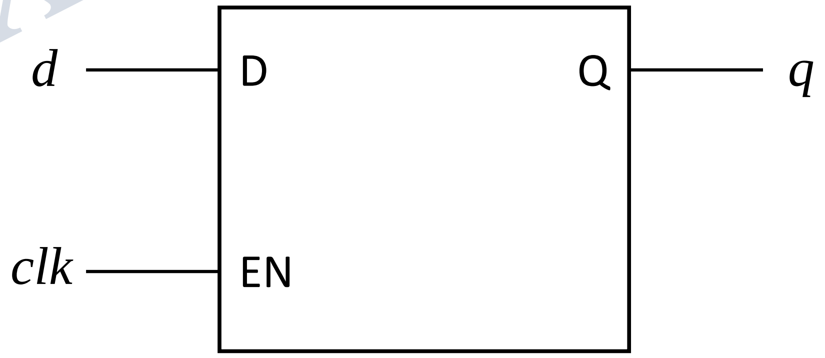
Module #12 : Verilog HDL Component Inference

12.1: Latches:

- A latch is inferred (put into the synthesized circuit) if a variable, or one of its bits, is not assigned in all branch of an if statement.
- A latch is also inferred in a case statement if a variable is assigned to in only some of the branches.
- To improve code readability, use the if statement to synthesize a latch because it is difficult to explicitly specify the latch enable signal using a case statement

Example 12 .1:

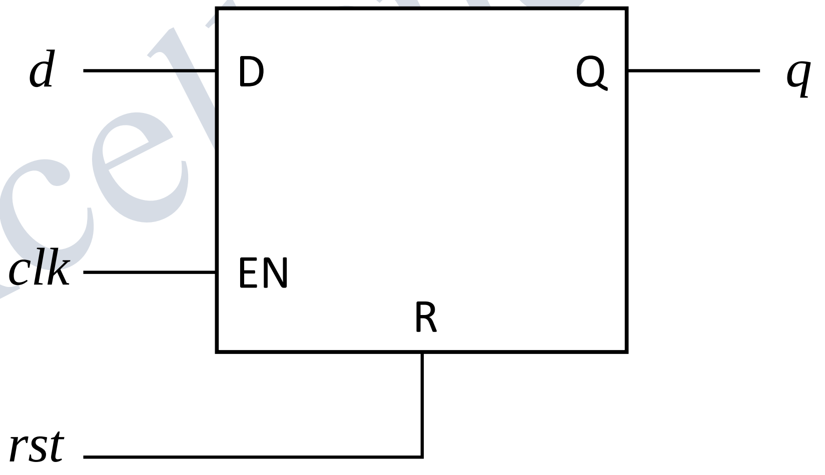
```
always @(clk,d)
   begin
      if (clk)
         q <=d;
   end
```



Module #12 : Verilog HDL Component Inference

Example 12 .2:

```
always @(clk or rst or d)
  begin
    if (rst)
      q <= 0;
    else if (clk)
      q <= d;
   end
```



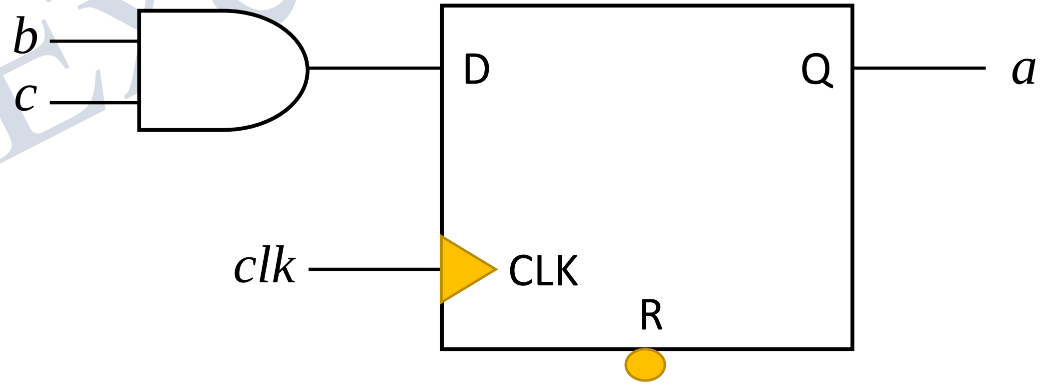
Module #12 : Verilog HDL Component Inference

12.2: Edge-Triggered Registers, Flip-flops, Counters:

- A register (flip-flop) is inferred by using posedge or negedge clause for the clock in the event list of an always block. To add an asynchronous reset, include a second posedge/negedge for the reset and use the if (reset) ... else statement.
- Note that when you use the negedge for the reset (active low reset), the if condition is (!reset).

