Design Rule Check with nLint & Completeness Check of Testbenches with Assertain

Advisor: 李昆忠 Kuen-Jong Lee
Presenter: 連唯証 Wei-Cheng Lien
E-mail: lwz98@beethoven.ee.ncku.edu.tw

Reference:
1. SystemVerilog rules supplied with Assertain v 2006.01
2. nLint Rule Category (ver1.0)
3. IP Qualification Guidelines (IPQ)
4. Reuse Methodology Manual (RMM)
Outline

• Introduction to nLint
• Illustration of execution flow of nLint
• Introduction to Assertain
• Illustration of execution flow of Assertain
• Conclusions
Outline

• Introduction to nLint
• Illustration of execution flow of nLint
• Introduction to Assertain
• Illustration of execution flow of Assertain
• Conclusions
Introduction (1/2)

• An SOC design usually contains many IP (Intellectual Property) cores.
• Issue of system integration
Introduction (2/2)

• Establishment of unified design rules for
  – Coding style
    ◆ Readability
  – Simulation
  – Synthesis
  – Others for ease of integration
• Guarantee quality of IPs
• nLint
  – Developed by SpringSoft
  – Design rule checker
Outline

- Introduction to nLint
- Illustration of execution flow of nLint
- Introduction to Assertain
- Illustration of execution flow of Assertain
- Conclusions
Start nLint

- Put some setup files (Ex. IPQ.rs, udr.ini, action.tcl) in your working directory.
- Unix% nLint –udr . –rs IPQ.rs –drm –gui &
- Unix% nLint –gui &
Rules to Be Checked

- Menu → Tools → Rule Organizer

Import IPQ
Design Rule Check

• Use nLint tool (include by Debussy) and the Verilog Coding Guideline from IPQ rules to check your designs and modify parts of code to match the coding guidelines.

• Three level definitions in the IPQ rules
  - **Mandatory 1 (M1, Error)** – Rules must be followed. If not, design must be fixed. Ex. Verilog and VHDL keywords are prohibited …etc.
  - **Mandatory 2 (M2, Warning)** – Rules must be followed. If not, documentation must be provided. Ex. A file must contain at most one module unit …etc.
  - **Recommended (R, Warning)** – Rules are recommended to be followed in the designs. Ex. Keep line length within 72 characters …etc.
Load Verilog Code (1/2)
Load Verilog Code (2/2)

```verilog
module traffic_light(sp_yellow, HG, HY, HR, HLEFT, FG, FY, FR, ST, ST_out, op);
input sp, HG, HY, HR, HLEFT, FG, FY, FR, ST, ST_out;
output sp_yellow, HLEFT, HG, HY, HR, FG, FY, FR, ST, ST_out;
reg ST, ST_out, // start timer to count the time passed
reg[2:0] state, nextstate;

parameter s1=3'b000, s2=3'b001, s3=3'b010, s4=3'b011, s5=3'b100, s6=3'b11;

// define the special light of different state
assign sp_yellow=(state==s1);
assign HG=(state==s3);
assign HY=(state==s4);
assign HR=(state==s5) || state==s6 || state==s2);
assign FG=(state==s5);
assign FY=(state==s6);
```

Copyright (C) 1995 - 2006 by Novas Software, Inc.

*** Working Directory [/home10/PHD/tyhsieh/System-Design-class/2006/nLint/] ***

Import design
Analyzing...
source file "mytraffic_light.v"
End of importing design. Memory used 2025984. Time used 3.
Run nLint Check

1. Lint Design (Compile)

Lint Design

---

```
module traffic_light(sp_yellow, HG, HY, HR, HLEFT, P0, FY, PR, ST, ST_out, op
input spart, ca0, TS1, TLEFT, reset, clk;
output sp_yellow, HLEFT, HG, HY, HR, F0, FY, PR, ST, ST_out;
reg ST, ST_out, //start timer to count the time passed
reg[2:0] state, nextstate;

parameter s1=3'b000, s2=3'b001, s3=3'b010, s4=3'b011, s5=3'b100, s6=3'b11

//define the special light of different state
assign sp_yellow=(state==s1);
assign HG=(state==s3);
assign HY=(state==s4);
assign HR=(state==s5) || state==s6 || state==s2);
assign F0=(state==s5);
assign FY=(state==s6);
```
nLint Check Results (1/4)
nLint Check Results (2/4)

