ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E
DIGITAL CIRCUITS LABORATORY HOMEWORK REPORT

HOMEWORK NO	:	2
LAB SESSION	:	FRIDAY - 16.00
GROUP NO	:	G18

GROUP MEMBERS:

150220770	:	Onur Baylam
150200916	:	Denis Iurıe Davıdoglu

SPRING 2023

Contents

1 INTRODUCTION

Memory units such as latches, flip-flops and their types are mainly focused on in this homework material. Moreover, we refer to differences between latches and flip-flops along with their operation characteristics and functionalities. In addition, truth tables of them are constructed to obtain a superior understanding to their working principle. In the experiment part, using our implementations of flip-flops and latches, we implement more complex structures in Verilog HDL such as asynchronous/synchronous counters and positive edge triggered pulse generator.

2 MATERIALS AND METHODS

2.1 Preliminary

1. **Explain what a flip flop is and why they are useful. Use your own words.**
   A flip-flop is a memory unit which has the ability of carrying 1-bit binary information. They are useful in terms of composing new memory units because it is simple and easy to implement. Moreover, with the help of clock cycle, we are able to synchronize flip-flops. This allows us to synchronize multiple flip-flops to create larger size of memory units such as registers which are used in CPUs.

2. **Explain what are the differences between latches and flip flops with your own words.**
   The main difference between a latch and a flip flop is that the first is asynchronous, and the second is synchronized with a clock. When, for instance, a set signal is given to latch, the time required to switch states will only depend on the internal delays of gate. Flip flops, however, are triggered either by positive or negative edge of clock signal, according to their type. Implementation of synchronization requires more logic gates.

3. **Briefly explain how an SR-latch works and what the functionalities of the input variables are.**
   SR latch has two control inputs, S and R. Also, its outputs Q and Q_N are fed back into the circuit and effective on next state outputs. According to the inputs S and R, SR latch performs some basic operations such as set, reset, preserve. For instance:
   When $S = 1$ and $R = 0$, SR latch sets Q to 1 (Set)
   When $S = 0$ and $R = 1$, SR latch resets Q to 0 (Reset)
   When $S = 0$ and $R = 0$, SR latch preserves its current state. (Preserve)

When $S = 1$ and $R = 1$, SR latch's outputs are undetermined. Thus, this input combination is forbidden.

4. **Construct the truth table of an SR latch which does not have an Enable input.**

S	R	$Q(t+1)$	$\overline{Q}(t+1)$	
0	0	$Q(t)$	$\overline{Q}(t)$	(preserve)
1	0	1	0	(set)
0	1	0	1	(reset)
1	1	x	x	(forbidden)

where $Q(t)$ and $Q(t+1)$, $\overline{Q}(t)$ and $\overline{Q}(t+1)$ represent previous and next states, and x is unknown.

5. **Construct the truth table of an SR latch which has an Enable input.**

E	S	R	$Q(t+1)$	$\overline{Q}(t+1)$	
0	ϕ	ϕ	$Q(t)$	$\overline{Q}(t)$	(disabled)
1	0	0	$Q(t)$	$\overline{Q}(t)$	(preserve)
1	0	1	0	1	(reset)
1	1	0	1	0	(set)
1	1	1	x	x	(forbidden)

where S and R are input signals, $Q(t)$ and $Q(t+1)$, $\overline{Q}(t)$ and $\overline{Q}(t+1)$ represent previous and next states, E is enable input, ϕ is don't care term and x is undetermined.

6. **Construct the truth table of a D flip flop.**

D	CLK	$Q(t+1)$	$\overline{Q}(t+1)$	
ϕ	ϕ	$Q(t)$	$\overline{Q}(t)$	(preserve)
1	↑	1	0	(set)
0	↑	0	1	(reset)

where $Q(t)$ and $Q(t+1)$, $\overline{Q}(t)$ and $\overline{Q}(t+1)$ represent previous and next states, ϕ is any state and ↑ symbol means rising edge, i.e. transition from low to high.

