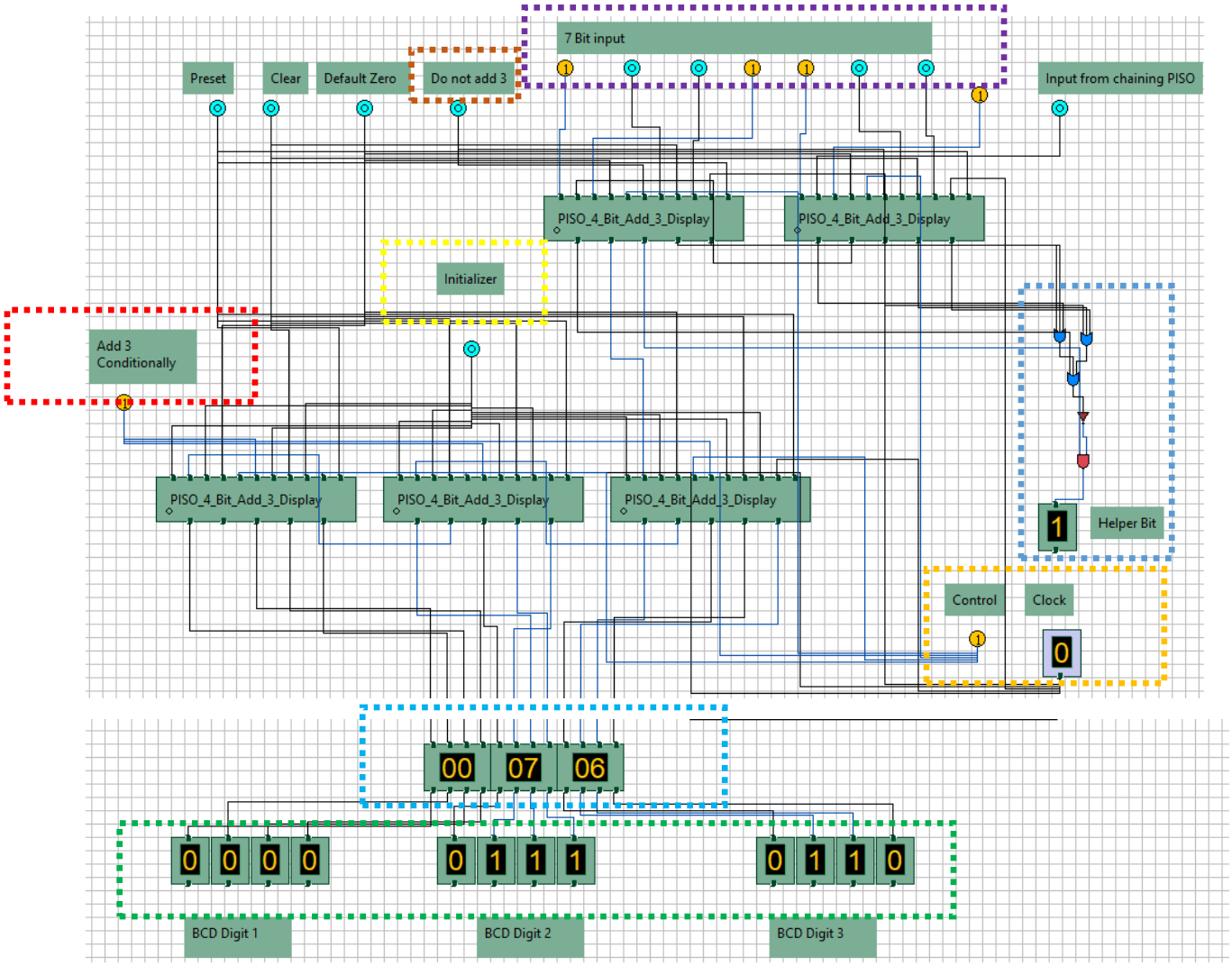
- To simulate, give values to the 4 input bits
- Set the values of the clear and the preset to LOW
- ***Make the control to zero***
- ***Start the clock*** and wait for some time, such that one rising edge followed by a negative edge comes from the clock.
  *(Immediately after the negative edge, in the above step, the data of the DFF gets initialised with the input data)*
- ***After the above step make the control to 1***
- Set the adding condition if required

All simulation is set and the shifting (also adding if 1) continues to take place and stop the clock after required time.

## Part 3: 7 Bit Double Dabble generator and Input reader

Double dabble algorithm refers to algorithm which converts a given binary number to its BCD representation which we are going to implement sequentially now.