- Clock Domain Analysis
- Total: 14 Error(s), 3 Warning(s)
  - Simulation: 7 Error(s)
    - 22139: Constant Event Expression
    - Constant Event Expression
      - Simulation, Language Construct: 6 Error(s)
    - :2301:1: Incomplete Sensitivity List
      - Incomplete Sensitivity List
        - Simulation, Synthesis: 1 Error(s)
          - mytraffic_light.v89: Error: signal "ST_out" should be included in the sensitivity list.
  - Synthesis: 1 Error(s)
  - Design Style: 3 Warning(s)
  - Language Construct: 6 Error(s)
nLint Check Results (3/4)

- Clock Domain Analysis
- Total: 14 Error(s), 3 Warning(s)
- Simulation: 7 Error(s)
  - 22138: Constant Event Expression - Simulation, Language Construct - 6 Error(s)
  - 23011: Incomplete Sensitivity List - Simulation, Synthesis - 1 Error(s)
  - mytraffic_light.v(89): Error: signal "ST_out" should be included in the sensitivity list.

Click

- Design Style: 3 Warning(s)
- Language Construct: 5 Error(s)

VLSI System Design

nLint & VN-Cover EDA Tools .15

NCKUEE-W.C. Lian
Click the “F1” bottom on the error/warning message:

23011 Incomplete Sensitivity List

Message
<filename> (<line no.>): <severity> <rule no.>: signal "%s" should be included in the sensitivity list.

Configurable Parameter
Rule group: Synthesis, Simulation;
Argument type: none;
Default severity: Level3 (Error)

Description
The rule checks whether all right-hand-side variables of a combinational always block are included in the sensitivity list.

Example
(Verilog)
Glue Logic

```plaintext
shift.v(13): glue logic shift:Always0#Sig0p0:13:13:Mux found in top module

assign shiftRes = (rl) ? (ALUData << shamt):(ALUData >> shamt);
```

See Also:

- shift.v(13): glue logic shift:Always1#Sig0p0:13:13:ShiftLeft found in top module
- shift.v(13): glue logic shift:Always2#Sig0p0:13:13:ShiftRight found in top module
- shift.v(17): glue logic shift:Always3#Sig0p1:15:15:Adder found in top module
- shift.v(17): glue logic shift:Always4#Sig0p1:15:15:Sub found in top module

- [Synthesis Warning] No glue logic allowed in **top module**.
Signals in sensitivity list

- All sensitivity signals of block must include in the list.
- [Simulation Error]

```
+ Simulation -2 Error(s)
+ SIM 4 - Extra Signal in Sensitivity List - Simulation - 2 Error(s)
   ALU.v(35): Error: signal "carry" should not be included in the sensitivity list.
   ALU.v(35): Error: signal "ALUOut" should not be included in the sensitivity list.

ALU.v(35): signal "carry" should not be included in the sensitivity list.
  (ALU.v)

34  **/
35   always@(SUBResult or carry or ALUOut or ALUControl or R_Data1 or ALUData or result0 or result1 or re
36     begin

See Also:
ALU.v(35): signal "ALUOut" should not be included in the sensitivity list.
```
Naming Conversions

```plaintext
40  | shift  u_shift1 (.result(result_0),.ALUData(ALUData),.shamt(shamt),.rl(rl1),.rs(rs1));
41  | shift  u_shift2 (.result(result_1),.ALUData(ALUData),.shamt(shamt),.rl(rl0),.rs(rs1));
42  | shift  u_shift3 (.result(result_2),.ALUData(ALUData),.shamt(shamt),.rl(rl1),.rs(rs0));
43  | shift  u_shift4 (.result(result_3),.ALUData(ALUData),.shamt(shamt),.rl(rl0),.rs(rs0));

ALU.v(40): 4 Warning(s)

+ Warning 30006: Instance name u_shift1 is not related to module name shift with prefix u_ (User Defined) :
+ Warning 30003: port name rs doesn’t match port instance name rs1 (User Defined) :
+ Warning 30003: port name rl doesn’t match port instance name r1l (User Defined) :
+ Warning 30003: port name result doesn’t match port instance name result_0 (User Defined) :
```
Connection Ports by Ordered List

