Behavioral Modeling

Dr.K.Sivasankaran
Associate Professor,
Department of Micro and Nano Electronics
School of Electronics Engineering

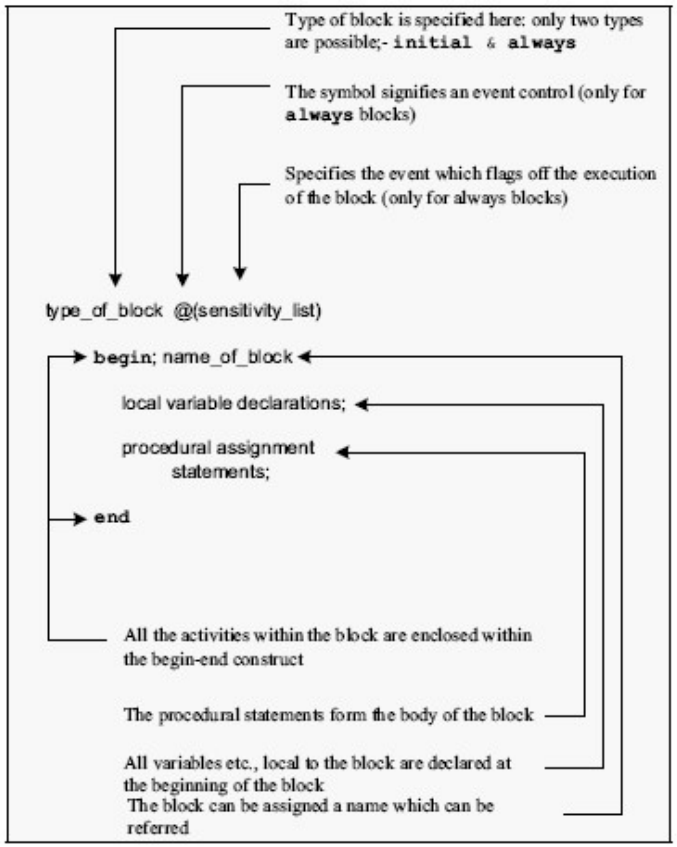
Behavioral Level Modeling

- This is the highest level of abstraction provided by Verilog HDL.
- A module can be implemented in terms of the desired design algorithm without concern for the hardware implementation details.
- Designing at this level is very similar to C programming.

Procedural Assignment

- There are two structured procedures in Verilog:
  - **initial**
  - **always**
- Concurrent execution is observed in between these procedures.
- Sequential / concurrent execution can be realized within these procedures.
- Only registers can be assigned in these procedures.
- The assignments in these procedures are called “procedural assignments”.

Typical Procedural Block



initial block

- Starts execution at ‘0’ simulation time and executes only once during the entire simulation.
- Multiple statements in initial block can be grouped with (**begin** & **end**) or (**fork** & **join**) keywords.
- These blocks are not synthesizable.
- initial blocks cannot be nested.
- Each initial block represent a separate and independent activity.
- initial blocks are used in generating test benches.

initial block

```
initial
 xor_out = in1 ^ in2;
```

```
 initial
 begin
   and_out = a_in & b_in;
 end
```

```
initial
begin
    enable = 1'b0;
    rst    = 1'b0;
    #100   rst = 1'b1;
    #20 enable = 1'b1;
end
```

```
initial
begin
    clk = 1'b0;
    reset   = 1'b0;
 initial
 begin
    #100   reset = 1'b1;
    #20 clk = 1'b1;
    end
    end
```

Multiple Initial Blocks

- A module can have as many **initial blocks as desired.**
- **All of them are** activated at the start of simulation.
- The time delays specified in one **initial** block are exclusive of those in any other block.

Multiple Initial Blocks

```
module nil1;
reg a, b;
initial
begin
a = 1'b0; b = 1'b0;
#2 a = 1'b1; #3 b = 1'b1; #1 a = 1'b0;
end
initial
begin
$monitor ($time, "monitor: a = %b, b = %b", a, b);
end
initial
begin
#100$stop;
end
endmodule
```

always statement

- Starts execution at ‘0’ simulation time and is active all through out the entire simulation.
- Multiple statements inside always block can be grouped with (**begin** & **end**) or (**fork** & **join**) keywords.
- Execution of always blocks is controlled by using the timing control.
- always blocks cannot be nested.

always statement

- An always block without any sensitivity control will create an infinite loop and execute forever.
- Each always block represent a separate and independent activity.
- These blocks can synthesize to different hardware depending on their usage.
- always block with timing control are synthesizable.

always statement - Example

```
always
   xor_out = in1 ^ in2;
```

```
always @(a_in or b_in)
  begin
    and_out = a_in & b_in;
  end
```

```
always @(posedge reset)
  begin
    if (reset == 1'b1)
    q_out =  1'b0;
    else
    q_out =  d_in;
  end
```

```
always
  begin
    cnt = 1'b0;
    reset   = 1'b0;
   always @(posedge clk)
    begin
      #100  cnt = 1'b1;
      #20 enable = 1'b1;
    end
 end
```

