Lab Session 2
Design and Simulation of Priority Encoder and Ripple Carry Adder

Advisor: Lih-Yih Chiou

Speaker: Sylvia

Date: 2021/3/3

EE

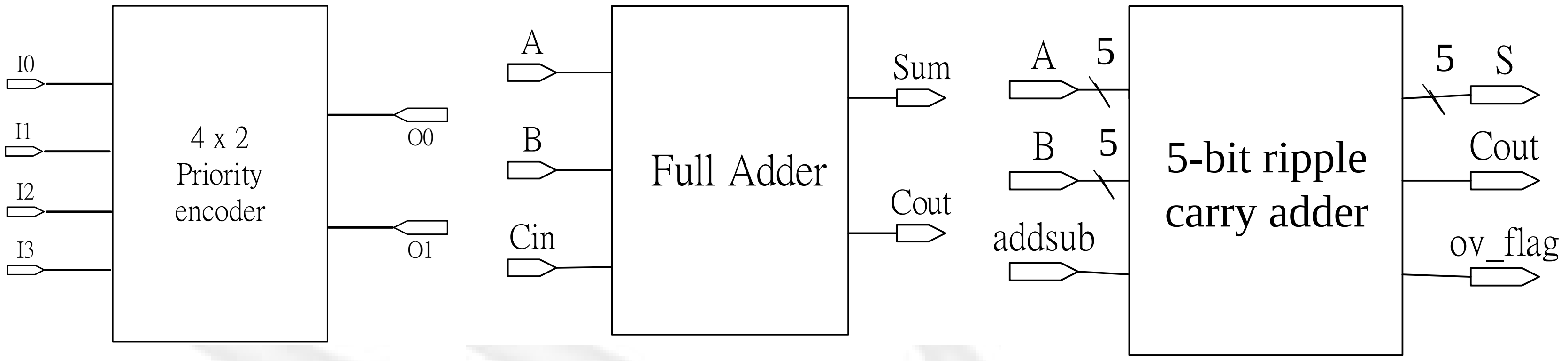
LPHPLAB VLSI Design LAB

Outline

- Introduction
- Design of priority encoder
  - Truth table and Boolean equation
  - Verilog code
- Design of adder
  - Half adder & Full adder
  - 5-bit ripple carry adder
- Testbench
- Compile & Simulation
- Lab Session 2
- Superlint Tutorial

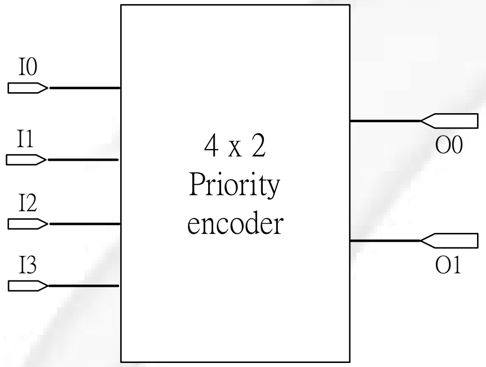
Introduction

- A priority encoder is a device that compresses several inputs into a smaller number of outputs.
- An adder is a digital circuit that performs addition of number.
- The following are block diagrams of 4-to-2 priority encoder, full adder and 5-bit ripper carry adder.



Truth table and Boolean equation

Block diagram



Boolean equation

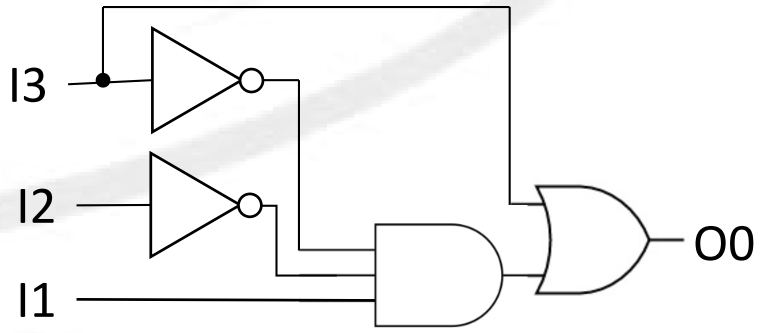
O0 = ~I3 ~I2 I1 + I3

O1 =

Truth table

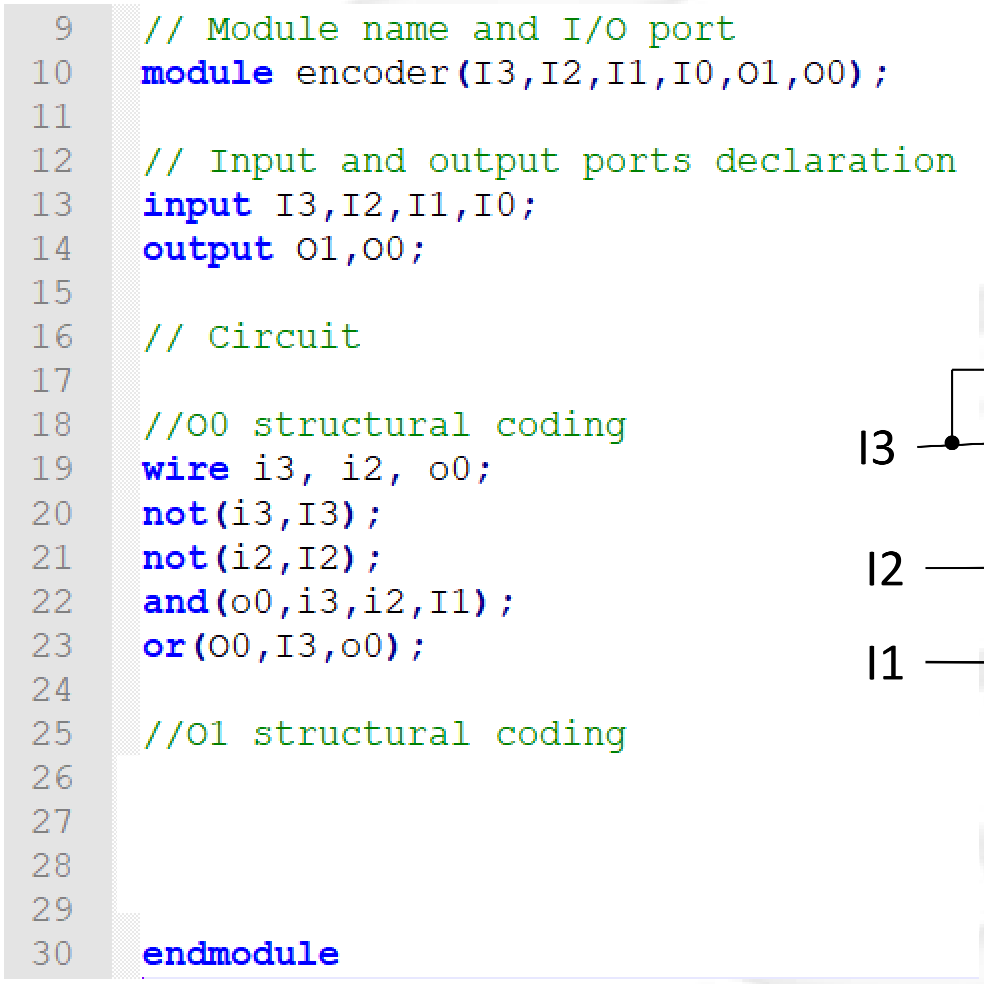
Inputs				Outputs	
I3	I2	I1	I0	O1	O0
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