7. **Construct the truth table of a JK flip flop.**

J	K	CLK	$Q(t+1)$	$\overline{Q}(t+1)$	
ϕ	ϕ	0	$Q(t)$	$\overline{Q}(t)$	
ϕ	ϕ	1	$Q(t)$	$\overline{Q}(t)$	
0	0	$\uparrow$	$Q(t)$	$\overline{Q}(t)$	(preserve)
0	1	$\uparrow$	0	1	(reset)
1	0	$\uparrow$	1	0	(set)
1	1	$\uparrow$	$\overline{Q}(t)$	$Q(t)$	(toggle)

where J and K are input signals, $Q(t)$ and $Q(t+1)$, $\overline{Q}(t)$ and $\overline{Q}(t+1)$ represent previous and next states, ϕ is don't care term and $\uparrow$ symbol means rising edge, i.e. transition from low to high.

2.2 Experiment

- **Part 1 - SR Latch (only NAND Gates)**

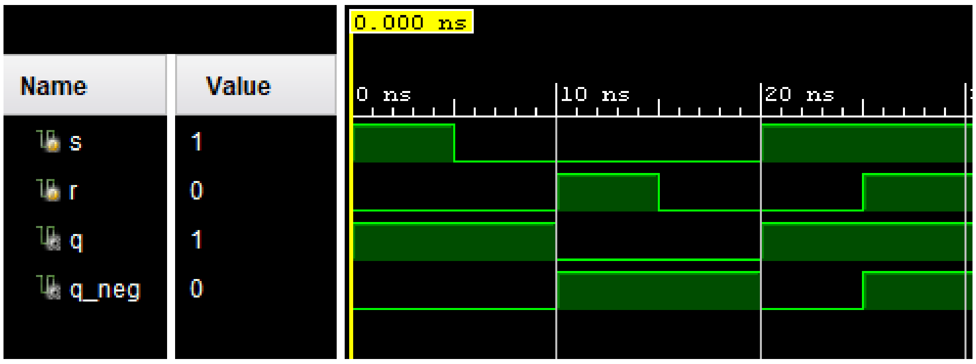


Figure 1: SR Latch simulation

The behaviour of SR Latch can be expressed with an equation of form $Q(t+1) = f(S; R; Q(t))$. From the truth table, all combinations of S and R are selected except the forbidden state, and $Q(t+1)$ is expressed and simplified using rules of Boolean algebra as follows:

$$\begin{aligned}
Q(t+1) &= \bar{S}\bar{R}Q(t) + \bar{S}R\cdot 0 + S\bar{R}\cdot 1 && \text{(Truth table)}\\
&= \bar{S}\bar{R}Q(t) + 0 + S\bar{R}\cdot 1 && \text{(Dominance)}\\
&= \bar{S}\bar{R}Q(t) + S\bar{R} && \text{(Identity)}\\
&= \bar{S}\bar{R}Q(t) + S\bar{R} + \bar{R}\bar{R}Q(t) && \text{(Consensus theorem)}\\
&= \bar{S}\bar{R}Q(t) + S\bar{R} + \bar{R}Q(t) && \text{(Idempotency)}\\
&= Q(t)(\bar{S}\bar{R} + \bar{R}) + S\bar{R} && \text{(Distributivity)}\\
&= Q(t)\bar{R} + S\bar{R} && \text{(Absorption)}\\
&= Q(t)\bar{R} + S
\end{aligned}$$

In the final step, $\bar{R}$ is omitted because state $S = 1, R = 1$ is forbidden and should never happen. Therefore omitting it does not change the truth table. To summarize, SR Latch's output is 1 whenever S is on and R is off, is 0 whenever R is on and S is off, and it maintains the previous state if both S and R go low. It is therefore called a memory circuit.

- **Part 2 - SR Latch with Enable input (only NAND Gates)**
  In our implementation of SR latch module with Enable (E) input, it is sufficient to use only NAND gates which are one of universal gates. However; before investigating the simulation results, we should write the characteristic equation of SR latch with Enable input. To achieve this, we benefit from the truth table we have constructed in the Preliminary section.

  Characteristic Equation of SR Latch with Enable:

$$\begin{aligned}
Q(t+1) &= \bar{E}Q(t) + E\bar{S}\bar{R}Q(t) + E\bar{S}R\cdot 0 + ES\bar{R}\cdot 1 && \text{(Truth table)}\\
&= \bar{E}Q(t) + E\bar{S}\bar{R}Q(t) + ES\bar{R} && \text{(Dominance)}
\end{aligned}$$

  Simulation Result of SR Latch with E:

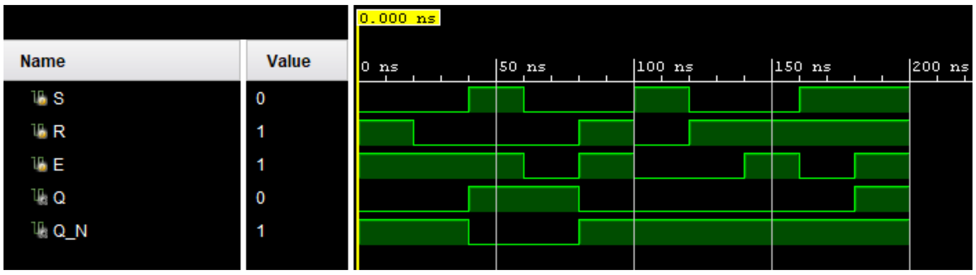


Figure 2: Simulation Result of SR Latch with E

For disallowed inputs $S = 1$ and $R = 1$, our SR latch with E gives the result as $Q(t+1) = 1$ and $Q_N(t+1) = 1$ which is observable by examining the simulation result. Since both Q and Q_N are 1, it indicates that SR latch does not work correctly for $S = 1$ and $R = 1$.

Main difference between SR Latch and SR Latch with E is the **Enable** input. When $E = 0$, SR Latch's output values are not affected by input values. We can also see that in simulation results.

- **Part 3 - D Flip-Flop from D-Latches**

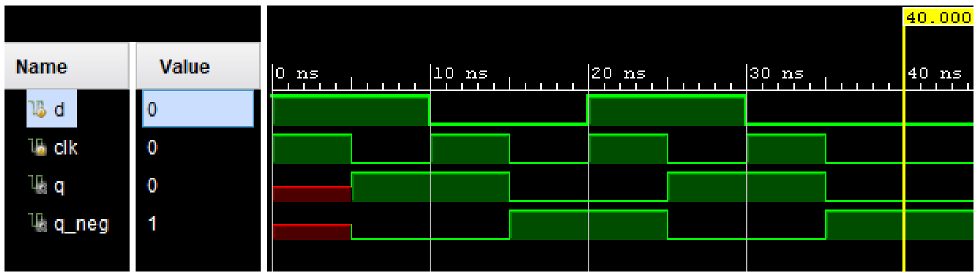


Figure 3: Negative edge triggered D flip-flop simulation

As it is seen from the wave form, D flip-flop can change state only on the falling clock edge.

- **Part 4 - JK Flip-Flop (only NAND gates)**

  Positive edge triggered JK Flip-Flop module has four input ports: J, K, $reset$, clk and two output ports: Q, Q_N.

NOTE: When $reset$ is active, Q is set to 0 regardless of J and K.

Simulation Result of JK Flip-Flop:

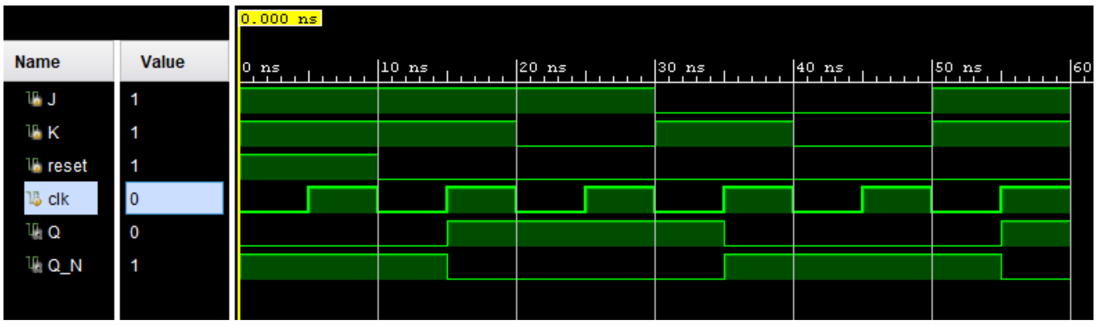


Figure 4: JK Flip-Flop Simulation Result

- **Part 5 - Asynchronous Up Counter (JK flip flop)**

  Our module of 4-bit asynchronous up counter has two inputs: *clk*, *reset* and one 3-bit output named *out*. We use input *reset* in order to reset the counter when it reaches 15. So, the counter never reaches 15.

