Experiment–8: Design of Flipflops using HDL

Digital Electronics Circuits Laboratory (EC2P004)
School of Electrical and Computer Sciences, IIT Bhubaneswar

Agenda of the Experiment

In this experiment, we will do the following:

- Implement a D flipflop.
- Implement a T flipflop using D flip-flop.
- Implement a JK flip-flop using D flip-flop.
- Design a simple clock divider to convert 100MHz to 1 Hz.

Behavioral modeling

- Behavioral modeling is used to describe logic using expressions, tables,etc. and is useful when the internal circuit is unknown or too complex. For sequential circuits
- All behavioral expressions are written inside an **always** block which executes and reexecutes indefinitely.
- The activity associated with behavioral statement can be controlled by a event control operator '@' that waits for certain event. An event can be an unconditional change in signal value @(A) or a specified transition of signal value @(posedge clock).
- **always** @(event control expression) **begin**
  // Procedural assignment statements that execute when the condition is met
  end
- Since the procedural statements does not get executed continuously, variables on the lhs of procedural statements must be **reg** type.
- The event control operation is also called as the sensitivity list.

Behavioral modeling

- There are two kinds of procedural assignments. Blocking and non-blocking. While describing sequential circuits, we use non-blocking assignments.
- $B <= A$ assigns the value of A to B. Non-blocking statements are preferred for sequential circuits since they model the concurrent operations of a physical hardware.
- Reading exercise: Section 5.6 of “Digital Design with an Introduction to Verilog HDL” by M. Moris Mano and Michael D. Ciletti.

Part 1: D Flip-flop

D Flip-Flop

D	$Q(t + 1)$
0	0
1	1

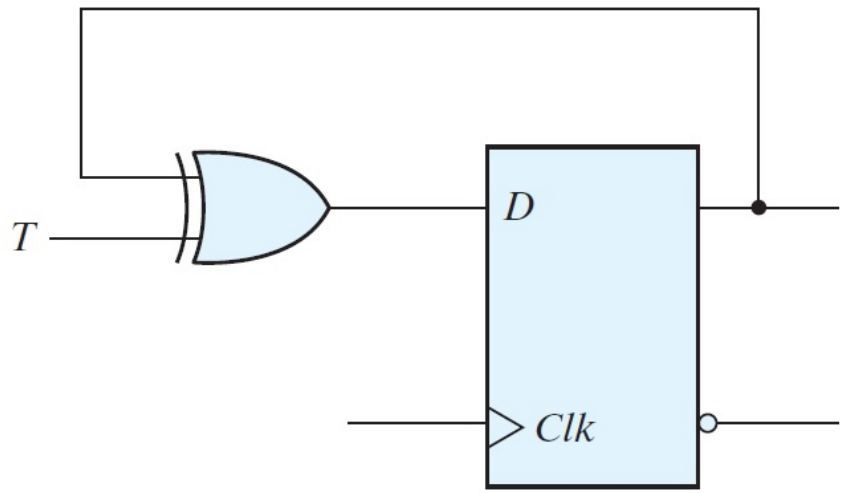
- The characteristic table of a D flip-flop is as given above. The output Q follows the input after a clock pulse.
- To help you get started with implementation a sketch of HDL code for D flip-flop is provided in the next page.
- Complete the code in next page. Use switches for *d* and *rst* and the clock signal constraint is found on the top of the constraint file. Uncomment three lines for the clock signal. Observe the output of the flip-flop on LED's. Show the output to TA's.

Skeletal HDL Code for D Flip-flop

```
module myDff(output reg q, output reg qbar, input d, input clk, input rst);
always @(posedge clk) //responds to the positive edge of clock
begin
if (rst==0)
begin
q <= 0;
qbar <= 1;
end
else
begin
.
.
end
end
endmodule
```

Part 2: T flip-flop using D flip-flop

- Now that you have a D flip-flop let us make use of it to construct a T flip-flop.



