Module #07 : Verilog HDL Behavioral Modeling Part#2

7.7: for Loops:

- Similar to for loops in C/C++, they are used to repeatedly execute a statement or block of statements.
- If the loop contains only one statement, the **begin ... end** statements may be omitted.

VLSI Excellence

Syntax:

```
for (count = value1; count </<=/>/>= value2; count = count +/- step)
begin
... statements ...
end
```

Example 7 .7:

```
for (j = 0; j <= 7; j = j + 1)
  begin
    c[j] = a[j] & b[j];
    d[j] = a[j] | b[j];
  end
```

Module #07 : Verilog HDL Behavioral Modeling Part#2

7.8: while Loops:

- The while loop repeatedly executes a statement or block of statements until the expression in the while statement evaluates to false.
- To avoid combinational feedback, a while loop must be broken with an @(posedge/negedge clock) statement.
- For simulation a delay inside the loop will suffice.
- If the loop contains only one statement, the begin ... end statements may be omitted.

Note: *While loops are not recommended for synthesizable RTL Code because when we synthesize the RTL code and turn into gates or registers and the synthesizer need to know exactly how many times the loop will execute.*

Syntax:

```
while (expression)
begin
... statements ...
end
```

Example 7 .8:

```
while (!overflow) begin
@(posedge clk);
a = a + 1;
end
```

VLSI Excellence

Module #07 : Verilog HDL Behavioral Modeling Part#2

7.9: forever Loops:

- The forever statement executes an infinite loop of a statement or block of statements.
- To avoid combinational feedback, a forever loop must be broken with an @(posedge/negedge clock) statement.
- For simulation a delay inside the loop will suffice.
- If the loop contains only one statement, the begin ... end statements may be omitted.

Syntax:

```
forever
begin
... statements ...
end
```

Example 7 .9:

```
forever begin
@(posedge clk); // or use a= #9 a+1;
a = a + 1;
end
```

VLSI Excellence

Module #07 : Verilog HDL Behavioral Modeling Part#2

7.10: repeat (Not synthesizable):

- The repeat statement executes a statement or block of statements a fixed number of times.

Syntax:

repeat (number_of_times)
begin
... statements ...
end

Example 7 .10:

```
repeat (2) begin    // after 50, a = 00,
#50 a = 2'b00;      // after 100, a = 01,
#50 a = 2'b01;      // after 150, a = 00,
end                 // after 200, a = 01
```

VLSI Excellence

Module #07 : Verilog HDL Behavioral Modeling Part#2

7.11: disable:

- Execution of a disable statement terminates a block and passes control to the next statement after the block.
- It is like the C break statement except it can terminate any loop, not just the one in which it appears.
- Disable statements can only be used with named blocks.

Syntax:

disable block_name;

Example 7 .11:

```
begin: accumulate
  forever
    begin
      @(posedge clk);
        a = a + 1;
        if (a == 2'b0111) disable accumulate;
    end
end
```

VLSI Excellence

Module #07 : Verilog HDL Behavioral Modeling Part#2

7.12: . if ... else if ... else:

- The **if ... else if ...** else statements execute a statement or block of statements depending on the result of the expression following the if.
- If the conditional expressions in all the if's evaluate to false, then the statements in the else block, if present, are executed.
- There can be as many else if statements as required, but only one if block and one else block.
- If there is one statement in a block, then the begin .. end statements may be omitted.
- Both the else if and else statements are optional. However if all possibilities are not specifically covered, synthesis will generated extra latches.

VLSI Excellence

Module #07 : Verilog HDL Behavioral Modeling Part#2

Syntax:

```
if (expression)
begin
... statements ...
end
else if (expression)
begin
... statements ...
end
... more else if blocks ...
else
begin
... statements ...
end
```

Example 7 .12:

```
if (alu_func == 2'b00)
   aluout = a + b;
else if (alu_func == 2'b01)
   aluout = a - b;
else if (alu_func == 2'b10)
   aluout = a & b;
else // alu_func == 2'b11
   aluout = a | b;
if (a == b)      /* This if with no else will generate
   begin            a latch for x and ot. This is so they
      x = 1;        will hold there old value if (a != b).*/
     ot = 4'b1111;
end
```

VLSI Excellence

Module #07 : Verilog HDL Behavioral Modeling Part#2

7.13: case:

- The case statement allows a multipath branch based on comparing the expression with a list of case choices.
- Statements in the default block executes when none of the case choice comparisons are true.
- With no default, if no comparisons are true, synthesizers will **generate unwanted latches.**
- Good practice says to make a habit of putting in a default whether you need it or not.
- If the defaults are don’t cares, define them as ‘x’ and the logic minimizer will treat them as don’t cares and save area.
- Case choices may be a simple constant, expression, or a comma-separated list of same.

VLSI Excellence

Module #07 : Verilog HDL Behavioral Modeling Part#2

Syntax:

```
case (expression)
case_choice1:
begin
... statements ...
end
case_choice2:
begin
... statements ...
end
... more case choices blocks ...
default:
begin
... statements ...
end
endcase
```

Example 7 .12:

```
case (alu_ctr)
2'b00: aluout = a + b;
2'b01: aluout = a - b;
2'b10: aluout = a & b;
default: aluout = 1'bx; /*Treated as don't cares for
endcase                    minimum logic generation.*/
```

Example 7 .13:

```
case ({w, y})
2'b00: aluout = a + b; //case if x, y is 2'b00.
2'b01: aluout = a - b;
2'b10: aluout = a & b;
2'b11; aluout = a|b;
default: $display("Invalid w,y = %b %b ", w, y);
endcase     //Display an error if w,y contain 'x's.
```

VLSI Excellence

Module #07 : Verilog HDL Behavioral Modeling Part#2

7.13: casex:

- In casex(a), when the case_choice constants contains z, x or ?, they match any value in “a”.
- With case(a), a = x (unknown) matches only a case_choice of x, not 1, 0 or z.
- Also a=z (3-state) matches only z. In short, case uses x to detect a signal value of a=‘x’(unknown).
- Casex uses x as a wild card which can match anything.

Syntax:

same as for case statement

Example 7 .14:

casex (a)
2’b1x: msb = 1; // msb = 1 if a = 10 or a = 11
// If this were case(a) then only a=1x would match.
2’bz1 : msb = 0; // msb = 0 if a = 01 or a = 11
default: msb = 0;
endcase

VLSI Excellence

Module #07 : Verilog HDL Behavioral Modeling Part#2

7.13: casez:

- Casez is the same as casex except only ? and z (not x) are used in the case choice constants as don't cares.
- Casez is favored over casex since in simulation, an inadvertent x signal, will not be matched by a 0 or 1 in the case choice.

Syntax:

same as for case statement

Example 7 .15:

```
casez (d)
3'b1??: b = 2'b11;      // b = 11 if d = 100 or greater
3'b01?: b = 2'b10;      // b = 10 if d = 010 or 011
default: b = 2'b00;
endcase
```

VLSI Excellence