Simulation Result of Asynchronous Up Counter:

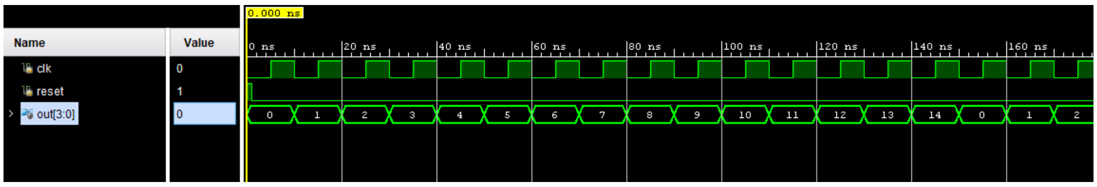


Figure 5: Simulation Result of 4-bit Asynchronous Up Counter

The truth table is as follows:

CLK	Reset	$\{A_3, A_2, A_1, A_0\}$	$\{A_3, A_2, A_1, A_0\}^+$
ϕ	1	$\{\phi, \phi, \phi, \phi\}$	$\{0, 0, 0, 0\}$
↓	0	$\{0, 0, 0, 0\}$	$\{0, 0, 0, 1\}$
↓	0	$\{0, 0, 0, 1\}$	$\{0, 0, 1, 0\}$
↓	0	$\{0, 0, 1, 0\}$	$\{0, 0, 1, 1\}$
↓	0	$\{0, 0, 1, 1\}$	$\{0, 1, 0, 0\}$
↓	0	$\{0, 1, 0, 0\}$	$\{0, 1, 0, 1\}$
↓	0	$\{0, 1, 0, 1\}$	$\{0, 1, 1, 0\}$
↓	0	$\{0, 1, 1, 0\}$	$\{0, 1, 1, 1\}$
↓	0	$\{0, 1, 1, 1\}$	$\{1, 0, 0, 0\}$
↓	0	$\{1, 0, 0, 0\}$	$\{1, 0, 0, 1\}$
↓	0	$\{1, 0, 0, 1\}$	$\{1, 0, 1, 0\}$
↓	0	$\{1, 0, 1, 0\}$	$\{1, 0, 1, 1\}$
↓	0	$\{1, 0, 1, 1\}$	$\{1, 1, 0, 0\}$
↓	0	$\{1, 1, 0, 0\}$	$\{1, 1, 0, 1\}$
↓	0	$\{1, 1, 0, 1\}$	$\{1, 1, 1, 0\}$
↓	0	$\{1, 1, 1, 0\}$	$\{0, 0, 0, 0\}$

- **Part 6 - Synchronous Up Counter (JK flip flop)**
  Similarly, 4-bit synchronous up counter module has two inputs: *clk*, *reset* and one 3-bit output named *out*. In order to reset the counter when it reaches 15, we use the *reset* port.

Simulation Result of Synchronous Up Counter:

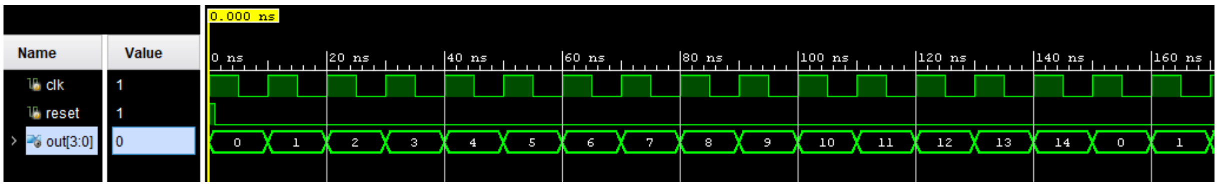


Figure 6: Simulation Result of 4-bit Synchronous Up Counter

The truth table is as follows:

CLK	Reset	$\{A_3, A_2, A_1, A_0\}$	$\{A_3, A_2, A_1, A_0\}^+$
ϕ	1	$\{\phi, \phi, \phi, \phi\}$	$\{0, 0, 0, 0\}$
↑	0	$\{0, 0, 0, 0\}$	$\{0, 0, 0, 1\}$
↑	0	$\{0, 0, 0, 1\}$	$\{0, 0, 1, 0\}$
↑	0	$\{0, 0, 1, 0\}$	$\{0, 0, 1, 1\}$
↑	0	$\{0, 0, 1, 1\}$	$\{0, 1, 0, 0\}$
↑	0	$\{0, 1, 0, 0\}$	$\{0, 1, 0, 1\}$
↑	0	$\{0, 1, 0, 1\}$	$\{0, 1, 1, 0\}$
↑	0	$\{0, 1, 1, 0\}$	$\{0, 1, 1, 1\}$
↑	0	$\{0, 1, 1, 1\}$	$\{1, 0, 0, 0\}$
↑	0	$\{1, 0, 0, 0\}$	$\{1, 0, 0, 1\}$
↑	0	$\{1, 0, 0, 1\}$	$\{1, 0, 1, 0\}$
↑	0	$\{1, 0, 1, 0\}$	$\{1, 0, 1, 1\}$
↑	0	$\{1, 0, 1, 1\}$	$\{1, 1, 0, 0\}$
↑	0	$\{1, 1, 0, 0\}$	$\{1, 1, 0, 1\}$
↑	0	$\{1, 1, 0, 1\}$	$\{1, 1, 1, 0\}$
↑	0	$\{1, 1, 1, 0\}$	$\{0, 0, 0, 0\}$

