Verilog Behavioral Modeling Part-III

Feb-9-2014

ASIC WORLD Directory
Verilog Tutorial
Examples
Questions
Tools
Books
Links
FAQ
Sponsor
Home
Disclaimer
FAQ
Hot Jobs

Looping Statements

Looping statements appear inside procedural blocks only; Verilog has four looping statements like any other programming language.

- forever
- repeat
- while
- for

The forever statement

The forever loop executes continually, the loop never ends. Normally we use forever statements in initial blocks.

syntax : forever < statement >

One should be very careful in using a forever statement: if no timing construct is present in the forever statement, simulation could hang. The code below is one such application, where a timing construct is included inside a forever statement.

Example - Free running clock generator

```
1  module forever_example ();
2
3  reg clk;
4
5  initial begin
6    #1  clk = 0;
7    forever begin
8      #5  clk = ! clk;
9    end
10 end
11
12 initial begin
13   $monitor ("Time = %d  clk = %b",$time, clk);
14   #100  $finish;
15 end
16
17 endmodule
```

You could download file forever_example.v here

The repeat statement

The repeat loop executes < statement > a fixed < number > of times.

syntax : repeat (< number >) < statement >

Example- repeat

```
1  module repeat_example();
2  reg  [3:0] opcode;
3  reg  [15:0] data;
4  reg        temp;
5
6  always @ (opcode or data)
7  begin
8    if (opcode == 10) begin
9      // Perform rotate
10     repeat (8) begin
11       #1  temp = data[15];
12       data = data << 1;
13       data[0] = temp;
14     end
15   end
16 end
17 // Simple test code
18 initial  begin
19     $display (" TEMP  DATA");
20     $monitor (" %b    %b ",temp, data);
21     #1  data = 18'hF0;
22     #1  opcode = 10;
23     #10  opcode = 0;
24     #1  $finish;
25 end
26
27 endmodule
```

You could download file repeat_example.v here

The while loop statement

The while loop executes as long as an < expression > evaluates as true. This is the same as in any other programming language.

syntax : while (< expression >) < statement >

Example- while

```
1  module while_example();
2
3  reg [5:0] loc;
```

```
reg [7:0] data;

always @ (data or loc)
begin
  loc = 0;
  // If Data is 0, then loc is 32 (invalid value)
  if (data == 0) begin
    loc = 32;
  end else begin
    while (data[0] == 0) begin
      loc = loc + 1;
      data = data >> 1;
    end
  end
  $display ("DATA = %b  LOCATION = %d",data,loc);
end

initial begin
  #1  data = 8'b11;
  #1  data = 8'b100;
  #1  data = 8'b1000;
  #1  data = 8'b1000_0000;
  #1  data = 8'b0;
  #1  $finish;
end

endmodule
```

You could download file while_example.v here

The for loop statement

The for loop is the same as the for loop used in any other programming language.

- Executes an < initial assignment > once at the start of the loop.
- Executes the loop as long as an < expression > evaluates as true.
- Executes a < step assignment > at the end of each pass through the loop.

syntax : for (< initial assignment >; < expression >, < step assignment >) < statement >
Note : verilog does not have ++ operator as in the case of C language.

Example - For

```
module for_example();

integer i;
reg [7:0] ram [0:255];

initial begin
```

```
  for (i = 0; i < 256; i = i + 1) begin
    #1  $display(" Address = %g  Data = %h",i,ram[i]);
    ram[i] <= 0; // Initialize the RAM with 0
    #1  $display(" Address = %g  Data = %h",i,ram[i]);
  end
  #1  $finish;
end

endmodule
```

You could download file for_example.v here