Schematic view

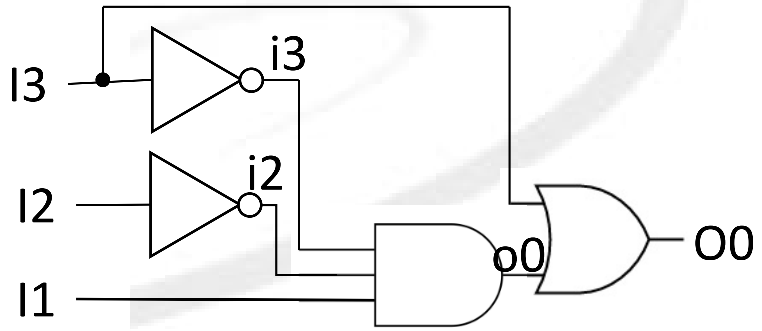


Verilog Code

- Implement O0 logic with Verilog code

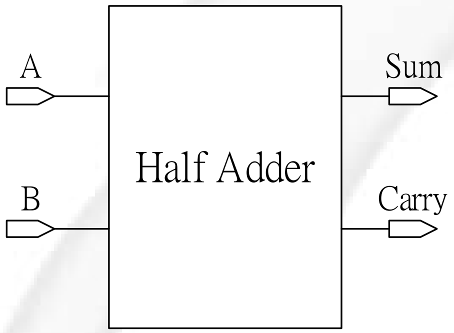


```
// Module name and I/O port
module encoder(I3,I2,I1,I0,O1,O0);

// Input and output ports declaration
input I3,I2,I1,I0;
output O1,O0;

// Circuit

//O0 structural coding
wire i3, i2, o0;
not(i3,I3);
not(i2,I2);
and(o0,i3,i2,I1);
or(O0,I3,o0);

//O1 structural coding




endmodule
```



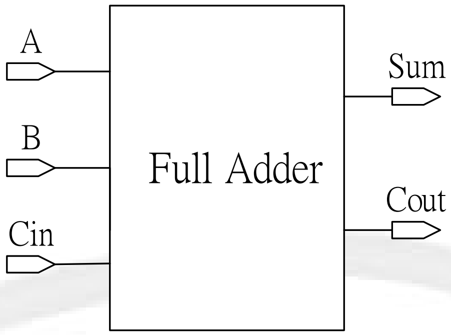
Half Adder & Full Adder

- True table of half adder



Inputs		Outputs	
A	B	C	S
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

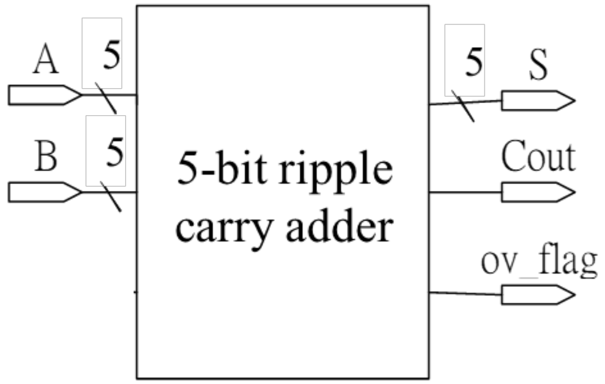
- True table of full adder

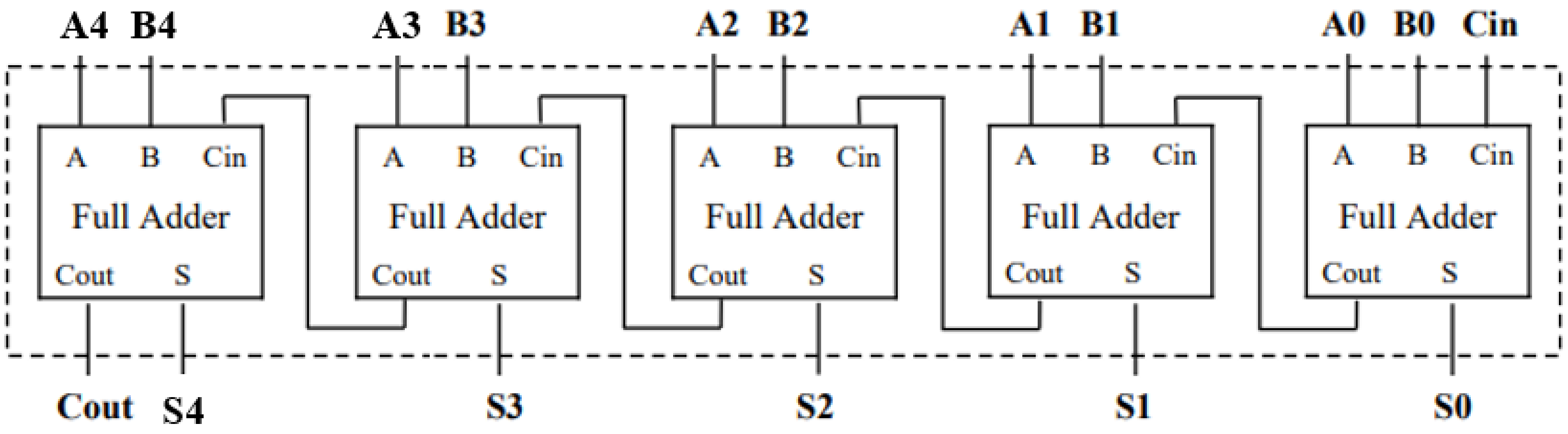


Inputs			Outputs	
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

5-bit Ripple Carry Adder(1/2)