- **Part 7**

  To implement pulse width modulation with a circular shift register, there must be a multiplexer which will choose between the input wave pattern and output of previous flip flop, and output the signal as input to the next flip flop. When load flag is low, flip flops behave like a circular shift register. When load flag becomes high, circular transmission is stopped and all flip flops are written to. This is the design of 2:1 multiplexer we used:

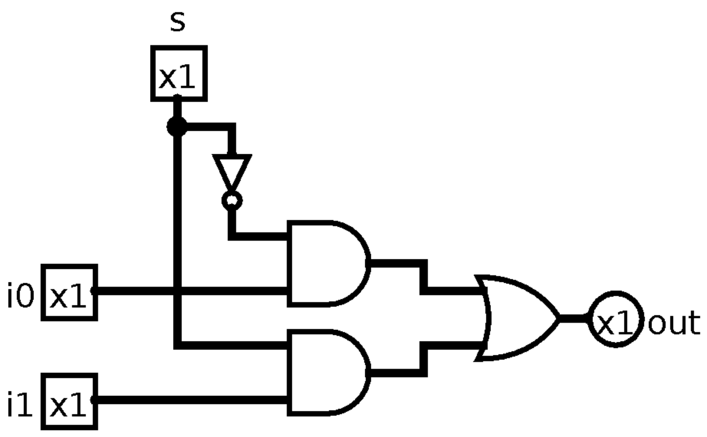
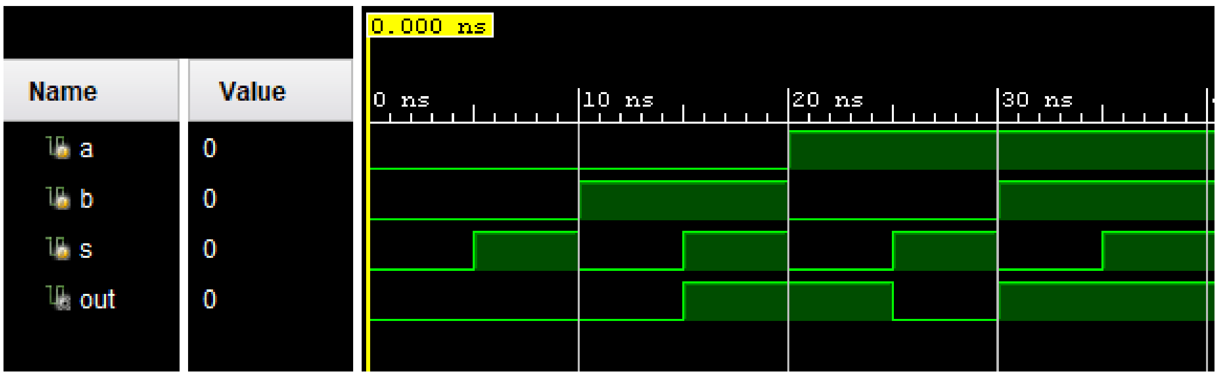


Figure 7: 2:1 multiplexer design in Logisim

It was also described and simulated with Verilog:



Name	Value
a	0
b	0
s	0
out	0

Figure 8: 2:1 multiplexer simulation in Verilog

The pulse generator itself is too long to fit on the page, and with that reason the following diagram is it's smaller version with same behaviour. Because D flip flops are negative edge triggered, the clock is inverted in order to make it a positive edge triggered device:

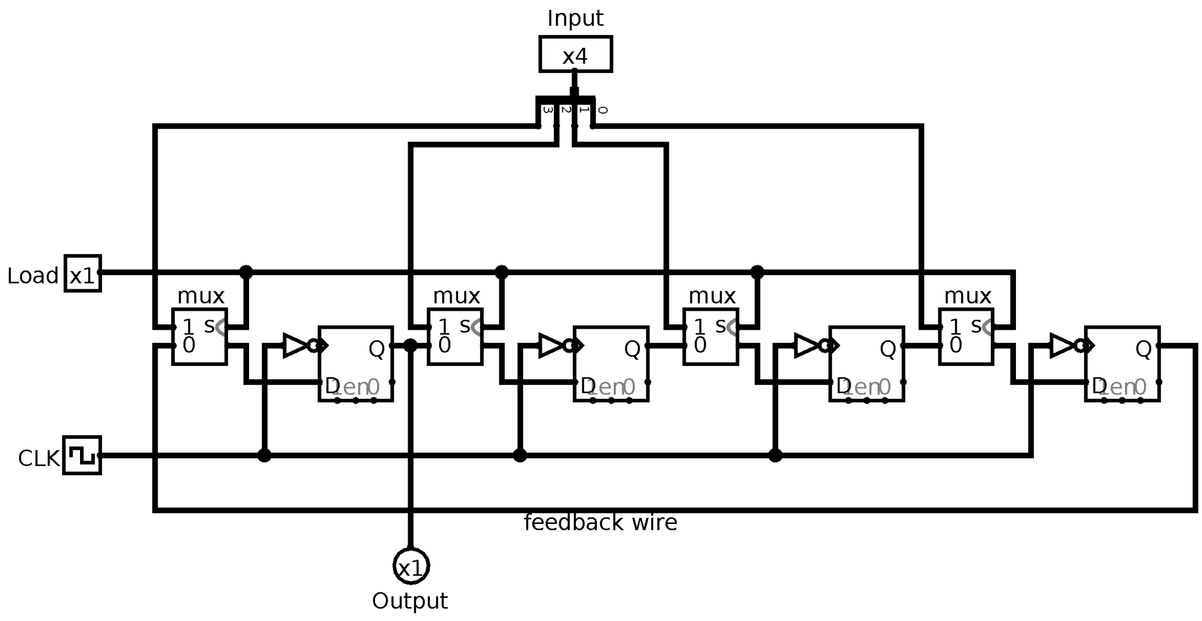


Figure 9: Pulse generator design in Logisim

In the Verilog description we prepared, pulse generator has a 16-bit input as required. The result of simulation confirms it can deliver various pulses, such as half, quarter, one eigth of clock frequency and signals with 1/7, 3/13, 11/5 pulse-gap duration rate:

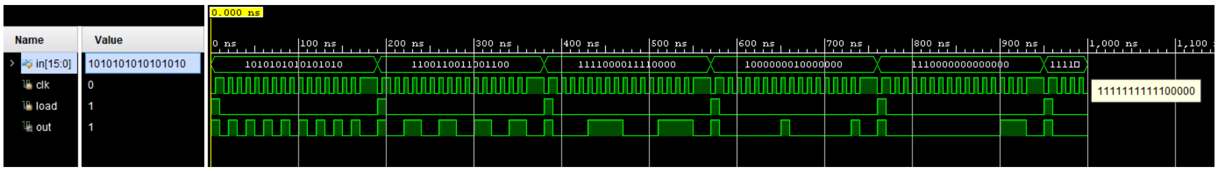


Figure 10: Pulse generator simulation in Verilog

3 CONCLUSION

As a consequence, flip-flops and latches are fundamental memory units which can be used as a powerful toolset. By understanding their internal structure and working principle properly, it is possible to create a wide range of more complex circuits such as registers which are highly utilized in CPUs. Therefore, we first implement flip-flops and latch memory units in Verilog HDL, then, to have a complete understanding on these digital circuits, we realize simulations with different input combinations and interpret the results, since these simulations enables us to detect possible errors in advance and fix them easily. So, we need to make sure that our mostly used modules such as JK and D flip flops are built correctly to prevent future problems in our further designs. On the other hand, one of the main difficulties we faced during implemantation phases is to specify the initial values of latches or flip-flops before the clock pulse, it is critically important in terms of having the ability to toggle the value of flip-flops. For instance, in our 4-bit asynchronous up counter module, we need to set both J and K to 1 which means toggling the current output, but if our current value is undetermined, we cannot toggle our current output. To overcome these problems, we use *reset* signal to define an initial value for our memory units.