Lecture - 12

✡ Blocking & Non-Blocking Assignments

✡ Recap: Examples : 2:1 MUX
D f/f negative edge triggered
4 bit counter

✡ Procedural Assignment Recap
- Procedural assignment → initial block
  → always block
- LHS of a procedural assignment can be of type REGISTER (i.e. reg, integer, real, time)
- Two types of procedural assignment statements
  i) Blocking (denoted by "=")
  ii) Non-Blocking (denoted by "<=")

✡ Blocking Assignment
- General syntax:
  variable_name (=) expression ;
  → specifies blocking assignment.
- Blocking statements are executed in the order they are specified in a procedural block.
- Execution of next assignments is BLOCKED till the current assignment gets executed.
- This does not mean that assignments in other always/initial blocks will also be blocked till execution is finished!

* Example:

```
integer a, b, c;
initial
begin
    a = 10;
    b = 20;
    c = 15;
    a = b+c;        // a = 35
    b = a+5;        // b = 40
    c = a-b;        // c = -5
end
```

* Example

```
integer a, b, c;
initial
begin
    a = 10;          } at time = 0
    b = 20;          }
    c = 15;          }
    #5  a = b+c;     // at time = 5
    #10 b = a+5;     // at time = 15
        c = a-b;     // at time = 15
end
```

* Non-Blocking Assignment

- General syntax:

variable_name <= expression;

<= specifies non-blocking assignment

- "<=" allows scheduling of assignments without blocking execution of statements that follow within the always/initial block.

- The assignments to the target gets scheduled for the end of the simulation cycle i.e. at the end of initial/always block.
- This allows concurrent procedural assignment, suitable for sequential logic. (Analogous to signal driver of VHDL)

*Example:

```
integer a, b, c;
initial
  begin
    a = 10;          // at time = 0
    b = 20;          // a = 10, b = 20 & c = 15
    c = 15;
  end
initial              // Both initial blocks
begin                // start at time = 0
  a <= #5 b + c;     // a = 35 at time = 5
  b <= #5 a + 5;     // b = 15 at time = 5
  c <= #5 a - b;     // c = -10 at time = 5
end
```

Q] Write a verilog code to swap values of 2 variables using blocking statement -

Q] Repeat above using non-blocking statement.