


EE461 Verilog-HDL

Week 9 Parameterized Modules

Alex Yang, Engineering School, NPU

Week 9 Outlines

- Constant Declaration in Module
  - Inside Declaration Style
    - e.g. parameter const1 = 10;
    - e.g. localparam const2 = 20;
  - Port List Style
    - e.g. module #(parameter A=1,parameter B=2)(in1,out1);
      ... ... endmodule
- Overriding Parameter
  - Module Instance Overriding
  - "defparam" Overriding
- Difference among parameter, localparam and specparam

Constant Declaration

- Inside Declaration Style
  - syntax: parameter A = 10;
  - must be outside of initial, always & assign blocks
  - must be before any program statements and right after port & variables declaration, and can't be used w/o declaration
  - local scope: valid only within module
  - can be overriden

```
module register (q, d, clk, rst);
    parameter SIZE=8;
    output [SIZE-1:0] q;
    input [SIZE-1:0] d;
    input clk, rst;
    reg [SIZE-1:0] q;
   always @(posedge clk or negedge rst)
      if (!rst_n) q <= 0;
      else q <= d;
endmodule
```

```
parameter SIZE = 7;
parameter WIDTHBUSA = 24, WIDTHBUSB = 8;
parameter signed [3:0] MUXSEL = 4'b0000;
-----------------------------------------------------------------
parameter A=1, B=2, C=3, D = 4;
```

Note: size of constant variable could be declared

Constant Declaration

- Port List Style: Verilog-2001 version

```
module modName #(parameter
Declarations)
(port_declarations);

    other_declarations;
    statements;
    ... ...
endmodule
```

```
module register2001 #(parameter SIZE=8)
(
            output reg [SIZE-1:0] q,
            input [SIZE-1:0] d,
            input clk, rst
); // Declare data type in the port list
      always @(posedge clk, negedge rst)
            if (!rst) q <= 0;
            else q <= d;
endmodule
```

Overriding Parameter

- ## Module Instance Overriding

```
module register (q, d, clk, rst);
     parameter SIZE=8;
    output [SIZE-1:0] q;
    input [SIZE-1:0] d;
    input clk, rst;
    reg [SIZE-1:0] q;
   always @(posedge clk or negedge rst)
      if (!rst) q <= 0;
      else q <= d;
endmodule
```

```
`include "register.v"
module two_regs1 (q, d, clk, rst);
   output [15:0] q;
   input [15:0] d;
   input clk, rst;
   wire [15:0] dx;
  // Redefine parameter by the order
  register #(16) r1 ( .q(q), .d(dx), .clk(clk), .rst(rst) );
  register #(16) r2 ( .q(dx), .d(d), .clk(clk), .rst(rst) );
endmodule
```

```
module myreg (q, d, clk, rst);
    parameter Trst = 1,
                  Tckq = 1,
                  SIZE = 4,
                  VERSION = "1.1";
    output [SIZE-1:0] q;
    input [SIZE-1:0] d;
    input clk, rst;
    reg [SIZE-1:0] q;
   always @(posedge clk or negedge rst)
      if (!rst) q <= #Trst 0;
      else q <= #Tckq d;
endmodule
```

Bad code:

```
`include "myreg.v"
module bad(q, d, clk, rst);
   output [7:0] q;
   input [7:0] d;
   input clk, rst;
   // illegal parameter passing only for 3rd value
   myreg #(,,8) r1 (.q(q), .d(d), .clk(clk), .rst_n(rst_n));
endmodule
```

Good code:

```
`include "myreg.v"
module good (q, d, clk, rst);
    output [7:0] q;
    input [7:0] d;
    input clk, rst;
 // must specify all values in the changed values even thought they don't change
    myreg #(1,1,8) r1 (.q(q), .d(d), .clk(clk), .rst(rst));
endmodule
```

Overriding Parameter

- Module Instance Overriding
  - Verilog-2001 Standard

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

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

endmodule
```

```
`include "regblk.v"
module demuxreg (q, d, ce, clk, rst);
    output [15:0] q;
    input [ 7:0] d;
    input ce, clk, rst;
    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(rst));
regblk #(.SIZE(16))u2 (.q(q), .d({d,n1}),
            .ce(ce_n), .clk(clk), .rst(rst));
endmodule
```

Overriding Parameter

- "defparam" Overriding
  - syntax: defparam A = 20;
  - must be outside of initial, always & assign blocks
  - In the case of multiple defparams for a single parameter, the parameter takes the value of the last defparam statement encountered in the submodule
  - local scope: valid only within module

```
module secretNum;
    parameter mySecret = 10;
initial begin
  $display("My secret number in module is %0d", mySecret);
end
endmodule

module testDefparam();
    defparam U0.mySecret = 11;          // Redefine parameter vaules for submodule
    defparam U1.mySecret = 22;

   secretNum U0();
   secretNum U1();
endmodule
```

Overriding Parameter

- "defparam" Overriding Abuse
  - Hierarchical defparams