26  |  shift sl(result0, ALUData, shamt, 1'b1, 1'b1); // shift left

ALU.v(26): Error: port "result0" should be connected by name.
ALU.v(26): Error: port "ALUData" should be connected by name.
ALU.v(26): Error: port "shamt" should be connected by name.
ALU.v(26): Error: port "1'b1" should be connected by name.
ALU.v(26): Error: port "1'b1" should be connected by name.
ALU.v(27): Error: port "result1" should be connected by name.
ALU.v(27): Error: port "ALUData" should be connected by name.
ALU.v(27): Error: port "shamt" should be connected by name.
ALU.v(27): Error: port "1'b0" should be connected by name.

ALU.v(26): port "result0" should be connected by name.
result0 (ALU.v)

17
18  wire[31:0] result0, result1, result2, result3;
19  wire[32:0] SUBResult; // for shift and rotate

result (shift.v)

1 module shift(result, ALUData, shamt, rl, rs);
2
Multi-line Comment

• [User defined warning]
Lost Comments

• **Assign** statements and **always** blocks must have comments in the head.

• [User Defined Warning]
Verilog and VHDL keywords are prohibited

- Verilog and VHDL keywords must not be used for identifier naming.
- [Naming Convention Error]

Control_Unit.v(23): 1 Error(s)

- Error NC.19: a VHDL reserved word should not be used as object name 'signal'. (Naming Convention) : Control_Unit
  22   /* wire judge;
  23   reg[12:0] signal;
  24   reg[12:0] p;
Fan-out limitation

- Reference: PrimeTime Commands Guideline
- [Design Style Warning] can be ignored in nLint and deal in synthesis.
- Don’t add buffer by yourself!

To remove maximum fanout limits from designs or ports, use `remove_max_fanout`.

**EXAMPLES**

The following example sets a maximum fanout limit of 2.0 units on ports "IN*".

```
pt_shell> set_max_fanout 2.0 [ge_ports "IN*"]
```

The following example sets the default maximum fanout limit of 5.0 units on the current design.

```
pt_shell> set_max_fanout 5.0 [current_design]
```
File header

- “+FHDR” and “-FHDR” as a file header boundary tag
- Each field must write as capitalization

```
1 //+FHDR---------------------------------------------
2 // (C) Copyright NCKU
3 // All Right Reserved
4 //-------------------------------------------------
5 // FILE NAME: cpu.v
6 // AUTHOR: W.C. Lian
7 // CONTACT INFORMATION: lwz98@beethoven.ee.ncku.edu.tw
8 //-------------------------------------------------
9 // RELEASE VERSION: V1.0
10 // VERSION DESCRIPTION: top module
11 //-------------------------------------------------
12 // RELEASE DATE: 11/19/2007
13 //-------------------------------------------------
14 // PURPOSE: cpu top module
15 //-------------------------------------------------
16 // PARAMETERS: None
17 //-------------------------------------------------
18 module cpu(next_pc, ALUOut, writeRData, w_Reg, osignal, ir, R_I
```
Errors (4)

- Register should have a set or reset signal.
- Module name should be the same file name.
- Wire line should be explicitly declared.
- Need comment for this functional block
Example - 1500_lib_2004.v (2/3)

