CSC3050 Tutorial

# Sequential Logic And Combinational Logic in Verilog

# Outline

- Sequential logic and combinational logic in Verilog
  - Combinational Logic Circuit With "Assign" Statement;
  - Combinational Logic Circuit With "Always" Block;
  - Sequential Logic Circuit With "Always" Block;
- Blocking and Non-blocking Assignments
- Example of D flipflop.

## Differences Between Combinational & Sequential Logic

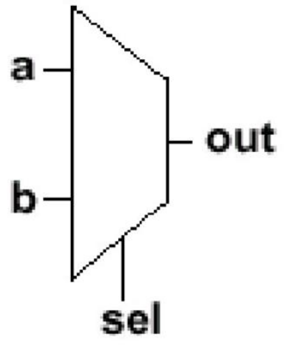
- Combinational logic circuit
  - Output at any time only depends on the input.
  - It has nothing to do with the circuit state. (Stateless)

- Sequential logic circuit
  - Output depends on both the input and the circuit state at current time.

## Combinational Logic Circuit With “Assign” Statement

**Take two-way selector as example:**

```
module mux2to1(a, b, sel, out);
      input  a, b;
      input  sel;
      output out;
      assign out = (sel)? b : a;
endmodule
```

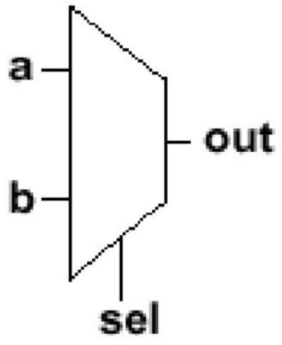


**Note:**
**Only wire type data can be assigned value in the “Assign” statement.**

## Combinational Logic Circuit With "Always" Block

**Still take two-way selector as example:**

```
module mux2to1(a, b, sel, out);
        input  a, b;
        input  sel;
        output out;
        reg out;
        always @ (sel or a or b)
              out = (sel) ? b : a;
endmodule
```



**Note:**

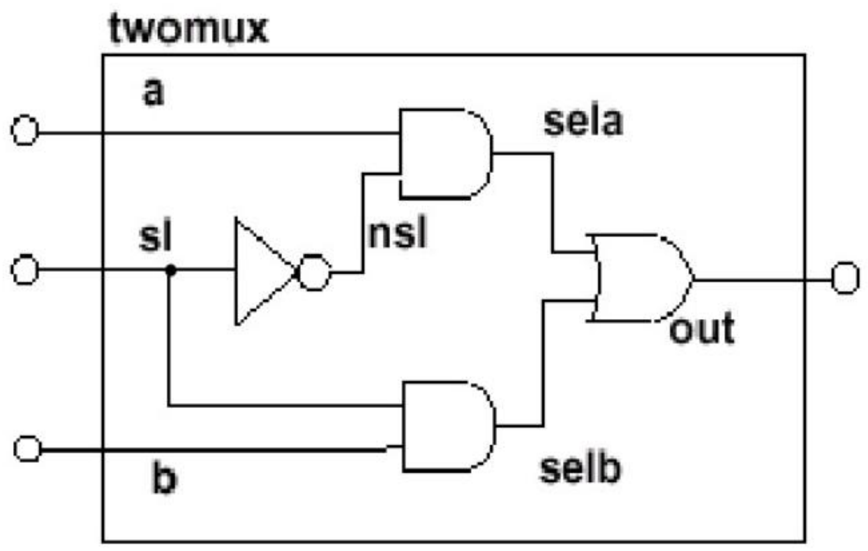
**1: Only reg type data can be assigned values in "Always" block.**

**2: Combinational logic circuit implemented by "Always" block is level triggered.**

## Combinational logic

**The effect of combinational logic described by “always” block and “assign” statement is the same.**

**The former 2 ways to define two-way selector will generate the same circuit as follows.**



## Sequential Logic Circuit With “Always” Block

```
module DFF(clk, rst, d, q);
      input  clk, rst, d;
      output q;
      reg q;
      always @ (posedge clk) begin // posedge or negedge
           if(rst)
               q <=1’b0;
           else
               q <= d
      end
endmodule
```

**Note:**

**1: Sequential logic circuit implemented by “Always” block is edge triggered.**

**2: Describing sequential logic circuit can only use non-blocking assignments.**

# Remark: differences between blocking and non-blocking assignments

- Non-blocking Assignment
  - use "<=".
  - Only in "Always" block
  - All assigning statements calculate their right value simultaneously, then update their left value.
- Blocking assignment
  - use "="
  - Sequential execution in an "Always" block

```
wire a;
reg b, c;
assign a = 1'b1;
always @(posedge clk or negative reset)
    if(!reset) begin
         b <= 0;
         c <= 1;
    end
    else begin
         b <= a;
         c <= b;
    end
//when the first signal comes, b=0, c=1;
//when the second signal comes, b=1, c=0.
```

Non-blocking

```
wire a;
reg b, c;
always @(a or b)
      begin
           b = a;
           c = b;
      end
```

**Suppose initial values: a=b=c=0, when a = 1, b = c = 1.**

Blocking

## Blocking Assignments Collision

```
module fbosc1 (y1, y2, clk, rst);

output reg y1, y2;

input clk, rst;


always @(posedge clk or negedge rst)

begin

if (!rst) y1 = 0; // reset

else y1 = y2;

end


always @(posedge clk or negedge rst)

begin

if (!rst) y2 = 1; // preset

else y2 = y1;

end

endmodule
```

After Reset, y1 = 0, y2 = 1;

Suppose "else" statement in the first "always" block is executed first, then we get y1 = y2 = 1.