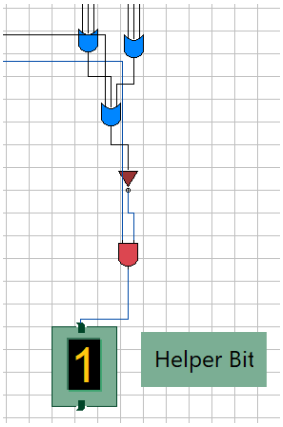
### Circuit Construction:




### Detailed Explanation of Circuit:

- ***Binary Input*** is the 7 bit input sent as data into the combined registers using the MUX if the control is zero (the 8th bit is used for other purpose which is described below)
- ***Do not add 3*** tells that the first two registers which correspond to getting input should not interfere in between regarding the conditional addition of 3 if >= 5
- ***Add 3 Conditionally*** tells the three registers below to take part in conversion of binary to double dabble using additions of 3 conditionally (if it greater than 5 ) to the previous output from the DFFs and send it as input into the MUX at the top where parallel input / serial shift is done based on the control
- ***Output*** is the BCD output of the given binary number obtained from the three registers after doing the operations. Here each digit of BCD is encoded in Binary
- ***The display*** of the above output is shown in decimal encoded digits in the BCD representation.

- ***Helper Bit***: It is a bit combinatorial computed from using the above 8th bit and the trailing "Input from chaining register which is zero since not being used". This will be 1 when required number of cycles are completed and all data (here 7 bits) have been transformed to BCD. Here the 8th bit set to 1 is used as follows:
    - When the helper bit computed by the following circuit is 1, then obviously it means that in the 8 bits of the above two registers, except the first bit all are 0 and the first bit is 1
    - So this condition arises only when the 8th bit of the initial state (1) and 7 zeroes come from the trailing input terminal
    - This means all the required 7 bits have been transferred and computed to BCD
    - By seeing this converting to 1, the user could stop the clock and hence helps him




    - The OR gates take last seven bits of present state of above registers and the and gate takes the first bit and the computed OR value
- ***Initialiser*** is a zero bit that makes the initialisation of states/bits in the register to zero initially (when control is zero)
- ***Control and Clock*** connects the control and clock of the five registers

## Simulation and Testing:

### Initialisation:

- To simulate, give values to the 7 input bits
- Set the values of the clear and the preset to LOW
- ***Make the control to zero***
- ***Start the clock*** and wait for some time, such that one rising edge followed by a negative edge comes from the clock.
  *(Immediately after the negative edge, in the above step, the data of the DFF gets initialised with the input data)*
- ***After the above step make the control to 1***
- Set the adding condition if required

All simulation is set and the conversion of binary to BCD completes and the helper bit changes to 1 seeing which the clock can be stopped manually.

## Notes:

1. It looks that we can make a FSM and use it for automating the switching between the 0 and 1 for the control line but the following scenario arises:
    a. If we want to make the FSM, we need memory and for that DFFs can be used
    b. But once we wish to use them, the initial states/memory of the FSM has to be initialised for which a similar concept of using a MUX and a control line is required.
    c. This leads to cyclic implementation which cannot be modelled by just using FSMs.

2. Also it feels like using a combinatorial circuit like OR and connect the Clock and the helper bit to it and use the output as the clock which helps as a automatically stopping clock, but that does not work as integrating clock with normal bits is giving unexpected behaviour and once we make that it becomes harder to run another cycle.

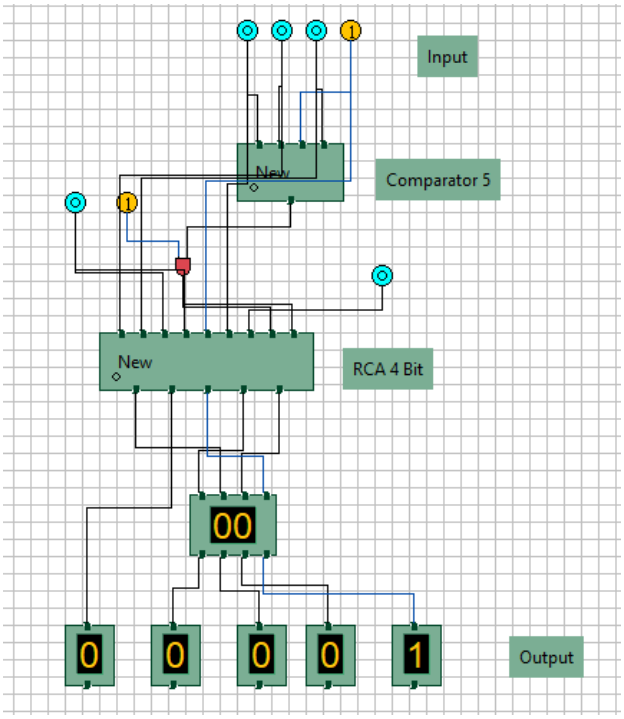
## Other Components Used:

### 1. 2-1 Multiplexer (MUX)

### 2. ≥5, +3 Adder

The RCA 4 Bit adder that has been designed as part of Assignment – 1 has been used to add the number conditionally (if ≥5) using the component ***Comparator 5*** (made in the previous part of the experiment) with three. Here three is obtained as constant just using the Bit Switches.

This circuit is saved as a component which takes a 4-Bit number and some default inputs such as 0, 1 for the making of 3 and $C_{in}$ which is not used but which is part of the RCA – 4 Bit adder used.




This is saved as a component ***Comparator 5 Add 3*** and is used in the upcoming parts of the experiment.

**THE END**