Session #5

Loops in Verilog

- If the statements are to be executed more than once then loops should be used.
- Loops generally do not have any significance in digital hardware, hence do not use loops for modeling hardware until and unless required.

forever – continuously executes a statement
forever statement;

repeat – executes a statement specified number of times
repeat (expression) statement;

while – executes a statement until the expression becomes false
while (expression) statement;

for – executes a statement while the expression is true and counts number of executions
for (reg_initiation; expression;
reg_modification) statement;

Loops in Verilog

Examples:

```
forever #10 clk = ! clk;
 // statement clk=!clk is executed continuously

repeat (5) begin
  shreg = shreg << 1;
  shreg[0] = din;
end
 // statements in begin…end block are executed 5 times.

while (acc < 8'b10000000)
  acc = acc + 8'b00011011;
 // statement acc = acc + 8'b00011011; is executed
 // while acc < 8'b11000000

for (i=0; i<32; i=i+1) mem[i] = 0;
 // statement mem[i] = 0; is executed 16 times.
 // i variable must be declared as integer for this
 // example.
```

For all loops, if a conditional expression evaluates to an unknown (x) or a high-impedance value then it is treated as false.

Timing Controls

- Timing controls are required to advance the simulation time.
- There are two methods of timing control:
  - Delay based timing control
  - Event based timing control
  - Level sensitive timing control

Delay based timing control

```
parameter latency = 20;
parameter delta = 3;
reg x, y, z;

initial
begin
        x = 0;
        #5 y = 10;
        #(latency + delta) z = 30;
        #y x = 40;
end
endmodule
```

Equivalent statements:

```
#5;
y = 10;
```
control with number

```
#(latency+delta);
z = 30;
```
control with expression

```
#y;
x = 40;
```
control with identifier

This is called inter-segment delay control.

Delay based timing control

```
reg x, y, z;

initial
begin
        x = 7;
        y = 10;
        z = #10 x + y;
end
endmodule
```

Even if the value of 'x' and 'y' changes between time 0 ns and 10 ns, value assigned to 'z' at time 10 ns is un-affected.

Equivalent statements:

```
reg x, y, z, t;
initial
begin
        x = 7;
        y = 10;
        t = x + y;
        #10 z = t;

end
endmodule
```

This is called intra-segment delay control.

Event based timing control

An event is a change in value of register or a net. Events can be utilized to trigger execution of a statement or a block of statements (e.g. sensitivity list).

Regular Event Control

The symbol ‘@’ is used to specify the event control.

Statements can be executed on:

- Changes in signal value
- Positive edge
- Negative edge

The keyword **posedge** is used for positive edge transition and **negedge** for negative edge transition.

Examples:

```
@(clock) q = d;
@(posedge clock) q = d;
@(negedge clock) q = d;
initial
begin
#10;
@(posedge clock) x = 10;
end
always@(a,b) y = a & b;
q = @(posedge clock) d;
```

Event based timing control

Named Event Control

Verilog provides a capability to declare an event and then trigger and recognize the occurrence of that event.

The event does not hold any data.

A named event is declared using the keyword ***event***.

An event is triggered by the symbol **->**.

The triggering of the event is recognized by the symbol @

Examples:

```
event rx_data;
always@(posedge clock)
begin
        if(last_data_pkt)
        -> rx_data;
end
always@(rx_data)
begin
        data_buf = data_pkt;
end
```

Event based timing control

Event OR Control

Some times a transition on any one of the multiple signals or events can trigger the execution of a statement or a block of statements. This is expressed as an OR of events or signals.

This list of events or signals expressed as an OR is also known as a ***sensitivity list***.

```
module decode_b (
input   IN,
input   [1:0] SEL,
output reg [3:0] OUT
);

  always @ ( IN or SEL )
    begin
      OUT = 4'b0000;
      case (SEL)
        2'b00:   OUT[0] = IN;
        2'b01:   OUT[1] = IN;
        2'b10:   OUT[2] = IN;
        2'b11:   OUT[3] = IN;
      endcase
    end
endmodule
```

Sensitivity list

Level sensitive timing control

- Verilog provides a facility to wait for a certain condition to be true before a statement or a block of statements are executed.
- The keyword ***wait*** is used for level sensitive contructs.

Examples:

```
always
wait( count_enable )
        #20 count = count + 1;
```