Example 12 .3:

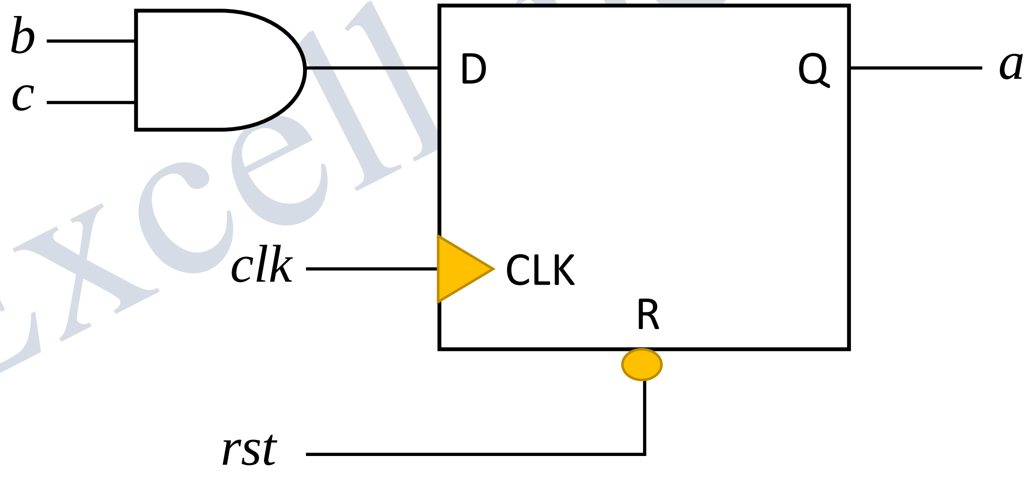
```
always @(posedge clk)
  begin;
    a <= b & c;
  end
```



Module #12 : Verilog HDL Component Inference

Example 12 .4:

```
always @(posedge clk or negedge rst);
  begin;
    if (! rst)
        a < = 0;
    else
        a <= b & c;
end
```



Module #12 : Verilog HDL Component Inference

Example 12 .5:

```
reg [7:0] count;
wire enable;
always @(posedge clk or posedge rst) // Do not include enable.
  begin;
    if (rst)
        count <= 0;
    else if (enable)
       count <= count+1;
  end; // 8 flip-flops will be generated
```

Module #12 : Verilog HDL Component Inference

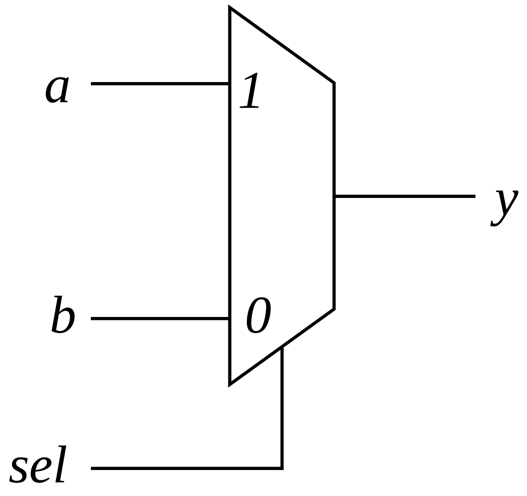
12.3: Multiplexers:

- A multiplexer is inferred by assigning a variable to different variables/values in each branch of an if or case statement.
- You can avoid specifying each and every possible branch by using the else and default branches.
- To improve readability of your code, use the case statement to model large multiplexers

Note that a latch will be inferred if a variable is not assigned to for all the possible branch conditions.

Example 12 .6:

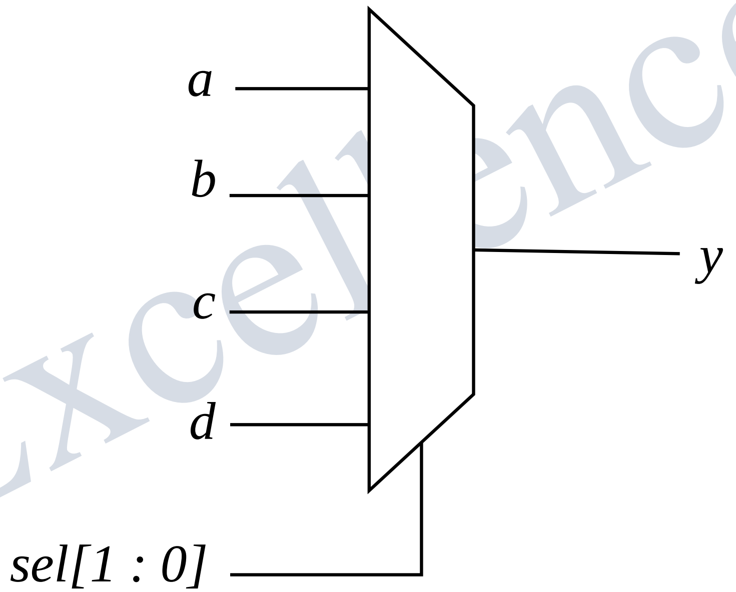
```
if (sel == 1)
    y = a;
else
   y = b;
```



Module #12 : Verilog HDL Component Inference

Example 12 .7:

```
case (sel)
  2'b00: y = a;
  2'b01: y = b;
  2'b10: y = c;
default: y = d;
endcase
```



