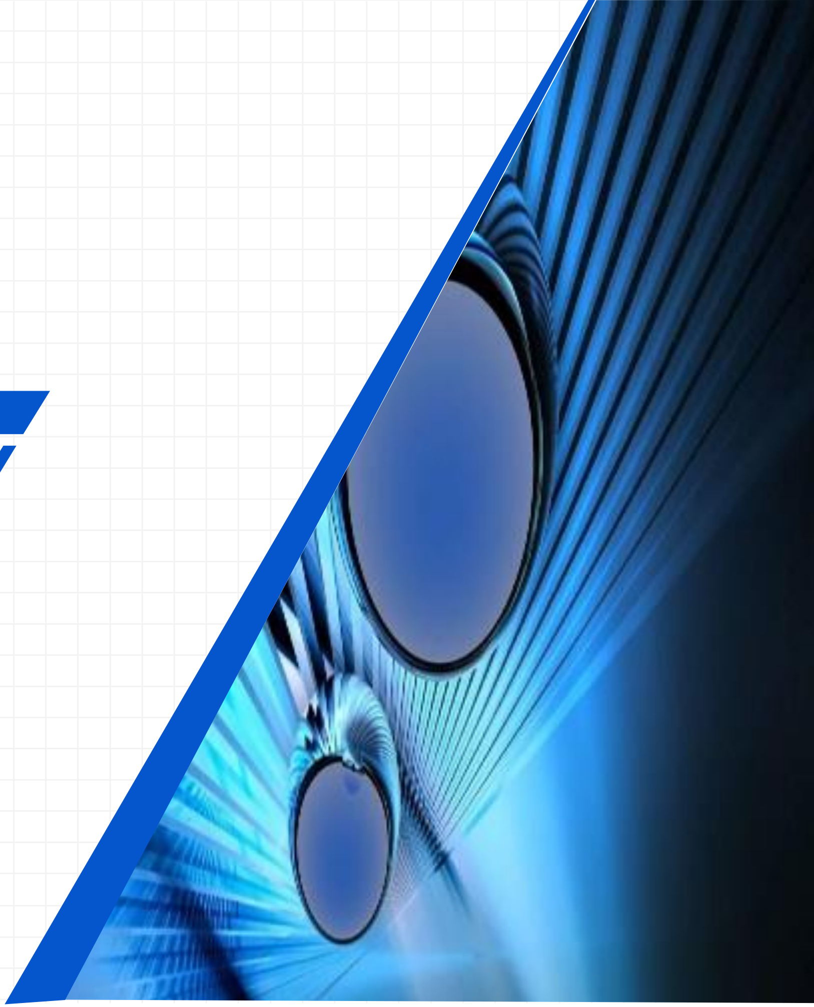
LINTING (RTL Quality Check)

L&T Technology Services

Parikshit Dhodapkar & Krishnakant Patil

Project Lead



Agenda

Purpose

- Front-end methodology during which rules are applied to developed RTL in order to identify potential errors / bugs which would otherwise be captured in later or upcoming design stages (synthesis, etc.)

- It points out where the code is most likely to have bugs

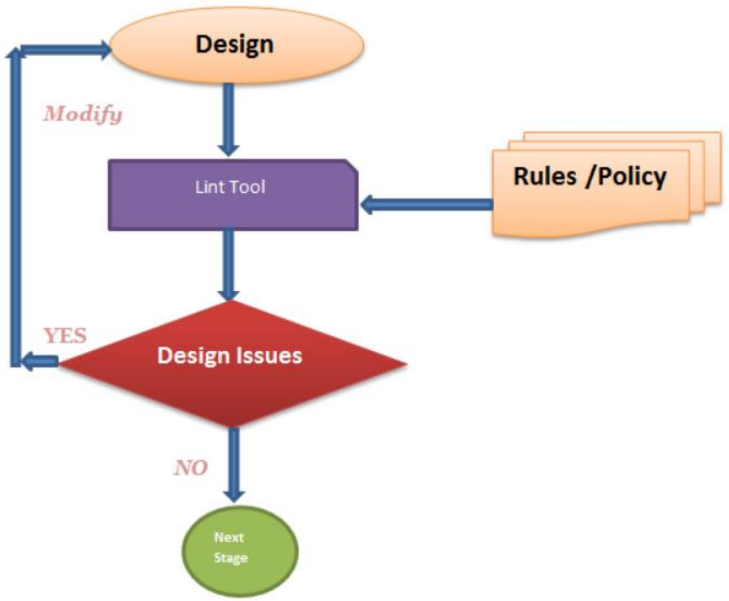
Working

- A set of rules (conditions) that needs to be checked on the design (RTL) is defined in the Lint tool (Ex. SpyGlass)