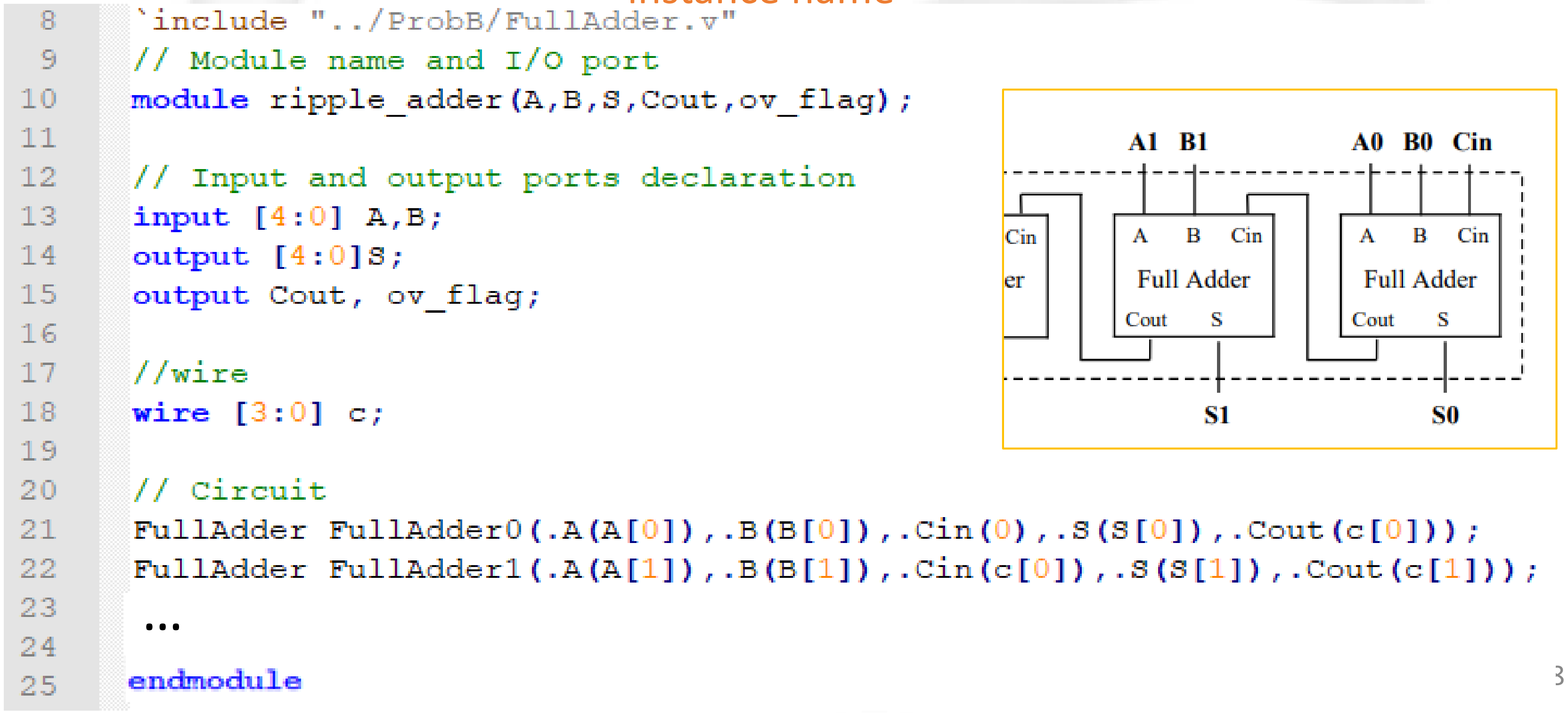
- Design a 5-bit ripple carry adder using five full adders.
  - Hierarchical Coding



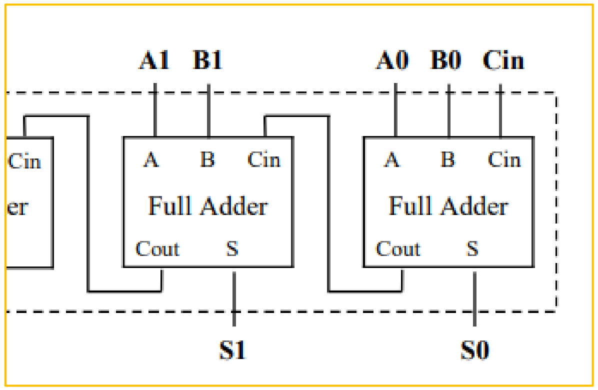


5-bit Ripple Carry Adder(2/2)

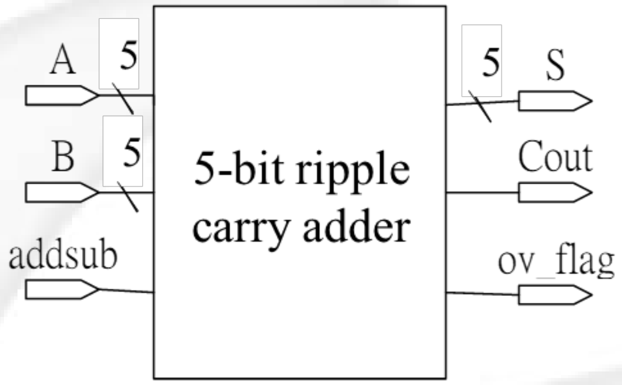
- Module instantiation
  - Positional mapping
    FullAdder FullAdder0(A, B, Cin, S, Cout);
  - Named mapping ← Recommended !!!
    FullAdder FullAdder0(.A(A), .B(B), .Cin(Cin), .S(S), .Cout(Cout));
    instance name



```
`include "../ProbB/FullAdder.v"
// Module name and I/O port
module ripple_adder(A,B,S,Cout,ov_flag);

// Input and output ports declaration
input [4:0] A,B;
output [4:0]S;
output Cout, ov_flag;

//wire
wire [3:0] c;

// Circuit
FullAdder FullAdder0(.A(A[0]),.B(B[0]),.Cin(0),.S(S[0]),.Cout(c[0]));
FullAdder FullAdder1(.A(A[1]),.B(B[1]),.Cin(c[0]),.S(S[1]),.Cout(c[1]));
 ...

endmodule
```



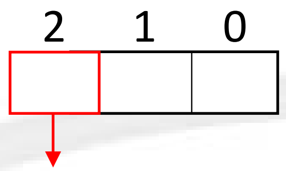
5-bit add/sub Ripple Carry Adder



Signal	Type	Bits	Description
A	Input	5	First operand
B	Input	5	Second operand
addsub	Input	1	If addsub is 0, operator is addition. If addsub is 1, operator is subtraction.
S	Output	5	Result after calculating
Cout	Output	1	Carry bit
ov_flag	Output	1	If result overflows, ov_flag pull to 1. Otherwise, ov_flag is 0.

Adder/Subtractor and Overflow

- addsub=0, X + Y = X + Y + 0
- addsub=1, X - Y = X + $\bar{Y}$ + 1 **(2's complement)**
- Overflow
  - 3-bit binary numbers
    - 2's complement, Range: -4 ~ 3
    - Out of range called overflow
  - Example (addsub=0)