• Warnings (14)
  – More than one top module detected.
  – Output signal should not be referenced inside the module.
  – More than one clock signal detected in the module.
  – The port is not in order with port update.
  – The signal has no load. (need to check)
  – Asynchronous clock detected.
  – The input port and output port should not be connected directly.
Example - 1500_lib_2004.v (3/3)

• Warnings (14)
  – The port should be connected by name.
  – Register inferred on signal pin not in library
  – Glue logic found in top module
  – Incomplete file header (10 fields)
  – Do not use multi-line comment.
  – Port name does not match port instance name.
  – Instance name is not related to module name prefix u_.

VLSI System Design
nLint & VN-Cover EDA Tools .28
NCKUEE-W.C. Lian
Fix Warnings/Errors (1/12)

Level: M2
Warning: **Register inferred on signal not in library**

Verilog code:
module aa(…….)
if (hold)
    pc_out_tmp <= pc_out_tmp

Level: M2
Warning: **Output signal “sel_br” should not be referenced inside the module.**

Verilog code:
output sel_br;
…….
Assign clk_br = sel_br & clk_br
Fix Warnings/Errors (2/12)

Warning: **more than one clock signal detected in the module, clocks.**

Verilog Code:
```verilog
wire[3:0] ir;
wirecell
    ir_3(.po(ir[3]),.so(s_0),.si(si),.clk(clk),.update(update),.rst(rst));
......
module
    always @(posedge clk or negedge rst)
        begin  .................  end
always @(posedge update or negedge rst)
    begin  .................  end
```

Error: **Register “shift_reg_out” should have a set or reset signal.**

Verilog Code:
```verilog
reg shift_reg_out;
always @(posedge clk) // WARNING
    begin
        shift_reg_out <= shift_reg_in;
    end
```

Modified code:
```verilog
reg shift_reg_out;
always @(posedge clk or posedge rst)
    begin if (reset)
        shift_reg_out <= 1'b0;
    else
        shift_reg_out <= shift_reg_in;
    end
```

VLSI System Design
nLint & VN-Cover EDA Tools .30
NCKUEE-W.C. Lian
Level:M1
Warning: **signal “shift_ir” has no load.**

Verilog Code:

```verilog
wire shift_ir;
............
assign shift_ir = select_wr & shift;
```

Please make sure if the shift_ir wire is used later.

Level:M2
Warning: **Asynchronous “rst” detected.**

Verilog Code:

```verilog
always @(posedge clk or negedge rst)
if(!rst)
............
else
............
```

Please check if the circuit is asynchronous.
Warning: **input port “transfer” and output port “transfer_by” should not be connected directly.**

Verilog Code:
```
input    transfer;
output transfer_by;
assign transfer_by = transfer
```

Please check if the input pin is connected with output pin.

Warning: **port “po,so,si,clk,update,rst” should be connected by name.**

Verilog Code:
```
wirecell u_wircell(po,so,si,clk,update,rst);
```

Modified code:
```
wirecell u_wircell(.po(po),
                   .so(so),
                   .si(si),
                   .clk(clk),
                   .update(update),
                   .rst(rst));
```
Fix Warnings/Errors (5/12)

Tree has only one root!

Level: M2
Warning: more than one top module detected, top module name: “wrapper_scan_mux, By pass.”

Verilog code:
module name:
  wrapper_scan_mux
Bypass
dr_mux
wir
wo_mux
wrapper_controller
wircell
WC_uni_1
WC_uni_2

Each verilog file should have only one top module.