- User can enable and disable the required rules as per his requirement

- On running these rules on the design, violation is reported if design (RTL code) does not conform to a rule

Working (cont…)



Steps

- Define linting rules; modify, change severity level as required
- Run linting tool with the defined rule
- Analyze generated log and report
- Make necessary fixes if required
- Rerun the tool

Popular Lint tools

1. SpyGlass (Atrenta now Synopsys)
2. Incisive HAL (Cadence)
3. RTL Compiler (RC) from Cadence also has a switch for Lint
4. Product companies may have their own proprietary lint tool

How does it help?

Running lint can help resolve issues in the following:

- Simulation – synthesis mismatches

  -
    ```
    always @ (a) begin
        b <= a && c
    end
    ```

    This will cause a simulation and synthesis mismatch. Lint tool will point this out

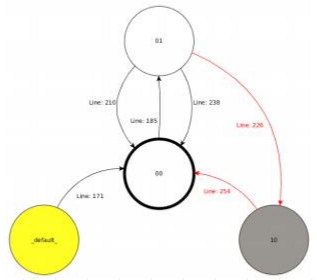
    Ex.:INC_SENS_LIST : If a signal is referenced and not used in the sensitivity list of the block, it reports the failure.

- Coding guidelines

  -
    ```
    always @ (negedge rstn or posedge clk )
        begin
        b = a;
        c = b;
    end
    ```

    Usage of blocking assignments inside an always block will be pointed out by the lint tool

- FSM state reachability
  - It reports, if a state in a FSM that once entered never reaches to another state via next state assignment
  - Ex.: TERMINAL_STATE: RTL code which results in the alongside FSM:



How does it help?

Running lint can help resolve issues in the following:

- Network and connectivity checks for clocks, resets, and tri-state driven signals
  - Unused and unconnected ports, wires are reported by the tool

- Possible synthesis issues. (Ex. latches or combo loops)
  - Inferring latches will throw Lint error
  - Also combo-loops:

  It reports if there is a combinational loop in the design

  Ex.: if (rst) cntr <= 0; else if (wobble) cntr <= cntr + 1;

  Here, a combo-loop would be reported. However, it is most likely to be intended design (a counter)

Examples of Lint Issues

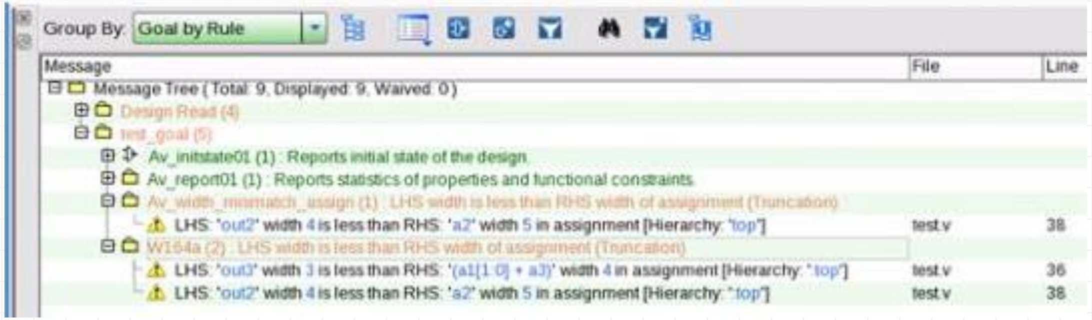
More lint examples:

- Av_width_mismatch_assign (LHS width is less than RHS width of assignment (Truncation))
  - module top(input clk, rst,
    input [4:0]a1,
    input [4:0]a2,
    input [2:0]a3,
    output reg [3:0] out1,
    output reg [3:0] out2,
    output reg [2:0] out3
    );

    .