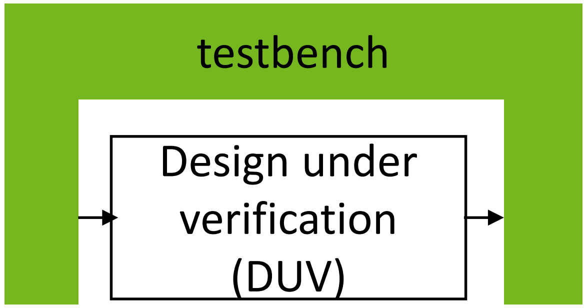


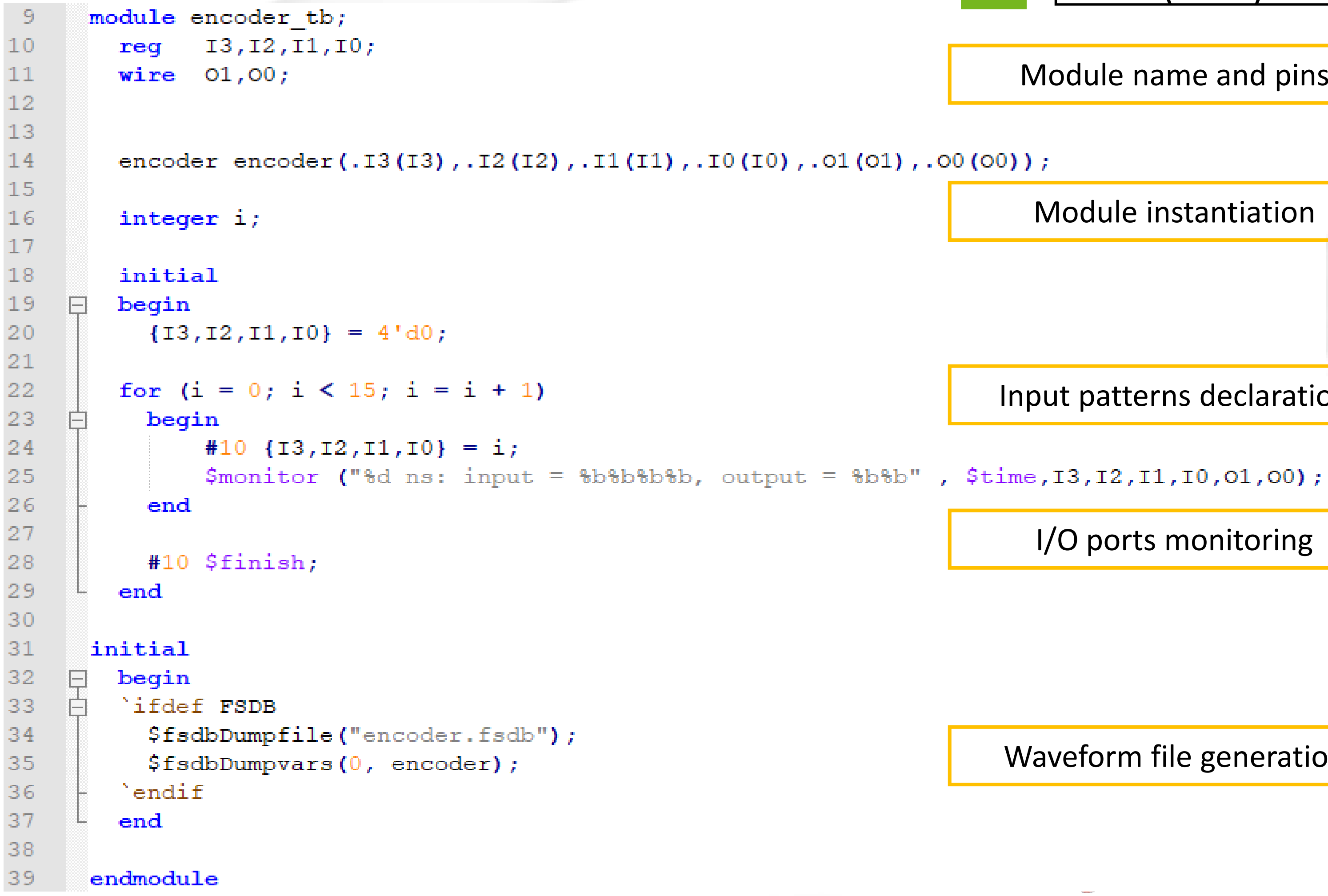
Carry:		0	1		
1			0	0	1
+ 3	+		0	1	1
+ 4			1	0	0

Carry:		1	0		
-3			1	0	1
+ -2	+		1	1	0
-5			0	1	1

Testbench

- Generate patterns to verify DUV





```verilog
module encoder_tb;
  reg   I3,I2,I1,I0;
  wire  O1,O0;


  encoder encoder(.I3(I3),.I2(I2),.I1(I1),.I0(I0),.O1(O1),.O0(O0));

  integer i;

  initial
  begin
    {I3,I2,I1,I0} = 4'd0;

  for (i = 0; i < 15; i = i + 1)
    begin
        #10 {I3,I2,I1,I0} = i;
        $monitor ("%d ns: input = %b%b%b%b, output = %b%b" , $time,I3,I2,I1,I0,O1,O0);
    end

    #10 $finish;
  end

initial
  begin
  `ifdef FSDB
    $fsdbDumpfile("encoder.fsdb");
    $fsdbDumpvars(0, encoder);
  `endif
  end

endmodule
```

Module name and pins

Module instantiation

Input patterns declaration

I/O ports monitoring

Waveform file generation

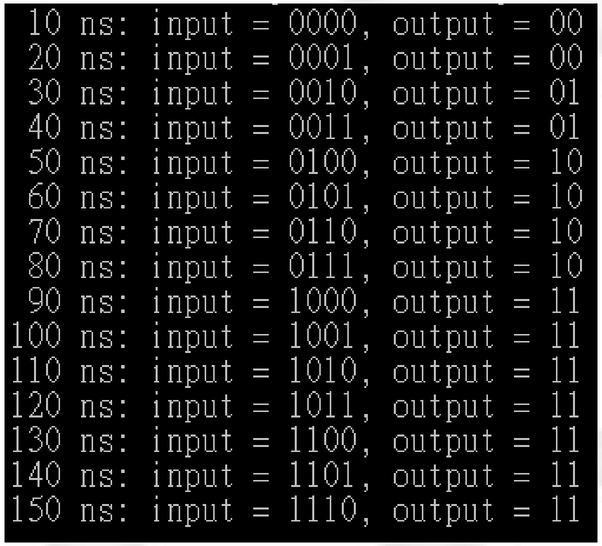
Compile & Simulation (1/3)

- Compile
  - To compile your code, please enter the following command
  - % ncverilog encoder.v
- Simulation
  - To run simulation, please enter the following command
  - % ncverilog encoder_tb.v encoder.v +access+r +define+FSDB

    dump *.fsdb (waveform)*
  - Make sure your code can compile and simulate under the SoC Lab environment.

Compile & Simulation (2/3)

- Terminal view (4-to-2 priority encoder)

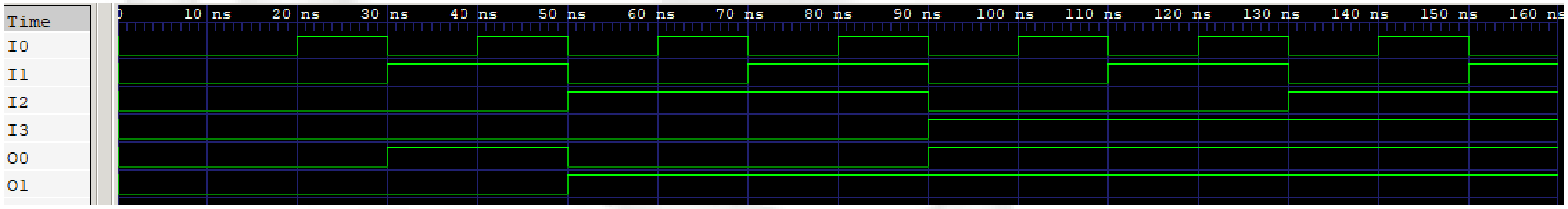


```
 10 ns: input = 0000, output = 00
 20 ns: input = 0001, output = 00
 30 ns: input = 0010, output = 01
 40 ns: input = 0011, output = 01
 50 ns: input = 0100, output = 10
 60 ns: input = 0101, output = 10
 70 ns: input = 0110, output = 10
 80 ns: input = 0111, output = 10
 90 ns: input = 1000, output = 11
100 ns: input = 1001, output = 11
110 ns: input = 1010, output = 11
120 ns: input = 1011, output = 11
130 ns: input = 1100, output = 11
140 ns: input = 1101, output = 11
150 ns: input = 1110, output = 11
```