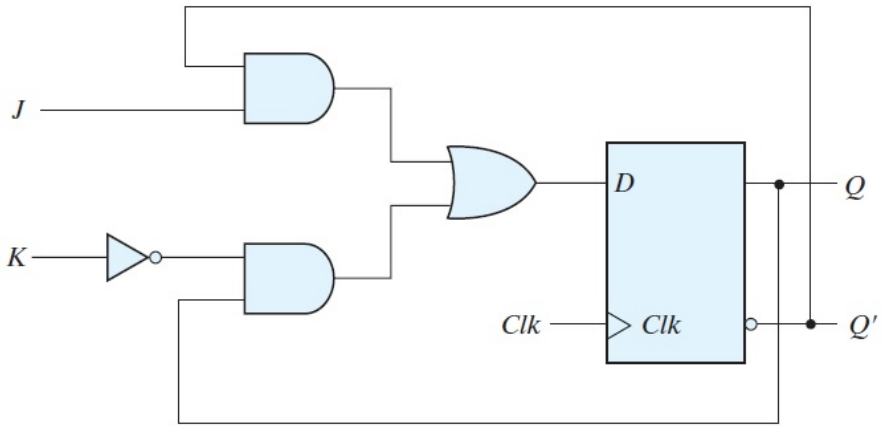
- Use switches for T and rst and the clk signal and use a D flip-flop to implement a T flip-flop. Show the output to TA's.
- You can notice that when $T = 1$ and $rst = 1$, we get an unexpected output. The reason is because the clock frequency of the Basys3 board is 100 MHz and the flip-flop is toggling at that frequency. Human eye has its limitations. This problem is solved by using either a manual clock or by a clock divider circuit.

Part 3: D flip-flop using manual clock

- In this part, we design a positive edge-triggered D flip-flop with an active low synchronous reset feature. (i.e, the reset will be considered only during a clock edge).
- The flip-flop should have D, rst and clk as inputs and output Q and $\overline{Q}$
- Use a push button switch for clk (manual clocking). There are five buttons available on the Basys3 board. Appropriately modify the xdc file. Search for buttons. Use normal switches for D and rst. Observe the outputs Q and $\overline{Q}$ on LED's.
- Whenever you put any signal name in the sensitivity list in the always block, Vivado software treats it as default clock signal, and it is likely that there will be an error in the implementation step.
- Add the following line in your constraint file
  set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets <signalname >_IBUF]
- Specifically add the below line.
  set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets clk_IBUF]
- Show the output to the TA's.

Part 4: T and JK flip-flop using manual clock

- Using the constructed *D* flip-flop, construct a T flip-flop. The flip-flop should take T, rst and clk as inputs and provide Q and $\overline{Q}$ as outputs. Show the output to TA's.
- Using the constructed *D* flip-flop, construct a *JK* flip-flop. The flip-flop should take J, K, clk and rst as inputs and provide Q and $\overline{Q}$ as outputs. Verify that you get the characteristic table given below. Show the output to TA's.



J	K	Q(t + 1)
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$Q'(t)$

Part 5: Clock Divider

- A counter is used as a frequency divider. You will learn more about this in class. The complete code is provided here for your understanding. Implement and test the code. In the constraint file, assign the clock input to the 100MHz clock, and assign the output to an LED.

```
module clkDivider(
input clk,
output reg clkOut
);
reg [25:0] count;//an internal register, just like a variable
always @(negedge clk)//responding to the positive edge of clock
begin
count <=count + 1;
if (count == 50000000)
begin
clkOut <= ~ clkOut;//change of state of clock after every 0.5 sec
count <= 0;
end
end
endmodule
```

Save this code; we are going to use it next week.

Report and additional work

- Verilog codes for all parts with proper comments.
- A brief conclusion about what you have learned from each part of the experiment.
- In the experiment, the reset was given as a synchronous input. We desire it to be asynchronous. What change needs to be made in the verilog HDL. Include this in your report. If time permits, verify this in the lab itself.
- In the report, include the behavioral description of a JK flip-flop. If time permits implement and verify this in the lab.