    .

  out3 <= a1[1:0] + a3;
  if(a1 > out1)
  out2 <= a2;



```
Message                                                                                         File     Line
⊟ Message Tree ( Total: 9, Displayed: 9, Waived: 0 )
  ⊞ Design Read (4)
  ⊟ test_goal (5)
    ⊞ Av_initstate01 (1) : Reports initial state of the design
    ⊞ Av_report01 (1) : Reports statistics of properties and functional constraints
    ⊟ Av_width_mismatch_assign (1) : LHS width is less than RHS width of assignment (Truncation)
        ⚠ LHS: 'out2' width 4 is less than RHS: 'a2' width 5 in assignment [Hierarchy: 'top']        test.v   38
    ⊟ W164a (2) : LHS width is less than RHS width of assignment (Truncation)
        ⚠ LHS: 'out3' width 3 is less than RHS: '(a1[1:0] + a3)' width 4 in assignment [Hierarchy: ':top']   test.v   36
        ⚠ LHS: 'out2' width 4 is less than RHS: 'a2' width 5 in assignment [Hierarchy: ':top']       test.v   38
```

Examples of Lint Issues (cont..)

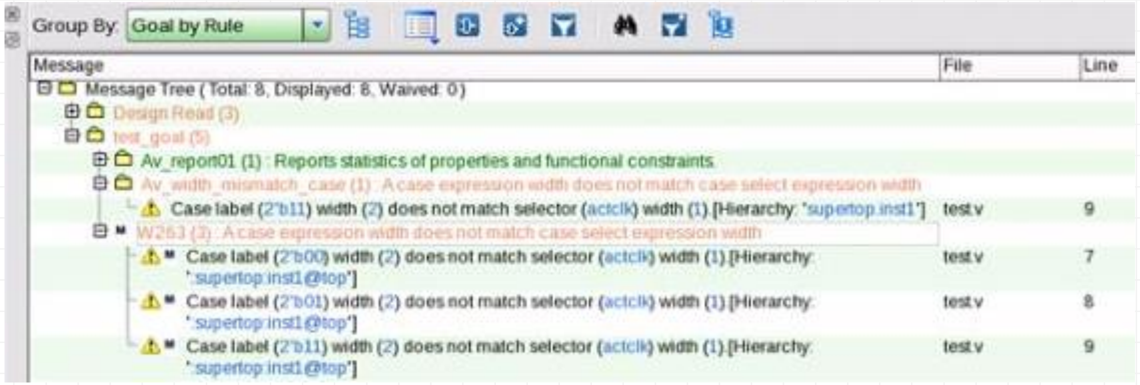
- Av_width_mismatch_case (A case expression width does not match case select expression width)
  - module top(input actclk);

```
module top(input actclk);
wire clk;
wire d1; reg q; reg [2:0]e;

always @ (actclk or clk or d1)
case (actclk)
2'b00 : e <= d1;
2'b01 : e <= d1 + 1;
2'b11 : e <= d1 + 2;
endcase

endmodule

module supertop(wire actclk, d, input [2:0]in_1, in_2, output reg q);
top inst1(actclk);
endmodule
```



```
Message                                                                                              File     Line
Message Tree ( Total: 8, Displayed: 8, Waived: 0 )
  Design Read (3)
  test_goal (5)
    Av_report01 (1) : Reports statistics of properties and functional constraints
    Av_width_mismatch_case (1) : A case expression width does not match case select expression width
      Case label (2'b11) width (2) does not match selector (actclk) width (1).[Hierarchy: 'supertop.inst1']   test.v   9
    W263 (3) : A case expression width does not match case select expression width
      Case label (2'b00) width (2) does not match selector (actclk) width (1).[Hierarchy:                     test.v   7
      ':supertop:inst1@top']
      Case label (2'b01) width (2) does not match selector (actclk) width (1).[Hierarchy:                     test.v   8
      ':supertop:inst1@top']
      Case label (2'b11) width (2) does not match selector (actclk) width (1).[Hierarchy:                     test.v   9
      ':supertop:inst1@top']
```

Examples of Lint Issues (cont..)