Debug! →

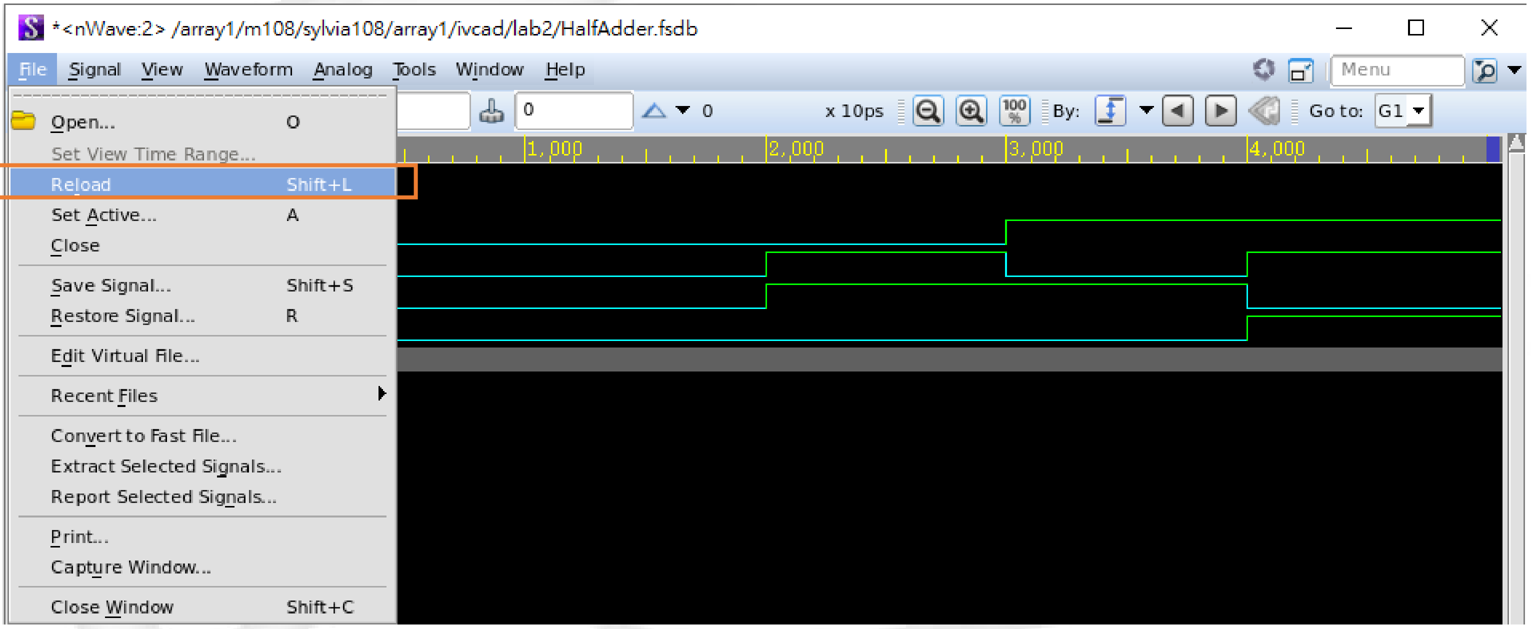
Inputs				Outputs	
I3	I2	I1	I0	O1	O0
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

- Waveform
  - → % nWave &



Compile & Simulation (3/3)

- How to reload waveform



Lab Session 2 (1/2)

- Please complete Lab Session 2 by yourself.
  - Prob A: Go through design steps and implement a 4-to-2 priority encoder
  - Prob B: Implement a full adder with structural coding
  - Prob C: Design a 5-bit add/sub ripple carry adder with hieratical coding
- Due 2020/03/10, Wednesday, 15:00
  - Please compress your homework in tar format and upload it to Moodle.
    - You must follow the file hierarchy for homework submission.
    - Make sure your code can be compiled and simulated under SOC Lab environment.
    - Use %tar -cvf Lab2_StudentID.tar Lab2_StudentID to compress your folder. The compressed file you upload should be named as "Lab2_StudentID.tar" (Ex. Lab2_E24081234.tar)

Lab Session 2 (2/2)

- Remark
  - Warning! Do not submit your homework in the last minute.
  - Warning! Any person found to be dishonest in homework assignments, will receive zero.
- Commands

Problem		Command
ProbA	Compile	% ncverilog encoder.v
	Simulate	% ncverilog encoder_tb.v encoder.v +access+r +define+FSDB
ProbB	Compile	% ncverilog FullAdder.v
	Simulate	% ncverilog FullAdder_tb.v FullAdder.v +access+r +define+FSDB
ProbC	Compile	% ncverilog ripple_adder.v
	Simulate	% ncverilog ripple_adder_tb.v ripple_adder.v +access+r +define+FSDB