```
`include "register2.v"
module defparamTB;
    parameter SIZE=8;
    wire [SIZE-1:0] q;
    reg [SIZE-1:0] d;
    reg clk, rst;

   register2 #(SIZE) r1  (.q(q), .d(d), .clk(clk), .rst(rst));
   // ...
endmodule
```

```
`include "dff.v"
module register2 (q, d, clk, rst);
    parameter WIDTH=8;
    output [WIDTH-1:0] q;
    input [WIDTH-1:0] d;
    input clk, rst;

    dff #(WIDTH) d1 (.q(q), .d(d), .clk(clk), .rst(rst));
endmodule
```

Note: WIDTH = SIZE = 8

```
module dff (q, d, clk, rst);
    parameter N=1;
    output [N-1:0] q;
    input [N-1:0] d;
    input clk, rst;
    reg [N-1:0] q;
    // dangerous, hierarchical defparam
    defparam defparamTB.SIZE = 1;

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

Note: hierarchical redefine is not good

Overriding Parameter

- "defparam" Overriding Abuse
  - multiple defparams in the same file

```
module register (q, d, clk, rst);
    parameter SIZE=8;
   output [SIZE-1:0] q;
   input [SIZE-1:0] d;
   input clk, rst;
   reg [SIZE-1:0] q;
   always @(posedge clk or negedge rst)
      if (!rst_n) q <= 0;
      else q <= d;
endmodule
```

```
`include "register.v"
module twoReg (q, d, clk, rst);
      parameter SIZE = 16;
     output [SIZE-1:0] q;
     input [SIZE-1:0] d;
     input clk, rst;
     wire [SIZE-1:0] dx;

   defparam r1.SIZE=16;
   register r1 (.q(q), .d(dx), .clk(clk), .rst_n(rst_n));

   defparam r2.SIZE=16;
   register r2 (.q(dx), .d(d), .clk(clk), .rst_n(rst_n));
   defparam r2.SIZE=4; // Design error!
endmodule
```

Note: port-size mismatch for r2 unit

Diff. b/w Localparam & Parameter

- localparam is valid only within module
- localparam can't be redefined in module overriding

```
module testLocalparam;
    localparam A = 10;
    //parameter A = 10;
    initial begin
        $display("Localparam in module is %0d", A);
    end

endmodule

module top;
    testLocalparam #(11) U0();
    testLocalparam #(22) U1();
endmodule
```

Running results for localparam:

```
[yangjh@npu29 Test_localparam]$ ./simv
Chronologic VCS simulator copyright 1991-2011
Contains Synopsys proprietary information.
Compiler version E-2011.03; Runtime version E-2011.03;  Mar  1 01:37 2014
Localparam in module is 10
Localparam in module is 10
            V C S   S i m u l a t i o n   R e p o r t
Time: 0
CPU Time:      0.290 seconds;       Data structure size:   0.0Mb
Sat Mar  1 01:37:44 2014
```

Running results for parameter:

```
[yangjh@npu29 Test_localparam]$ ./simv
Chronologic VCS simulator copyright 1991-2011
Contains Synopsys proprietary information.
Compiler version E-2011.03; Runtime version E-2011.03;  Mar  1 01:40 2014
Localparam in module is 11
Localparam in module is 22
            V C S   S i m u l a t i o n   R e p o r t
Time: 0
CPU Time:      0.300 seconds;       Data structure size:   0.0Mb
Sat Mar  1 01:40:53 2014
```

Diff. b/w Localparam & Parameter

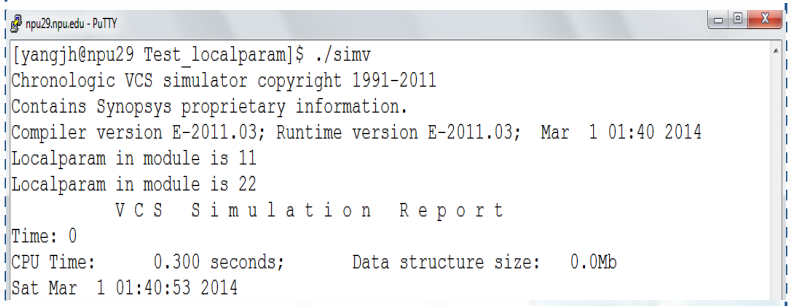
- localparam can't be changed by defparam

```
module testDef;
        localparam A = 10;
        //parameter A = 10;
        initial begin
                $display("Localparam in module is %0d", A);
        end
endmodule
module top;
        defparam U0.A=11;
        defparam U1.A=12;

        testDef U0();
        testDef U1();
endmodule
```

Note: compilation errors

Running results for parameter A= 10;



Diff. b/w specparam & parameter

- specparam(Timing Control)
  - must be in specify block
  - Invalid in the outside of specify block
  - can't be redefined by defparam
  - could be overridden by SDF(standard delay file)
- parameter
  - must be in the outside of specify block
  - Valid only in the outside of specify block
  - could be redefined by defparam