Modified code:
  wrapper_scan_mux.v
  Bypass.v
dr_mux.v
  wir.v
  wo_mux.v
  wrapper_controller.v
  wircell.v
  WC_uni_1.v
  WC_uni_2.v
Fix Warnings/Errors (6/12)

Level: M2
Warning: **Glue logic found in top module**

Verilog Code:

module Top(t0,t1,t2,t3);
    ......
    aa_00 a_0(a0,a1,a2,a3);
    assign out_1= (sel & i1) | (~sel & i0);
    aa_01 a_1(b0,b1,b2,b3);
    ......
endmodule

Modified code: n_00.v

module Top(t0,t1,t2,t3);
    ......
    aa_00 a_0(a0,a1,a2,a3);
    new n_00(i0,i1,sel,out_1);
    aa_01 a_01(b0,b1,b2,b3);
    .........
module n_00(i0,i1,sel,out_1);
    output out_1;
    wire o1;
    wire o2;
    wire o3;
    assign o1 = sel & i1;
    assign o2 = ~sel;
    assign o3 = o2 & i0;
    assign out_1 = o1 | o2;

The word length of ports must be larger than 1
Warning: port “so” is not in order with port update.

Verilog Code:
Module
wir(clk,rst,si,so,update,bypass,sample,extest,s_scan,p_scan,bist);

input clk,rst;
input si;
input update;
output so;
output bypass;
output sample;
output extest;
output s_scan;
output p_scan;
output bist;

Modified code:
module wir(clk,rst,si,so,update,bypass,sample,extest,s_scan,p_scan,bist);
input clk;
input rst;
input si;
output so;
input update;
output bypass;
output sample;
output extest;
output s_scan;
output p_scan;
output bist;
Level: M1
Error: module name "wrapper_scanmux" should be the same file name.

Verilog Code:
wrapper_scan_mux.v

module wrapper_scanmux (i0,i1,sel,o);
........

Modified code: wrapper_scanmux.v

Module
wrapper_scanmux(i0,i1,sel,o);
Level: M1
Error: **wire “s_0” should be explicitly declared.**

Verilog Code:
```verilog
code
wire [3:0] ir;
wircell
    ir_3(.po(ir[3]),.so(s_0),.si(s_i),.clk(clk),.update(update),.rst(rst));
wircell
    ir_2(.po(ir[2]),.so(s_1),.si(s_0),.clk(clk),.update(update),.rst(rst));
wircell
    ir_1(.po(ir[1]),.so(s_2),.si(s_1),.clk(clk),.update(update),.rst(rst));
wircell
    ir_0(.po(ir[0]),.so(s_2),.si(s_2),.clk(clk),.update(update),.rst(rst));
```

Modified code:
```verilog
wire [3:0] ir;
wire s_0; Wire s_1; Wire s_2;
wircell
    ir_3(.po(ir[3]),.so(s_0),.si(s_i),.clk(clk),.update(update),.rst(rst));
wircell
    ir_2(.po(ir[2]),.so(s_1),.si(s_0),.clk(clk),.update(update),.rst(rst));
wircell
    ir_1(.po(ir[1]),.so(s_2),.si(s_1),.clk(clk),.update(update),.rst(rst));
wircell
    ir_0(.po(ir[0]),.so(s_2),.si(s_2),.clk(clk),.update(update),.rst(rst));
```
Error: **need comment for this functional block.**

Verilog Code:

```verilog
class o1 = sel & i1;
class o2 = ~sel;
class o3 = o2 & i0;
class o = o1 | o2;
```

Warning: **do not use multi-line comment.**

Verilog Code:

```verilog
/* ..... 
   ..... 
   .....*/
```

Modified code:

```verilog
//comment1
assign o1 = sel & i1;
//comment2
assign o2 = ~sel;
//comment3
assign o3 = o2 & i0;
//comment4
assign o = o1 | o2;
```

Modified code:

```verilog
//..... 
//..... 
//..... 
```
Warning: **port name “po”, “so” does not match port instance name ir[3].**

Verilog Code:

```verilog
wircell
  ir_3(.po(ir[3]),.so(s_0),.si(si),.clk(clk),.update(update),.rst(rst));
```

Modified code:

```verilog
wircell
  ir_3(.po(po),.so(so),.si(si),.clk(clk),.update(update),.rst(rst));
```
Warning: instance name `wir_cell00` is not related to module name `wircell` prefix `u_`.