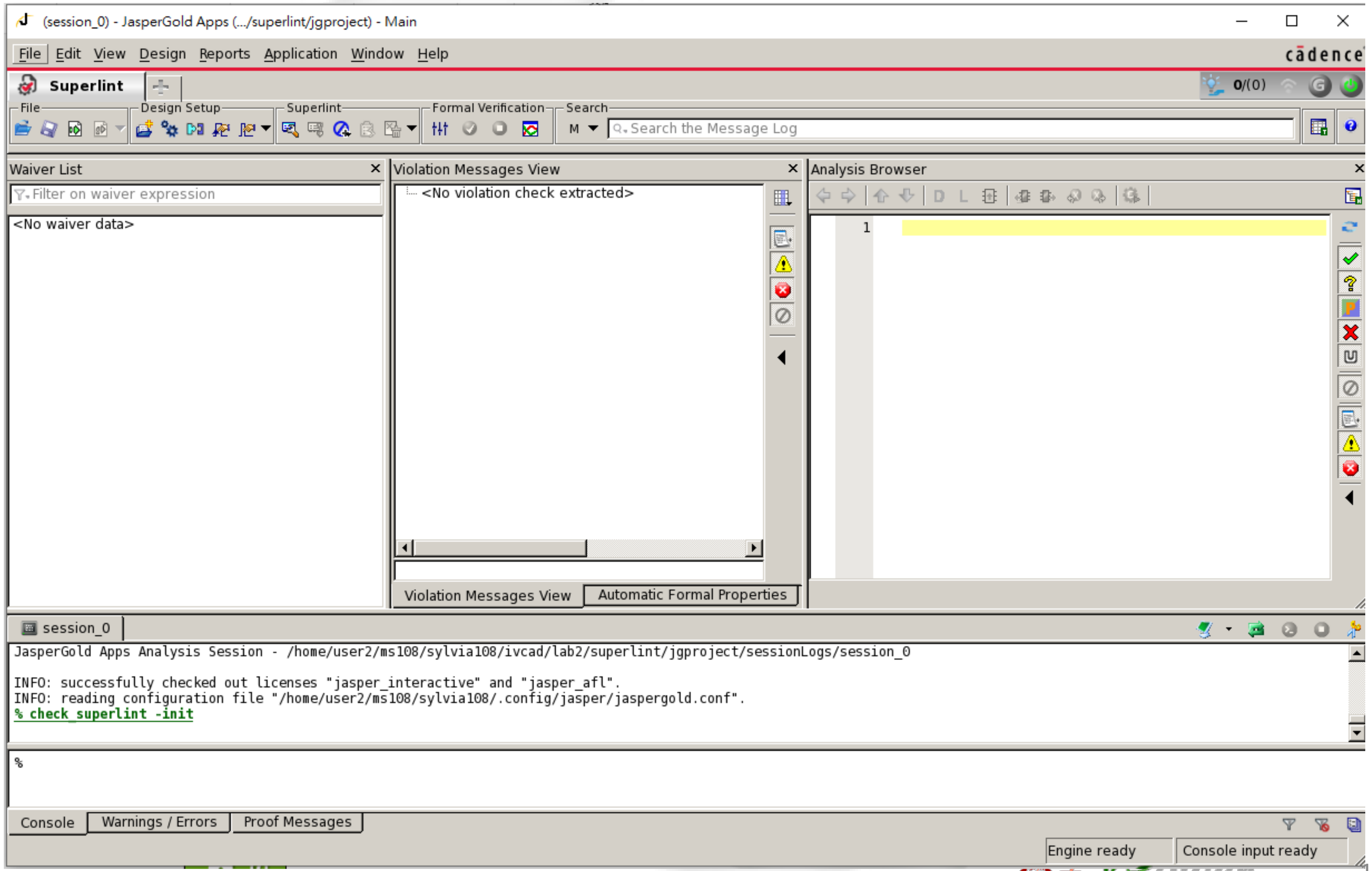
Superlint Tutorial

Target

- Find bugs without requiring specific test patterns
- Some example bugs:
  - Non-synthesizable constructs
  - Unintentional latches
  - Unused declarations
  - Race conditions
  - Out-of-range indexing
- Good coding style is important！

Start Superlint

- >> jg -superlint (or >>jg -superlint superlint.tcl)

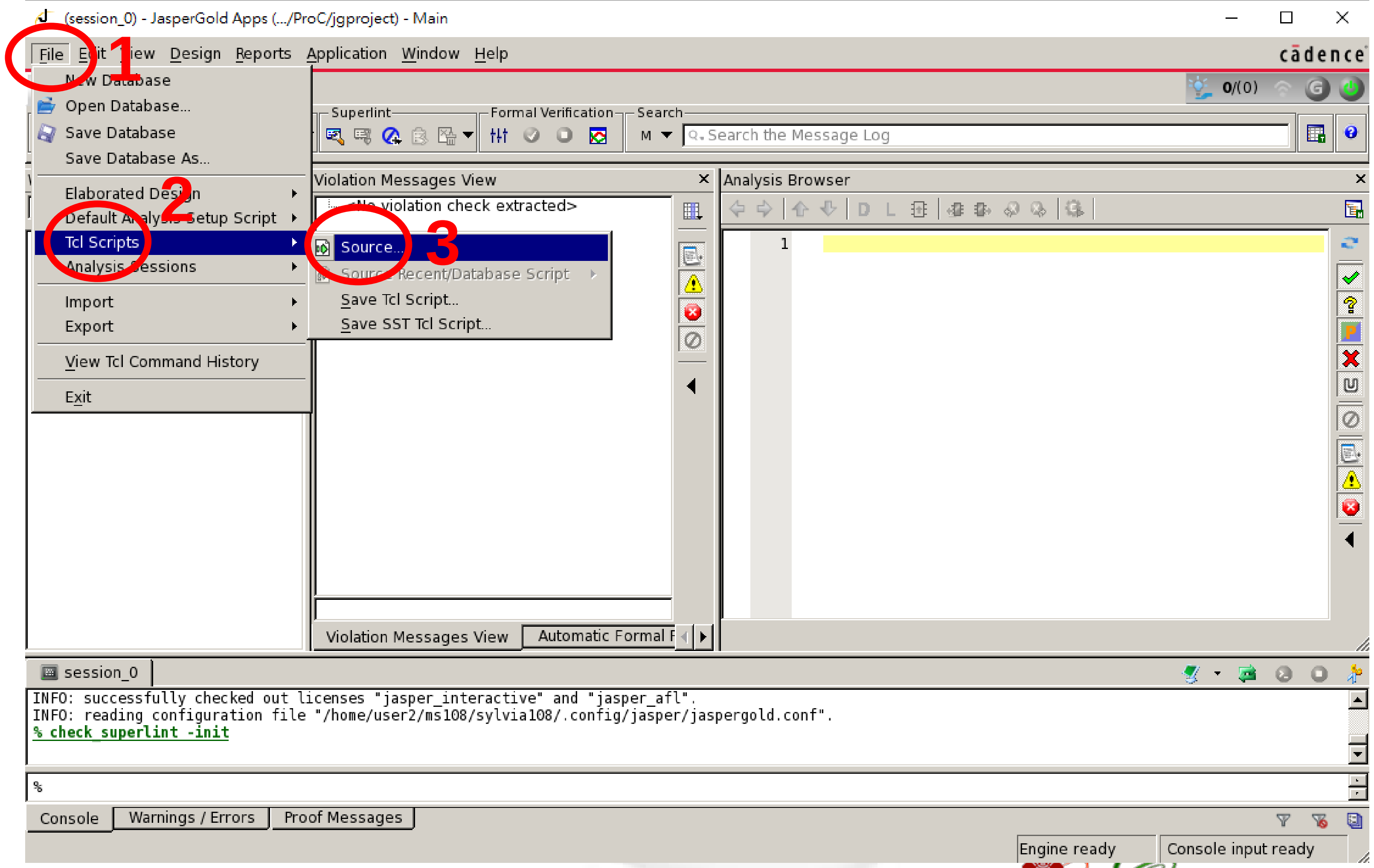


```
JasperGold Apps Analysis Session - /home/user2/ms108/sylvia108/ivcad/lab2/superlint/jgproject/sessionLogs/session_0

INFO: successfully checked out licenses "jasper_interactive" and "jasper_afl".
INFO: reading configuration file "/home/user2/ms108/sylvia108/.config/jasper/jaspergold.conf".
% check_superlint -init
```

```
%
```

