Lecture - 10

✯ Procedural Assignment (contd.)

✯ Recap:
- while loop
- for loop
- repeat loop
- # time_value
- @ (event_expression)
- Laboratory tasks

✱ Example: 2:1 MUX using procedural style

```
module  mux21 (in1, in0, s, f);
   input  in1, in0, s;
   output  reg f;
   always @ (in1 or in2 or s)
        if (s)                  // if (s = 1)
            f = in1;
        else
              f = in0;
endmodule
```

- A 2:1 MUX will be synthesized.

- always @(in1 or in2 or ~~in3~~ s);
      ||(equal)

  always @ (in1, in2, s)
      ||(equal)

  always @ (*)   //always block gets executed when
                      // any of the "input" type variable
                      // changes

~~*~~ * Example: A D type negetive edge triggered flip-flop

```
module dff_negedge (D, clock, Q, Qbar);
    input D, clock;
    output reg Q, Qbar;
    always @(negedge clock)
    begin
         Q= D;
         Qbar= ~D;
    end
endmodule
```

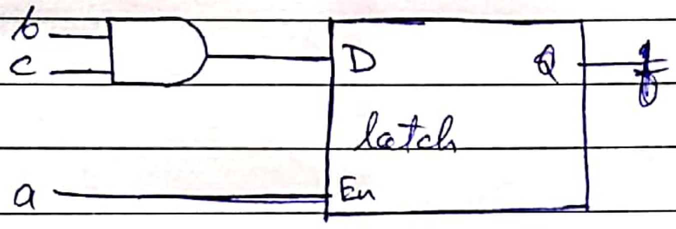
* Example: A 4 bit counter with asynchronous reset

```
module counter (clk, rst, count);
    input clk, rst;
    output reg [3:0] count;
    always @(posedge clk or posedge rst)
    begin
        if (rst)
            count <= 0;
        else
            count <= count + 1;
    end
endmodule
```

Scanned with CamScanner

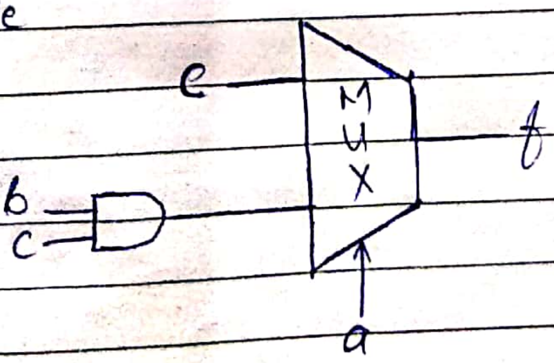
*Example: Determine the hardware generated / synthesized by the following codes

①
```
module xyz (input a,b,c, output reg f);
    always @ (*)
        if (a==1)
            f = b & c;
endmodule
```



Note: New compilers of verilog support the following format
```
module xyz(input a,b,c, output reg f);
```

②
```
module xyz1 (input a,b,c, output reg f);
    always @ (*)
    begin
        f = c;
        if (b==1)
            f = b & c;
    end
endmodule
```



H.W. Write a verilog code for an 8 bit priority encoder.