Verilog Code:
```
wirecell `wir_cell00
  (.po(po),.so(so),.si(si),.clk(clk),.update(update),.rst(rst));
```

Modified code:
```
wirecell `u_wircell
  (.po(po),.so(so),.si(si),.clk(clk),.update(update),.rst(rst));
```

**Ex. u_wircell_1 (Warning)**

**Naming Convention**

Ex. `Xxx_cs` (Current State)
```
Xxx_ns (Next State)
rst_a
```
(Asynchronous reset)
```
clk_n (neg clock)
```
VN-Cover

Reference:
1. SystemVerilog rules supplied with Assertain v 2006.01
2. Assertain User Guide (Version 2006.01)
Introduction

- The completeness of testbenches
  - Guarantee correct function of our design under any conditions
  - Shorten time-to-market
- VN-Cover
  - Developed by TransEDA
  - Check the completeness of a testbench
  - Identify the code segments that are not activated during simulation.

- `Unix% source /usr/cad/transeda/CIC/license.csh`
Completeness Measurement

- Code coverage
  - Statement coverage
  - Toggle coverage
  - Condition coverage
  - Branch coverage
  - Path coverage
- Finite state machine coverage
  - State coverage
  - Arc coverage
Statement Coverage

• How many of the total lines of code were executed?
• Ex:

```verilog
if (parity == ODD || parity == EVEN) begin
    tx <= compute_parity(data, parity);
    #(tx_time);
end
```

- `tx <= 1'b0;`
- `#(tx_time);`

- `if (stop_bits == 2) begin`
  - `tx<=1'b0;`
  - `#(tx_time);`
- `end`

When `parity!=ODD & Parity!=EVEN & stop_bits=2`

• Statement Coverage = $\frac{6}{8} = 75\%$
Path Coverage

• All possible ways you can execute a sequence of statements

• Ex:

```plaintext
if (parity == ODD || parity == EVEN) begin
    tx <= compute_parity(data, parity);
    #(tx_time);
end

if (stop_bits == 2) begin
    tx <= 1'b0;
    #(tx_time);
end
```

This path is not exercised. Hence path Coverage=75%
Statement Coverage=100%
Finite State Machine Coverage
Outline

• Introduction to nLint
• Illustration of execution flow of nLint
• Introduction to VN-Cover
• Illustration of execution flow of VN-Cover
• Conclusions
Start VN Check (1/2)

- Unix% Assertain &
- Unix% vn &
Add File
Select Simulator

[Image of software interface for selecting simulators]
Define Coverage Criteria (1/3)
Define Coverage Criteria (2/3)
Define Coverage Criteria (3/3)

Assist by TransEDA Version: 2006.01

<table>
<thead>
<tr>
<th>Design</th>
<th>Language</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>.../slave_top.v</td>
<td>Verilog</td>
<td></td>
</tr>
<tr>
<td>.../slave_testbench.v</td>
<td>Verilog</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Static</th>
<th>Code</th>
<th>Dynamic</th>
<th>Assertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

Select a Verilog Command File

Command File: assertain.t

Criteria/Static Selection Key:

- ✗ None Selected / Off
- ✓ Some Selected
- ✗ All Selected / On

Compilation Completed

Compilation of the design has been completed. Please check messages in the Log window.
License Fail

- Assertion License Fail can’t effect the dynamic code coverage.
Simulate (1/2)
Simulate (2/2)
Results (1/5)

[Image of a software interface showing a results summary with code coverage details and a dialog box for selecting results files.]
Results (2/5)
Results (3/5)

1. Statement
2. Branch
3. Condition
4. Triggering
5. Toggle
6. Trace
7. Path
8. Excluded
Results (4/5)
Results (5/5)
Conclusions

- Quality of IP is critical for system integration
- nLint EDA tool to check design rules
- Writing of good testbenches for complete verification
- Assertain EDA tool to examine completeness of testbenches