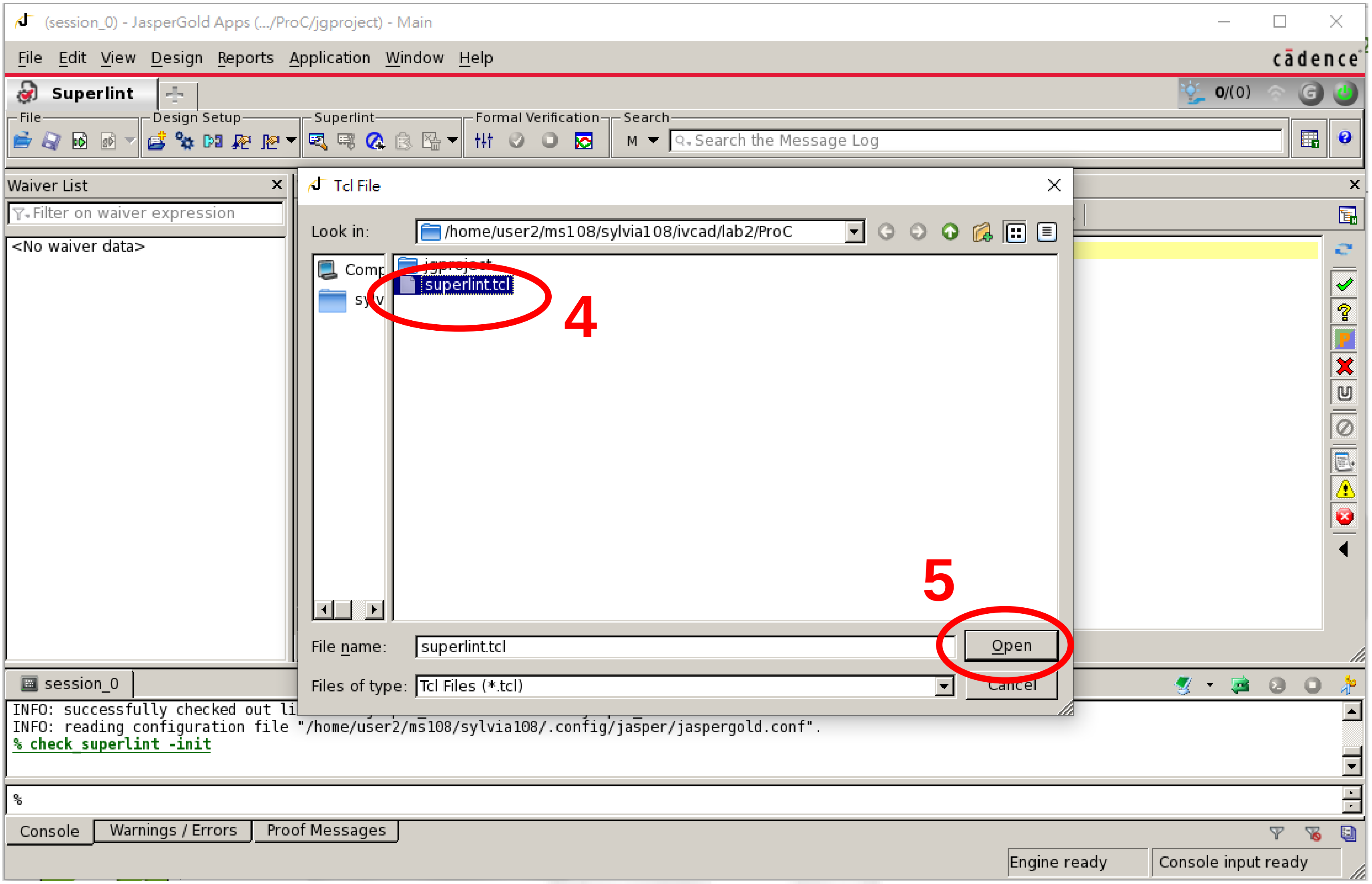
Import tcl file (1/2)

- File -> Tcl Scripts -> Source



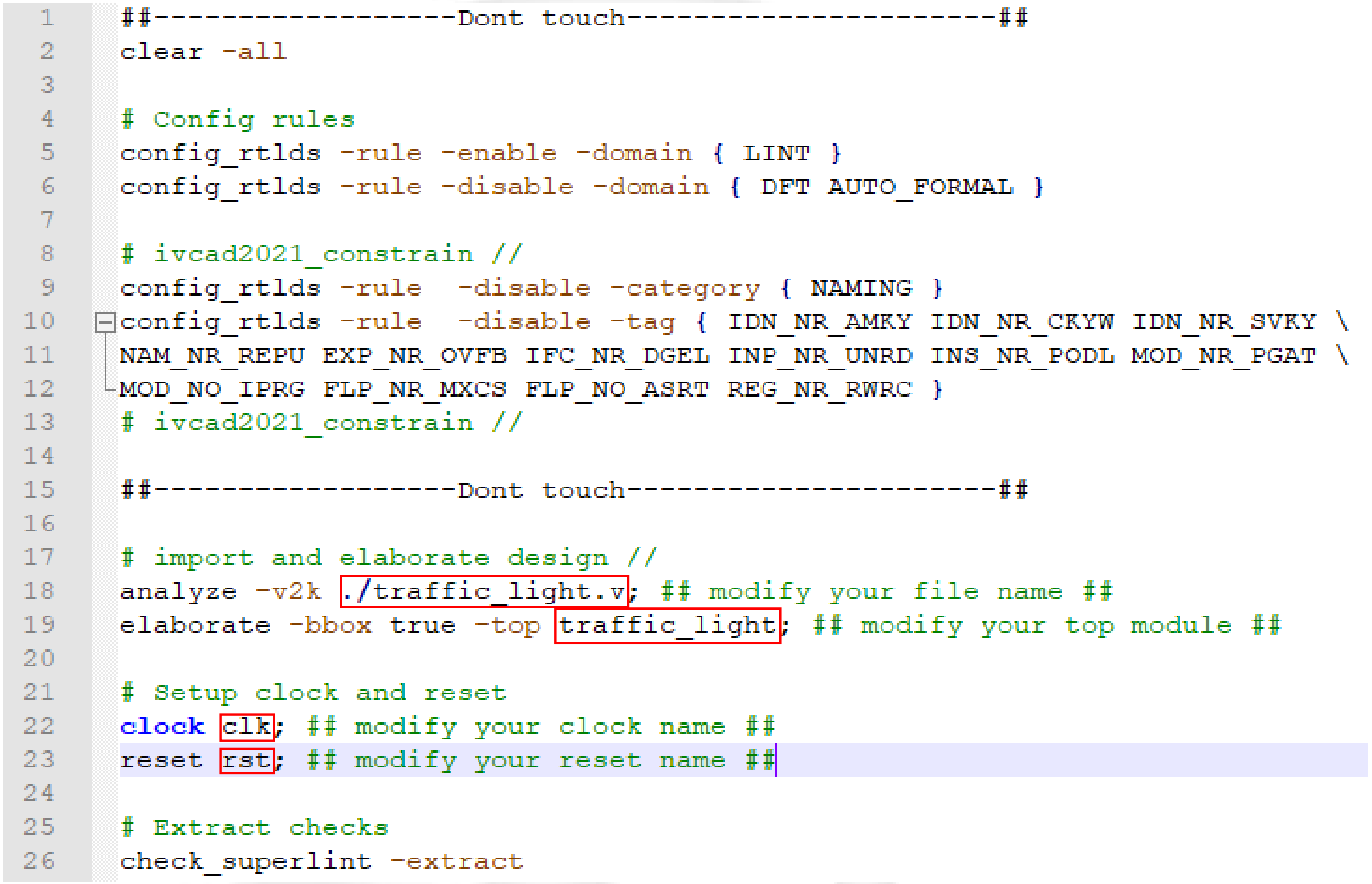
```
INFO: successfully checked out licenses "jasper_interactive" and "jasper_afl".
INFO: reading configuration file "/home/user2/ms108/sylvia108/.config/jasper/jaspergold.conf".
% check_superlint -init
```

