שפת תכנון חומרה
ורילוג - Verilog

Verilog – Parameterized Module

Dr. Avihai Aharon

Parameterized Modules

```
module register (q, d, clk, rst_n);
  parameter SIZE=8;
  output [SIZE-1:0] q;
  input  [SIZE-1:0] d;
  input             clk, rst_n;
  reg    [SIZE-1:0] q;

  always @(posedge clk or negedge rst_n)
    if (!rst_n) q <= 0;
    else        q <= d;
endmodule
```

- Parameterized register model - Verilog-1995 style

Parameterized Modules

```
module register2001 #(parameter SIZE=8)
   (output reg [SIZE-1:0] q,
    input      [SIZE-1:0] d,
    input                 clk, rst_n);

   always @(posedge clk, negedge rst_n)
     if (!rst_n) q <= 0;
     else        q <= d;
endmodule
```

- Parameterized register model - Verilog-2001 style

Parameterized Modules

```
module two_regs1 (q, d, clk, rst_n);
  output [15:0] q;
  input  [15:0] d;
  input                clk, rst_n;
  wire   [15:0] dx;

  register #(16) r1 (.q(q), .d(dx),
          .clk(clk), .rst_n(rst_n));

  register #(16) r2(.q(dx), .d(d),
          .clk(clk), .rst_n(rst_n));
endmodule
```

Instantiation using parameter redefinition

Parameterized Modules

```
module myreg (q, d, clk, rst_n);
  parameter Trst = 1,
            Tckq = 1,
            SIZE = 4;
  output [SIZE-1:0] q;
  input  [SIZE-1:0] d;
  input             clk, rst_n;
  reg    [SIZE-1:0] q;

  always @(posedge clk or negedge rst_n)
    if (!rst_n) q <= #Trst 0;
    else        q <= #Tckq d;
endmodule
```

Module with four parameters

Parameterized Modules

```
module bad_wrapper (q, d, clk, rst_n);
  output [7:0] q;
  input  [7:0] d;
  input        clk, rst_n;

  // illegal parameter passing example
  myreg #(,,8) r1 (.q(q), .d(d),
                   .clk(clk), .rst_n(rst_n));
endmodule
```

Parameter redefinition with #(,,8) syntax error

Parameterized Modules

```
module good_wrapper (q, d, clk, rst_n);
  output [7:0] q;
  input  [7:0] d;
  input        clk, rst_n;

  // the first two parameters must be
  // explicitly passed even though the
  // values did not change
  myreg #(1,1,8) r1 (.q(q), .d(d),
                     .clk(clk), .rst_n(rst_n));
endmodule
```

Parameter redefinition with correct #(1,1,8) syntax

Parameterized Modules

```
module demuxreg (q, d, ce, clk, rst_n);
  output [15:0] q;
  input  [ 7:0] d;
  input         ce, clk, rst_n;
  wire   [15:0] q;
  wire   [ 7:0] n1;

  not           u0 (ce_n, ce);
  regblk #(.SIZE( 8)) u1
          (.q(n1), .d (d), .ce(ce),
           .clk(clk), .rst_n(rst_n));
  regblk #(.SIZE(16)) u2
          (.q (q), .d({d,n1}), .ce(ce_n),
           .clk(clk), .rst_n(rst_n));
endmodule

module regblk (q, d, ce, clk, rst_n);
  parameter SIZE = 4;
  output [SIZE-1:0] q;
  input  [SIZE-1:0] d;
  input             ce, clk, rst_n;
  reg    [SIZE-1:0] q;

  always @(posedge clk or negedge rst_n)
    if      (!rst_n) q <= 0;
    else if (ce)     q <= d;
endmodule
```

Instantiation using named parameter passing

Parameterized Modules – clock divider by 2

```
module frequency_divider_by2 ( clk ,rst,out_clk );
        output reg out_clk;
        input clk ;
input rst;
always @(posedge clk)
begin
if (~rst)
    out_clk <= 1'b0;
else
    out_clk <= ~out_clk;
end
endmodule
```

Parameterized Modules – clock divider

```
module Clock_divider (clock_in,clock_out );
    input clock_in; // input clock on FPGA
    output clock_out; // output clock after dividing the input clock by div
    reg[27:0] counter=28'd0;
    parameter DIV = 28'd2;
// The frequency of the output clk_out
//  = The frequency of the input clk_in divided by DIV
// For example: Fclk_in = 50Mhz, if you want to get 1Hz signal to blink LEDs
// You will modify the DIV parameter value to 28'd50.000.000
// Then the frequency of the output clk_out = 50Mhz/50.000.000 = 1Hz
    always @(posedge clock_in)
    begin
        counter <= counter + 28'd1;
        if(counter>=(DIV-1))
            counter <= 28'd0;
    end
    assign clock_out = (counter<DIV/2)?1'b0:1'b1;
endmodule
```

Parameterized Modules – clock divider - TB

```
module tb_clock_divider;
    // Inputs
    reg clock_in;
    // Outputs
    wire clock_out;
    // Instantiate the Unit Under Test (UUT)
    // Test the clock divider in Verilog
    Clock_divider #(.DIV(4)) dut (
      .clock_in(clock_in),
      .clock_out(clock_out) );

    initial begin
        // Initialize Inputs
        clock_in = 0;
        // create input clock 50MHz
         forever #10 clock_in = ~clock_in;
     end
endmodule
```