- Av_width_mismatch_port (An instance port connection has different width compared to the port definition)

  -
    ```
    module top(input clk);
    wire [3:0] sel;
    wire signed [4:0] a = -8;
    wire unsigned [4:0]a1;
    wire [4:0] b;
    wire [5:0] c;
    parameter p1 = 1;
    parameter p2 = 1;
    parameter p = 1;
    slave inst(.a(a) , .aa(a1[p1:0]) , .b(b[p2:0]) ,
      .c(c[p:0]));
    endmodule
    module slave (input [3:0]a , input [3:0]aa , input [3:0]b ,
    input [3:0]c);
    endmodule
    module supertop(input clk);
    top #(3,4,3) inst1(clk);
    top #(4,3,3) inst2(clk);
    endmodule
    ```
  - Violation 1
    - Incompatible width for port 'aa'(width 4 in module 'slave') on instance 'inst'(terminal width 5) [Hierarchy: 'supertop.inst2'].
  - Violation 2
    - Incompatible width for port 'b'(width 4 in module 'slave') on instance 'inst'(terminal width 5) [Hierarchy: 'supertop.inst1'].

Examples of Lint Issues (cont..)

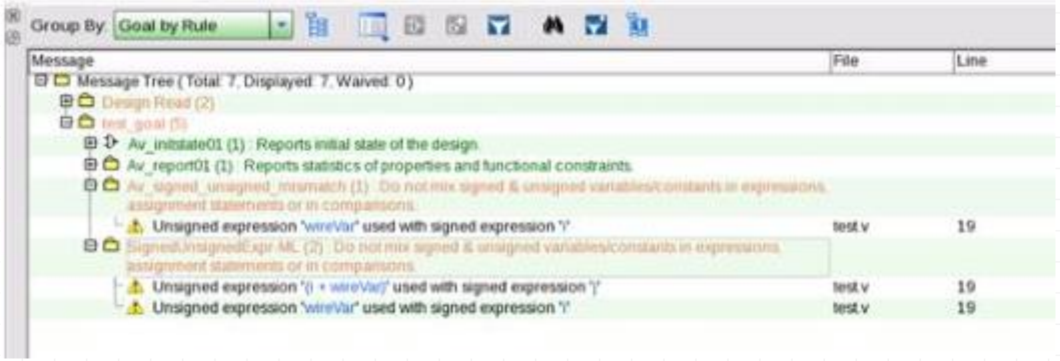
- Av_signed_unsigned_mismatch (Mixed signed and unsigned types)

  -
    ```
    module top( input clk, rst,
    input a1,
    output reg b1
    );

    //Av_signed_unsigned_mismatch
    test_sub test_sub_instance(a1,b1);

    endmodule

    module test_sub(d, q);
    input d;
    output q;
    reg q;
    reg r, clk;
    integer i, j;
    wire wireVar;
    wire a = i + wireVar + j;
    always @(posedge clk)
    begin
    if(r)
    q = d<<j;
    end
    endmodule
    ```



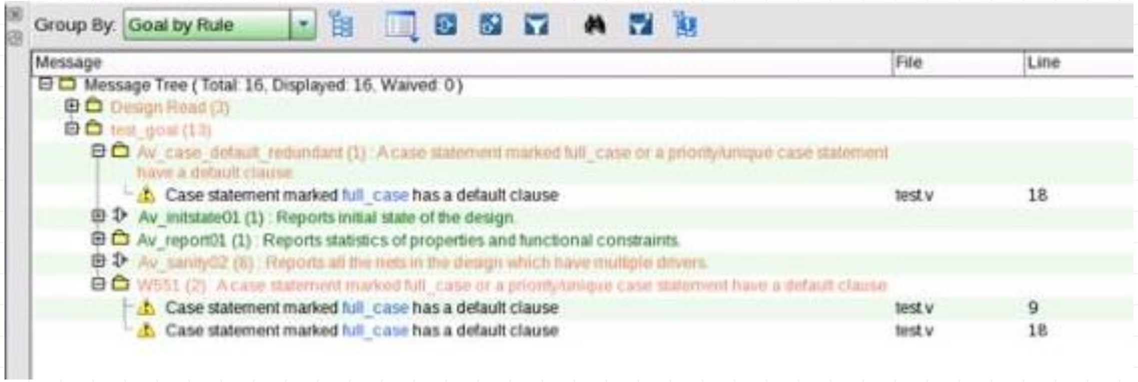
```
Message                                                                                          File     Line
Message Tree ( Total: 7, Displayed: 7, Waived: 0 )
  Design Read (2)
  test_goal (5)
    Av_initstate01 (1) : Reports initial state of the design
    Av_report01 (1) : Reports statistics of properties and functional constraints
    Av_signed_unsigned_mismatch (1) : Do not mix signed & unsigned variables/constants in expressions,
    assignment statements or in comparisons
      Unsigned expression 'wireVar' used with signed expression 'i'                              test.v   19
    SignedUnsignedExpr-ML (2) : Do not mix signed & unsigned variables/constants in expressions,
    assignment statements or in comparisons
      Unsigned expression '(i + wireVar)' used with signed expression 'j'                        test.v   19
      Unsigned expression 'wireVar' used with signed expression 'i'                              test.v   19
```