If "else" statement in the second "always" block is executed first, then y1 = y2 = 0.

Collision leads to instability.

## Exercise: Nonblocking assignment

```
module fbosc1 (y1, y2, clk, rst);

output reg y1, y2;

input clk, rst;


always @(posedge clk or negedge rst)

begin

if (!rst) y1< = 0; // reset

else y1< = y2;

end


always @(posedge clk or negedge rst)

begin

if (!rst) y2 <= 1; // preset

else y2 <= y1;

end

endmodule
```

After Reset, y1 = 0, y2 = 1

## Exercise: Nonblocking assignment

```
module fbosc1 (y1, y2, clk, rst);

output reg y1, y2;

input clk, rst;


always @(posedge clk or negedge rst)

begin

if (!rst) y1< = 0; // reset

else y1< = y2;

end


always @(posedge clk or negedge rst)

begin

if (!rst) y2 <= 1; // preset

else y2 <= y1;

end

endmodule
```

After Reset, y1 = 0, y2 = 1;

Second step: the right-hand side of "else" statement in two "always" block is executed simultaneously.  Then assign values to left-hand side simultaneously. So the result is y1 = 1 and y2 = 0.

## Summary:
## Blocking and Non-blocking Assignment

Combinational logic circuit with "Assign" statement uses "=";
Data type: wire.

Combinational logic circuit with "Always" statement uses "=";
Data type: reg. (In some cases, using non-blocking assignment is also correct)

Sequential logic circuit with "Always" statement uses "<=".
(Blocking assignment leads to instability)

# D-Flip-flops

## Flip-flops

The truth table for a positive-edge triggered D flip-flop shows an up arrow to remind you that it is sensitive to its *D* input only on the rising edge of the clock; otherwise it is latched. The truth table for a negative-edge triggered D flip-flop is identical except for the direction of the arrow.

| Inputs | | Outputs | | |
|---|---|---|---|---|
| *D* | CLK | *Q* | $\overline{Q}$ | Comments |
| 1 | ↑ | 1 | 0 | SET |
| 0 | ↑ | 0 | 1 | RESET |

(a) Positive-edge triggered

| Inputs | | Outputs | | |
|---|---|---|---|---|
| *D* | CLK | *Q* | $\overline{Q}$ | Comments |
| 1 | ↓ | 1 | 0 | SET |
| 0 | ↓ | 0 | 1 | RESET |

(b) Negative-edge triggered

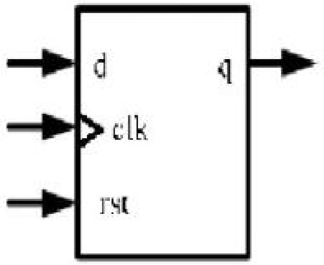
## D Flip-Flop in Verilog

Asynchronous reset:

```
module asynrst_DFF(clk, rst, d, q);
    input clk, rst, d;
    output q;
    reg q;
    always @(posedge clk or posedge rst)
        if(rst)
             q <= 1’b0;
        else
             q <= d;
endmodule
```

Synchronous reset:

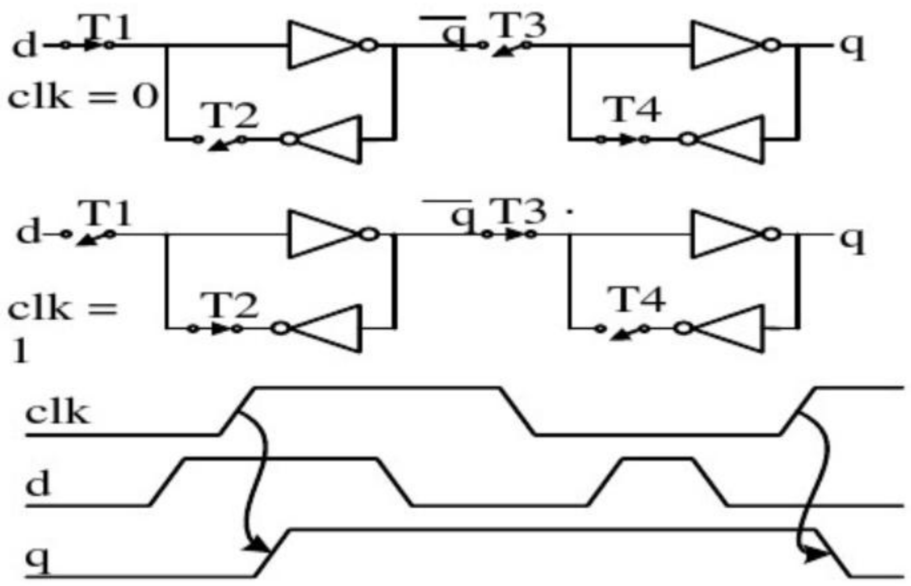
```
module synrst_DFF(clk, rst, d, q);
    input clk, rst, d;
    output q;
    reg q;
    always @(posedge clk)
        if(rst)
             q <= 1’b0;
        else
             q <= d;
endmodule
```



DFF Symbol

D flip-flop is an information storage device with memory function and two stable states. It is the most basic logic unit of a variety of sequential circuits and an important unit circuit in digital logic circuits。

## D Flip-Flop: analysis of working state



Analysis:

1.when clk is low level(0), T1 is open, the signal of input d is transited to $\bar{q}$, when clk changes from low level to high level(posedge), T3 will be open, the value of $\bar{q}$ will be outputed.

2. when clk remains high level, T1 keeps close, whatever changes input d makes, it will not be transitted to the output.