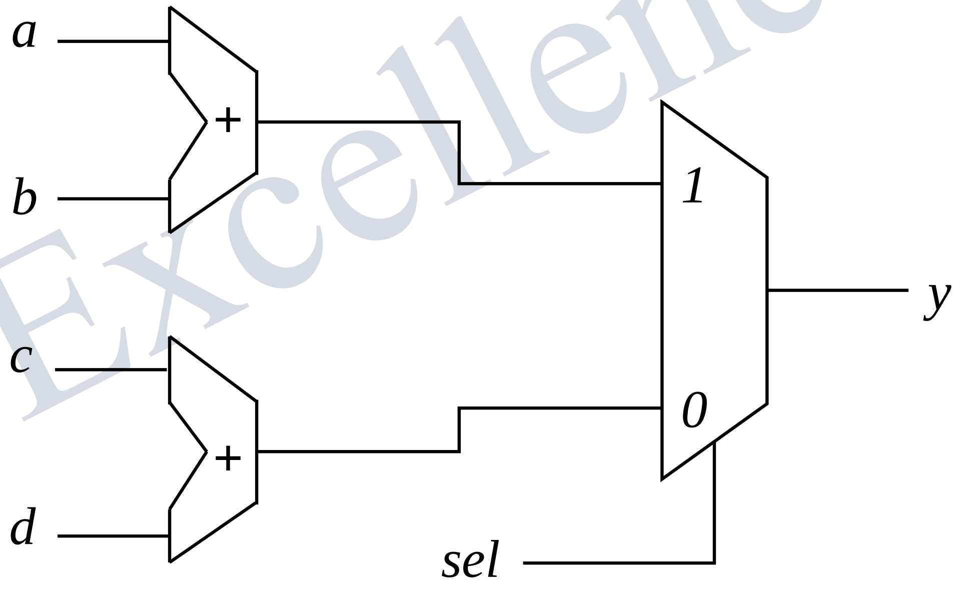
Module #12 : Verilog HDL Component Inference

12.4: Adders/Subtracters:

- The +/- operators infer an adder/subtracter whose width depend on the width of the larger operand

Example 12 .8:

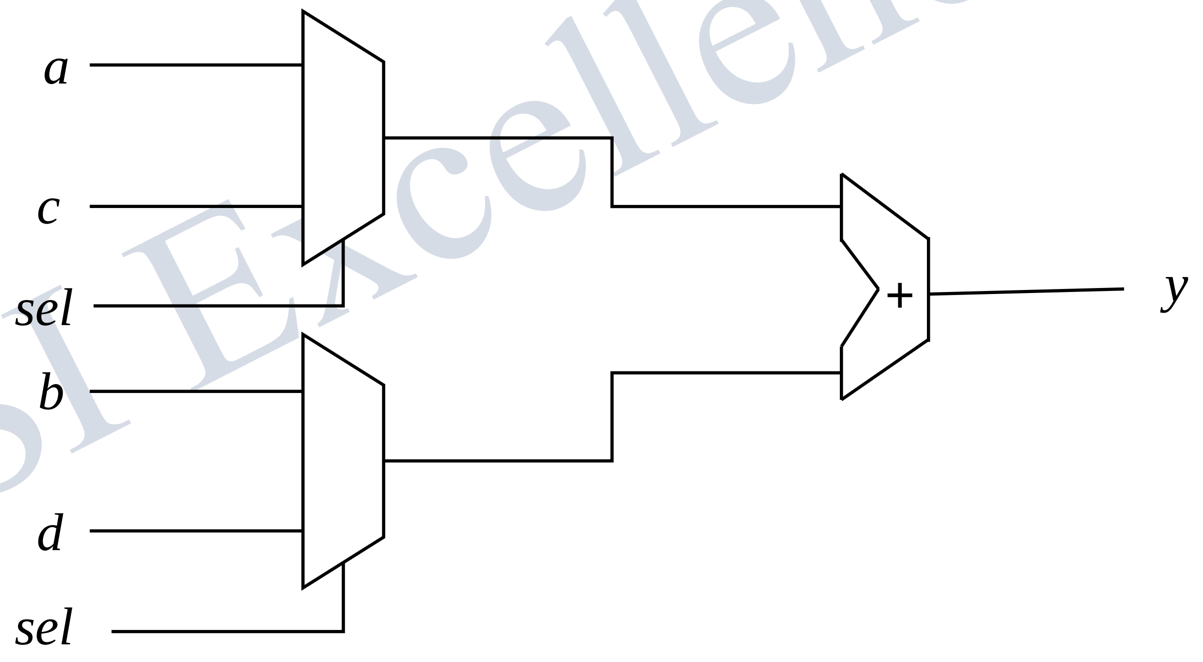
```
if (sel == 1)
  y = a + b;
else
  y = c + d;
```



Module #12 : Verilog HDL Component Inference

Example 12 .9:

```
if (sel == 1)
  y = a + b;
else
  y = c + d;
```



Module #12 : Verilog HDL Component Inference

12.5: Tri-State Buffers:

- A tristate buffer is inferred if a variable is conditionally assigned a value of z using an if, case or conditional operator.

Example 12 .10:

```
if (en == 1)
   y = a;
else
   y = 1'bz;
```