

Tutorials Home

VERILOG BASIC TUTORIAL

Verilog Introduction

Installing Verilog and Hello World

Simple comparator Example

Code Verification

Simulating with verilog

Verilog Language and Syntax

Verilog Syntax Contd..

Verilog Syntax - Vector Data

Verilog \$monitor

Verilog Gate Level Modeling

Verilog UDP

Verilog Bitwise Operator

Viewing Waveforms

Full Adder Example

Multiplexer Example

Always Block for Combinational ckt

if statement for Combinational ckt

Case statement for Combinational ckt

Hex to 7 Segment Display

casez and casex

full case and parallel case

Verilog for loop

Verilog localparam and parameter

SEQUENTIAL CKT TUTORIAL

Sequential Circuits

Serial Shift Register

Binary Counters

Ring Counter

MISC VERILOG TOPICS

Setting Path Variable

Verilog Tutorial Videos

Verilog Interview questions #1

Verilog Interview questions #2

Verilog Interview questions #3

Verilog Books

Synchronous and Asynchronous Reset

Left and Right shift &lt;&lt; and &gt;&gt;

Negative Numbers

Blocking Vs Non Blocking

wand and wor

## Sequential Circuits

So far we have only been discussing combinational circuits. This was good as far as our learning of Verilog language and its constructs are concerned. Practical FPGA circuits, however, almost always contain sequential circuits.

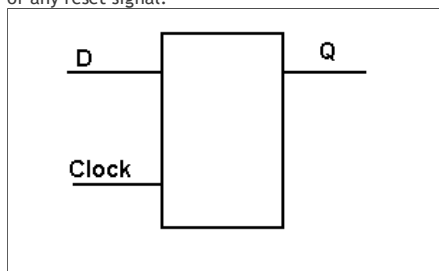
Combinational circuits do not have memory and its present output is a function only of present inputs. A sequential circuit, on the other hand, has memory and its present output depends not only upon present input but also upon past input(s).

A better term for past inputs is "state". A sequential circuit consists of finite states and its output depends upon present input and one of these states.

Implicit in the design of the sequential circuits is a global clock and the circuit operates on the rising or falling edge of the clock.

### D Flip Flop

A D Flip Flop is the most basic building block of sequential circuit. From the abstraction at the top level, a D Flip Flop has an Clock and a Data D as input. It has one output designated as Q. For simplicity we do not assume presence of any reset signal.



This D Flip Flop functions as follows

1. When the Clock stays low or the clock stays high, the output does not change, it stays at its previous value.
2. When the Clock edge changes from low to high, the Output Q gets the value of the input D.

Here is the verilog implementation of D Flip Flop.

```

1. // referencedesigner.com
2. // D Flip Flop without Reset
3.
4. module d_ff
5. (
6.     input wire Clock,
7.     input wire D,
8.     output reg Q
9. );
10.
11.     always @(posedge Clock)
12.         Q = D;
13. endmodule

```

The new thing to note is the line

```
always @(posedge Clock)
```

that triggers the positive edge of clock event in the sensitivity list.

The posedge keyword specifies the direction of the clock signal changing from 0 to 1. Similarly, negedge keyword specifies the clock direction from 1 to 0.

This D flip flop is a positive edge-triggered FF. An important thing to note is that the input signal D is not present in the sensitive list. The D signal is sampled only at the rising edge of the clk signal.

Let us now write a test bench for the D Flip flop and verify its behavior. We will also add capability to see its waveform in GTK wave.

```
1. `timescale 1ns / 1ps
```