Examples of Lint Issues (cont..)

- Av_case_default_redundant (Ensure that a case statement marked full_case or a priority/unique case statement does not have a default clause)

  -
    ```
    module top( input clk, rst,
    input [4:0]a1,
    input [4:0]a6p,
    input [4:0]a7p,
    output reg [3:0]b6p,
    output reg [3:0]b7p);
    always @ (a1 or a6p or a7p)
    case (a1) //synopsys full_case
    2'b00 : b6p <= a6p;
    2'b01 : b7p <= a7p;
    2'b11 : b7p <= a6p + a7p;
    2'b10 : b7p <= a7p - a6p;
    default: b7p <= 2'b11;
    endcase

    always @ (a1 or a6p or a7p)
    casez (a1[1]) //synopsys full_case
    1'b0 : b6p <= a6p;
    1'b1 : b7p <= a6p + a7p;
    default: b7p <= 2'b11;
    endcase
    endmodule
    ```



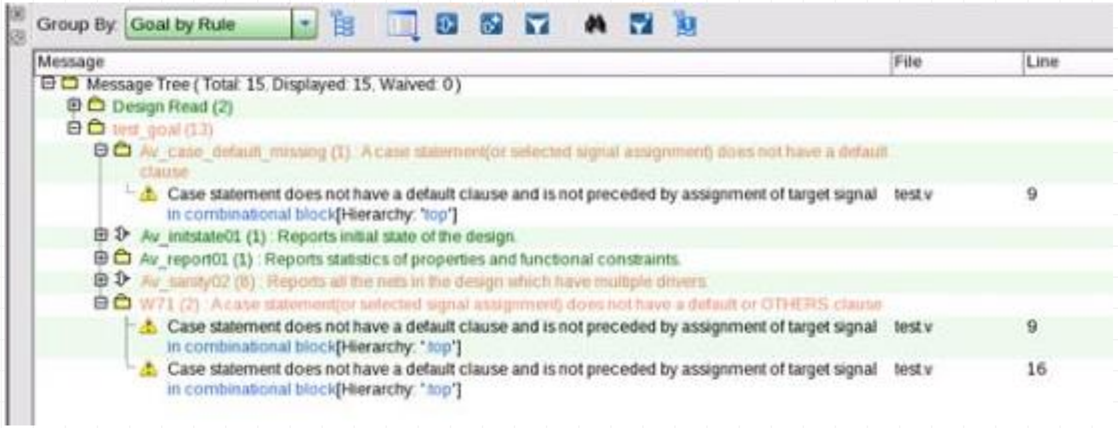
Message	File	Line
Message Tree (Total: 16, Displayed: 16, Waived: 0)		
Design Read (3)		
test_goal (13)		
Av_case_default_redundant (1) : A case statement marked full_case or a priority/unique case statement have a default clause		
Case statement marked full_case has a default clause	test.v	18
Av_initstate01 (1) : Reports initial state of the design		
Av_report01 (1) : Reports statistics of properties and functional constraints.		
Av_sanity02 (8) : Reports all the nets in the design which have multiple drivers.		
W551 (2) : A case statement marked full_case or a priority/unique case statement have a default clause		
Case statement marked full_case has a default clause	test.v	9
Case statement marked full_case has a default clause	test.v	18

Examples of Lint Issues (cont..)

- Av_case_default_missing (Ensure that a case statement or a selected signal assignment has a default clause)
  - module top(input clk, rst,
    input [4:0]a1,
    input [4:0]a6p,
    input [4:0]a7p,
    output reg [3:0]b6p,
    output reg [3:0]b7p);

    always @ (a1 or a6p or a7p)
    case (a1)
    2'b00 : b6p <= a6p;
    2'b01 : b7p <= a7p;
    2'b11 : b7p <= a6p + a7p;
    endcase

    always @ (a1 or a6p or a7p)
    case (a1[0])
    1'b0 : b6p <= a6p;
    1'b1 : b7p <= a6p + a7p;
    endcase
    endmodule