always statement - Example

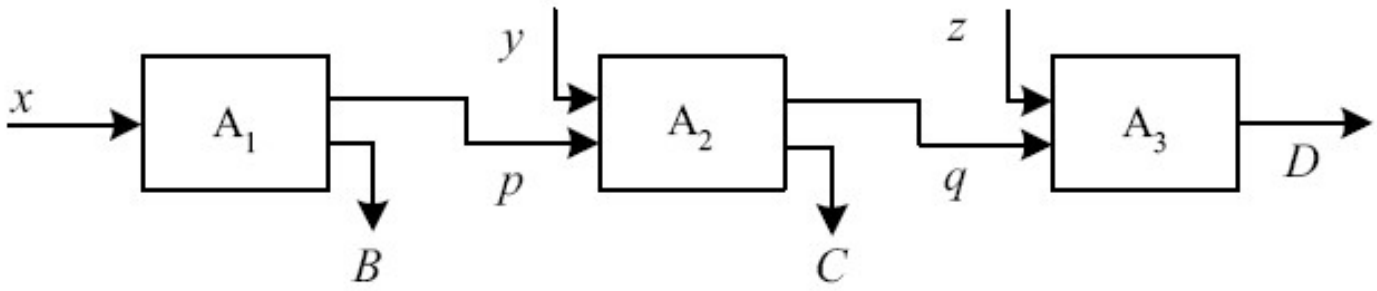
```
module regis (A,B,Y);
output  Y;
input   A,B;
reg  Y;
always @(A,B)
begin
if (A==1 && B==1)
Y=1'b1;
else
Y=1'b0;
end
endmodule
```

```
module dff (clk, rst,din,dout);
input clk, rst, din;
output dout;
reg dout;
always @(posedge clk)
begin
if (rst)
dout = 1'b0;
else
dout=din;
end
endmodule
```

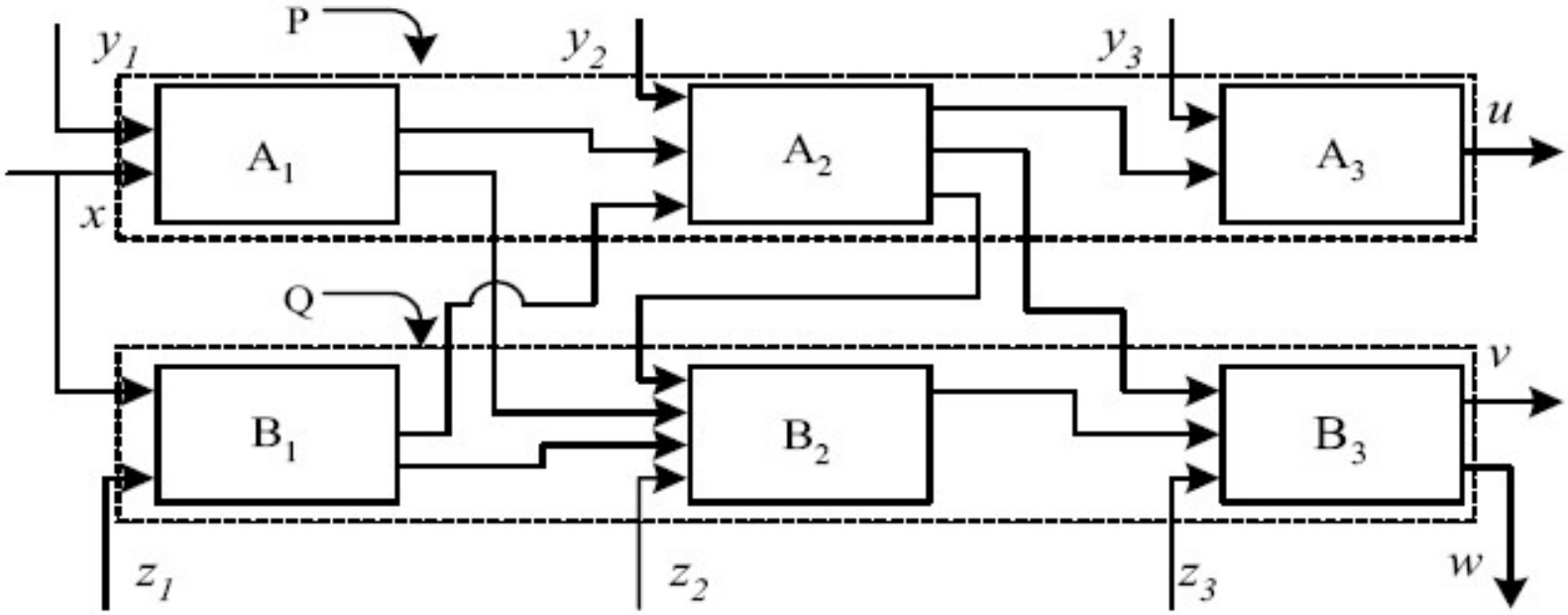
Multiple always statement

- All the activities within an always block are scheduled for sequential execution.
- The activities can be of a combinational nature, a clocked sequential nature, or a combination of these two.
- Basically, any circuit block whose end-to-end operation can be described as a continuous sequence can be described within an **always** block.

A module where execution proceeds through three blocks sequentially



A module where execution proceeds concurrently through two groups of blocks



Test for Understanding

- always block can be nested – True/ False
- initial block will execute only once – True/False
- always block without timing control leads to _____
- Concurrent is observed between initial and always block - True/False
- initial block requires timing control – True / False

Reading Material

- Samir Palnitkar, “VerilogHDL – A guide to digital design and Synthesis” – Chapter-7. Section 7.1
- T.N.Padmanabhan, “Design Through VerilogHDL” – Chapter 7. Section 7.1 to 7.6