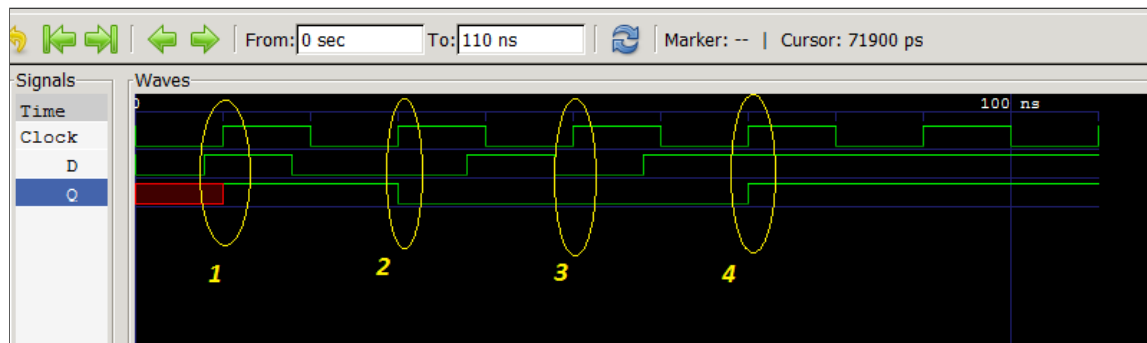
[delay in verilog](#)  
[\\$dumpfile and \\$dumpvars](#)  
[Useful Resources](#)  
[Verilog Examples](#)  
**VERILOG QUIZS**  
[Verilog Quiz # 1](#)  
[Verilog Quiz # 2](#)  
[Verilog Quiz # 3](#)  
[Verilog Quiz # 4](#)  
[Verilog Quiz # 5](#)  
[Verilog Quiz # 6](#)  
[Verilog Quiz # 7](#)  
[Verilog Quiz # 8](#)  
[Verilog Quiz # 9](#)  
**OTHER TUTORIALS**  
[Verilog Simulation with Xilinx ISE](#)  
[VHDL Tutorial](#)

```

2. module stimulus;
3.     reg Clock;
4.     reg D;
5.     wire Q;
6.     // Instantiate the Unit Under Test (UUT)
7.     d_ff d1 (
8.         .Clock(Clock),
9.         .D(D),
10.        .Q(Q)
11.    );
12.
13.    integer i;
14.
15.    initial begin
16.        $dumpfile("test.vcd");
17.        $dumpvars(0,stimulus);
18.        D = 0;
19.        #8 D= 1;
20.        #10 D= 0;
21.        #10 D= 0;
22.        #10 D =1;
23.        #10 D =0;
24.        #10 D = 1;
25.        #40;
26.    end
27.
28.    initial begin
29.        Clock = 0;
30.        for ( i =0; i <=10; i= i+1)
31.            #10 Clock = ~Clock;
32.    end
33.
34.
35.    initial begin
36.        $monitor("Clock=%d,D=%d,Q=%d \n",Clock,D,Q);
37.    end
38.
39. endmodule

```

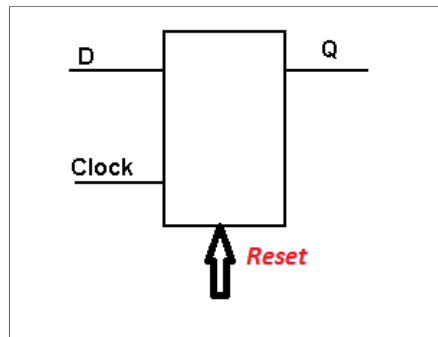
One thing you will note that there are two initial statements in the stimulus. One that contains the clock and other that contains Data. It is a good idea to let the clock run freely using a for loop etc. And then we can focus on change in the Data. We have randomly assigned some values to Data that changes ( or does not change) 2 ns before the rising edge of the clock . Look at the waveform generated using this stimulus



1. At rising edge of clock marked 1 - the Data D Input is 1, hence Q becomes 1 right after that. It is undefined x before that.
2. At rising edge of clock marked 2, the Data is 0, hence Q becomes 0 right after the clock rising edge.
3. At 3, the Data is again 0, hence Q stays 0.
4. At 4, Data is 1 at the rising edge of clock, hence Q becomes 1

### D Flip Flop with Asynchronous Reset

A D Flip Flop with Asynchronous Reset adds another Reset Input.



At the falling edge of the reset, the Output Q is reset to zero. This happens irrespective of the clock value. In other words, the reset is async.

The following is the implementation of D Flip Flop with asynchronous reset.

```

1. // referencedesigner.com
2. // D Flip Flop with Asynchronous Reset
3.
4. module d_ff
5. (
6.     input wire Clock,
7.     input wire D,
8.     input Rst,
9.     output reg Q
10. );
11.
12. always @(negedge Rst or posedge Clock )
13. begin
14.     if (!Rst)
15.         Q = 0;
16.     else
17.         Q = D;
18.     end
19. endmodule

```

We will leave it as an exercise to write a testbench and verify the functionality of D Flip Flop with asynchronous reset.

### D Flip Flop with Synchronous Reset

A D Flip Flop with Synchronous Reset also allows the reset, but the reset takes place only at clock edge. Let us say, we are allowing reset at its negative edge and the effect takes place at positive edge of clock.

The following is the implementation of D Flip Flop with synchronous reset in verilog.

```

1. // referencedesigner.com
2. // D Flip Flop with Synchronous Reset
3.
4. module d_ff
5. (
6.     input wire Clock,
7.     input wire D,
8.     input Rst,
9.     output reg Q
10. );
11.
12. always @(posedge Clock )
13. begin
14.     if (!Rst)
15.         Q = 0;
16.     else
17.         Q = D;
18.     end
19. endmodule

```

< Previous

Next >