```
Message                                                                                                   File     Line
Message Tree ( Total: 15, Displayed: 15, Waived: 0 )
  Design Read (2)
  test_goal (13)
    Av_case_default_missing (1) : A case statement(or selected signal assignment) does not have a default
    clause
      Case statement does not have a default clause and is not preceded by assignment of target signal   test.v   9
      in combinational block[Hierarchy: 'top']
    Av_initstate01 (1) : Reports initial state of the design.
    Av_report01 (1) : Reports statistics of properties and functional constraints.
    Av_sanity02 (8) : Reports all the nets in the design which have multiple drivers
    W71 (2) : A case statement(or selected signal assignment) does not have a default or OTHERS clause
      Case statement does not have a default clause and is not preceded by assignment of target signal   test.v   9
      in combinational block[Hierarchy: ':top']
      Case statement does not have a default clause and is not preceded by assignment of target signal   test.v   16
      in combinational block[Hierarchy: ':top']
```

Examples of Lint Issues (cont..)

- Av_dontcare_mismatch (Use of don’t-care except in case labels may lead to simulation/synthesis mismatch)

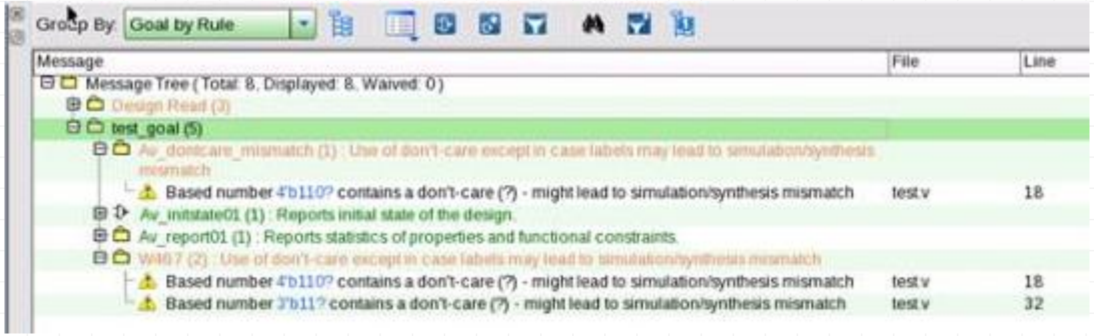
  -
    ```
    `define S0 2'b00
    `define S1 2'b01
    `define S2 2'b10
    `define S3 2'b11

    module top( input clk, rst,
    output reg [3:0] out0,
    output reg [2:0] out1
    );

    reg [1:0] state1;
    reg [1:0] state2;
    reg [1:0] state3;
    reg [1:0] state4;

    always @ (out0)
    begin
    out0 <= 4'b110?;
    end

    always @(posedge clk or posedge rst) begin
    if(rst) begin
    state3 <= `S0;
    end
    else begin
    priority case (state3)
    `S0: state3 <= `S1;
    `S1: state3 <= `S2;
    `S2: state3 <= `S0;
    default: begin
    state3 <= `S3;
    out1 = 3'b11?;
    end
    endcase
    end
    end
    endmodule
    ```



Group By: Goal by Rule

```
Message                                                                                        File     Line
Message Tree ( Total: 8, Displayed: 8, Waived: 0 )
  Design Read (3)
  test_goal (5)
    Av_dontcare_mismatch (1) : Use of don't-care except in case labels may lead to simulation/synthesis
    mismatch
      Based number 4'b110? contains a don't-care (?) - might lead to simulation/synthesis mismatch  test.v   18
    Av_initstate01 (1) : Reports initial state of the design.
    Av_report01 (1) : Reports statistics of properties and functional constraints.
    W467 (2) : Use of don't-care except in case labels may lead to simulation/synthesis mismatch
      Based number 4'b110? contains a don't-care (?) - might lead to simulation/synthesis mismatch  test.v   18
      Based number 3'b11? contains a don't-care (?) - might lead to simulation/synthesis mismatch   test.v   32
```

Examples of Lint Issues (cont..)

- INC_SENS_LIST : If a signal is referenced and not used in the sensitivity list of the block, it reports the failure.

Lint Setup Example

- Any lint setup requires the following basic inputs:
  - Filelist – list of all RTL files of the design on which lint is to be run
  - Waiver list – A file with a list of all issues reported by the toolthose need to be waived off
  - BlackBox list – A file with a list of all the modules in the design which on which Lint is not supposed to run
  - Run script – A shell script to provide inputs to and invoke the tool

Lint Setup Example.. (cont)

- Shell script example for providing inputs to and invoking the tool:

```
- <tool_path> \
        -top <top_module_name> \
        -wd <working directory path> \
        -wf <waiver file> \
        -cfg <configuration file>\
        -rep <report file name>\
        -od <output directory> \
        -f <filelist> \
        -bb <blackbox list>
        > & <log file path>
```

Lint Setup Example.. (cont)

- Changing the Severity of SpyGlass Built-In Rules

```
set_option overloadrules <rule-name>+<lang>+severity=<severity-label>
```

```
set_goal_option overloadrules <rule-name>+<lang>+severity=<severity-label>
```

Where,

<rule-name>	: This argument is the name of the rule being overloaded.
<lang>	: This argument is the language for which you want to overload a rule. The allowed values are VHDL, Verilog, Verilog+VHDL [case-insensitive]. This argument is optional. Omitting this option is applicable for all languages.
<severity-label>	: This argument is the overloaded severity label. The severity label can be one of the SpyGlass predefined severity labels: Fatal, Error, Warning, Info, and Data. One of the severity labels defined in the corresponding product, or a user-specified severity label defined using the define_severity command.

If the specified severity label is not any of these labels, SpyGlass auto-registers the severity label under the ERROR severity class.

Lint Setup Example.. (cont)

Waiver file

- A waiver file (.awl file) is used to waive messages reported after SpyGlass runs.
- It is an SGDC-format file that contains specifications of the waive constraint that is used to waive specific types of messages.
- You can specify waiver files to SpyGlass through GUI, project file, or batch.

Lint Setup Example.. (cont)

Effects of Waiving Messages

- Waiving message(s) affect the following:
  - SpyGlass results summary that is generated at the end of a SpyGlass analysis run
  - Count of waived messages is not added in message counts in the SpyGlass results summary. Instead, the following message appears to indicate the number of such messages:

    Suppressed 20 messages (5 waived)

  In the above example, 20 messages are suppressed due to message waiver or rule message over limit settings. Out of these suppressed messages, 5 messages were suppressed due to message waiver.

- Violation database
  - SpyGlass modifies the rule severity of waived messages to waiver[original-severity] in the violation database. For example, consider the following violation message:

    W127@@@@Warning@@rules_w127_1.v@@31@@1@@5@@Delay value should not contain X or Z

  If you waive the above message, and the issue reported by the corresponding rule message is present in a design, the following message is written in the violation database:

    W127@@@@Waiver[Warning]@@rules_w127_1.v@@31@@1@@5@@Delay value should not contain X or Z

- SpyGlass shows the severity of the waived message as waiver[original-severity].
- Waiver report
  - The Waiver report lists all waived messages. You can view this report from the Reports menu option.

Lint Analysis

- Though a lint report points to the issues very clearly; every issue reported may not be a real issue

- For example: A tool might report width mismatch for an LHS <= RHS assignment.

However, it may not be a real issue for something like a counter design i.e. cntr <= cntr_1 +1’b1; because, this statement has an inherent width mismatch of 1 bit; i.e. even if cntr and cntr_1 have same width, the +1’b1 makes the widths unequal for a usecase.

Lint Analysis (cont..)

- It is advisable to analyze each and every issue reported by the tool irrespective of its severity – Error or Warning

- Nevertheless, some of the most important issues to look for are –
  - Inferring latches
  - Width mismatches
  - Unused wires (These can be typos)
  - Unconnected outputs (These can be typos or you might have missed connections)
  - FSM – unreachable state, Deadlock condition
  - Default missing in case statement

Thank You

L&T Technology Services

Parikshit Dhodapkar & Krishnakant Patil

Project Lead