Import tcl file (2/2)



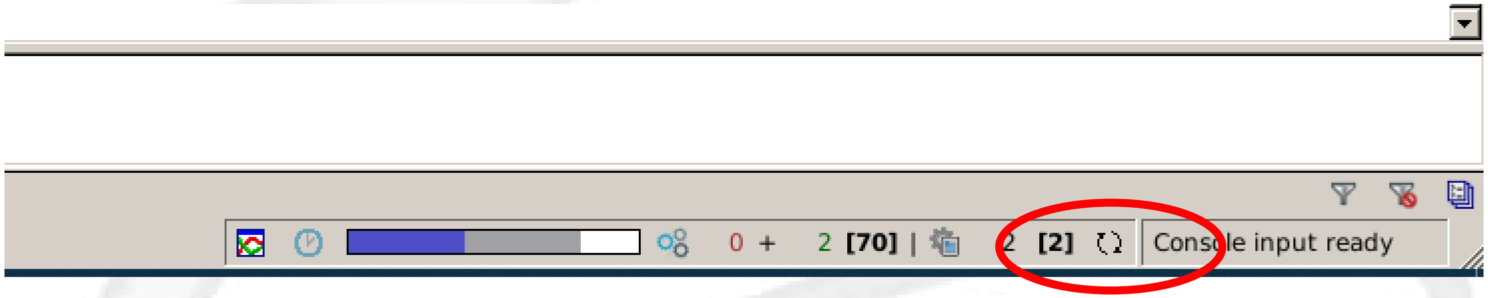
```
INFO: successfully checked out li
INFO: reading configuration file "/home/user2/ms108/sylvia108/.config/jasper/jaspergold.conf".
% check_superlint -init
```

Tcl file

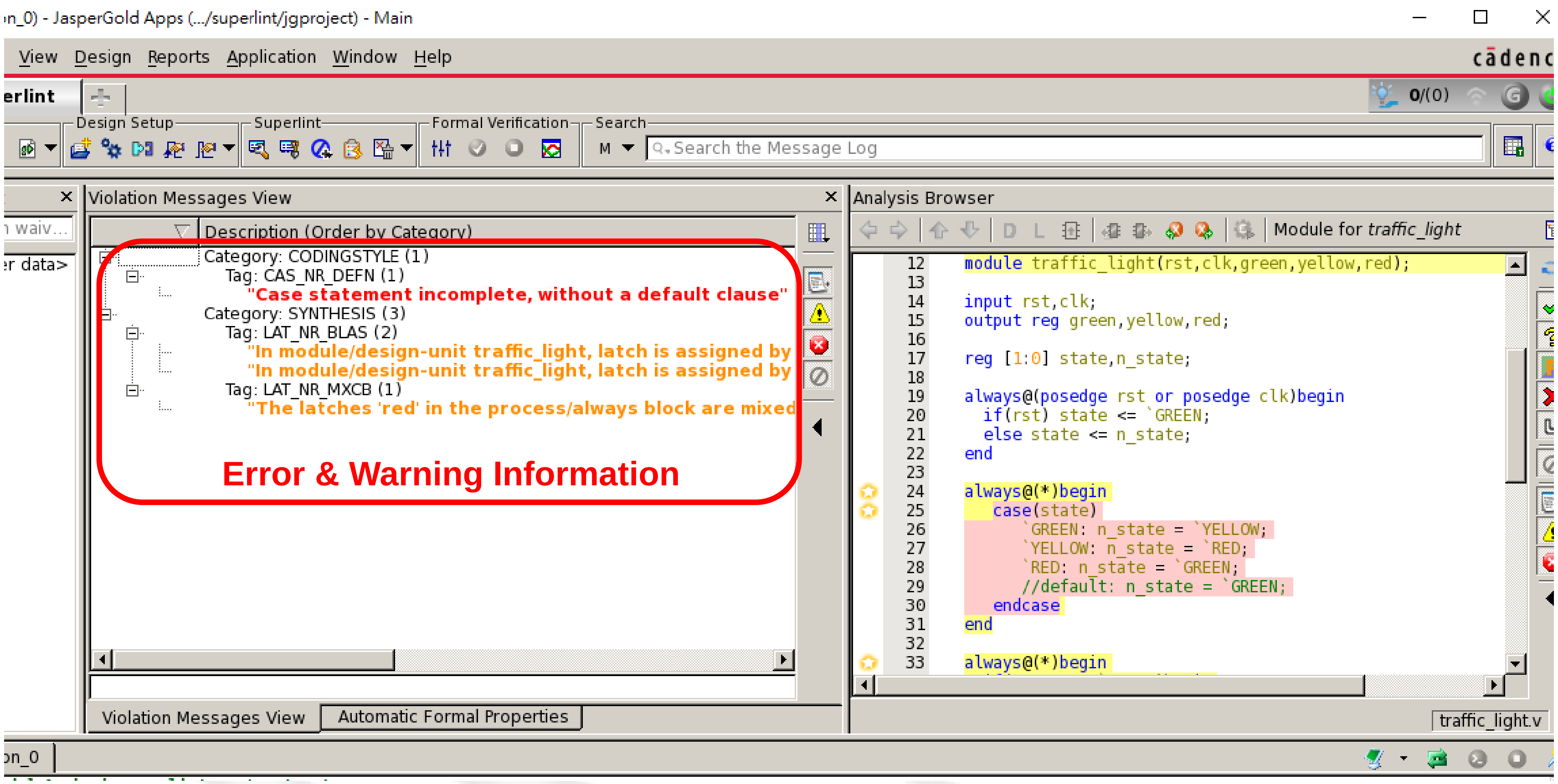


```
##------------------Dont touch----------------------##
clear -all

# Config rules
config_rtlds -rule -enable -domain { LINT }
config_rtlds -rule -disable -domain { DFT AUTO_FORMAL }

# ivcad2021_constrain //
config_rtlds -rule  -disable -category { NAMING }
config_rtlds -rule  -disable -tag { IDN_NR_AMKY IDN_NR_CKYW IDN_NR_SVKY \
NAM_NR_REPU EXP_NR_OVFB IFC_NR_DGEL INP_NR_UNRD INS_NR_PODL MOD_NR_PGAT \
MOD_NO_IPRG FLP_NR_MXCS FLP_NO_ASRT REG_NR_RWRC }
# ivcad2021_constrain //

##------------------Dont touch----------------------##

# import and elaborate design //
analyze -v2k ./traffic_light.v; ## modify your file name ##
elaborate -bbox true -top traffic_light; ## modify your top module ##

# Setup clock and reset
clock clk; ## modify your clock name ##
reset rst; ## modify your reset name ##

# Extract checks
check_superlint -extract
```

Busy state



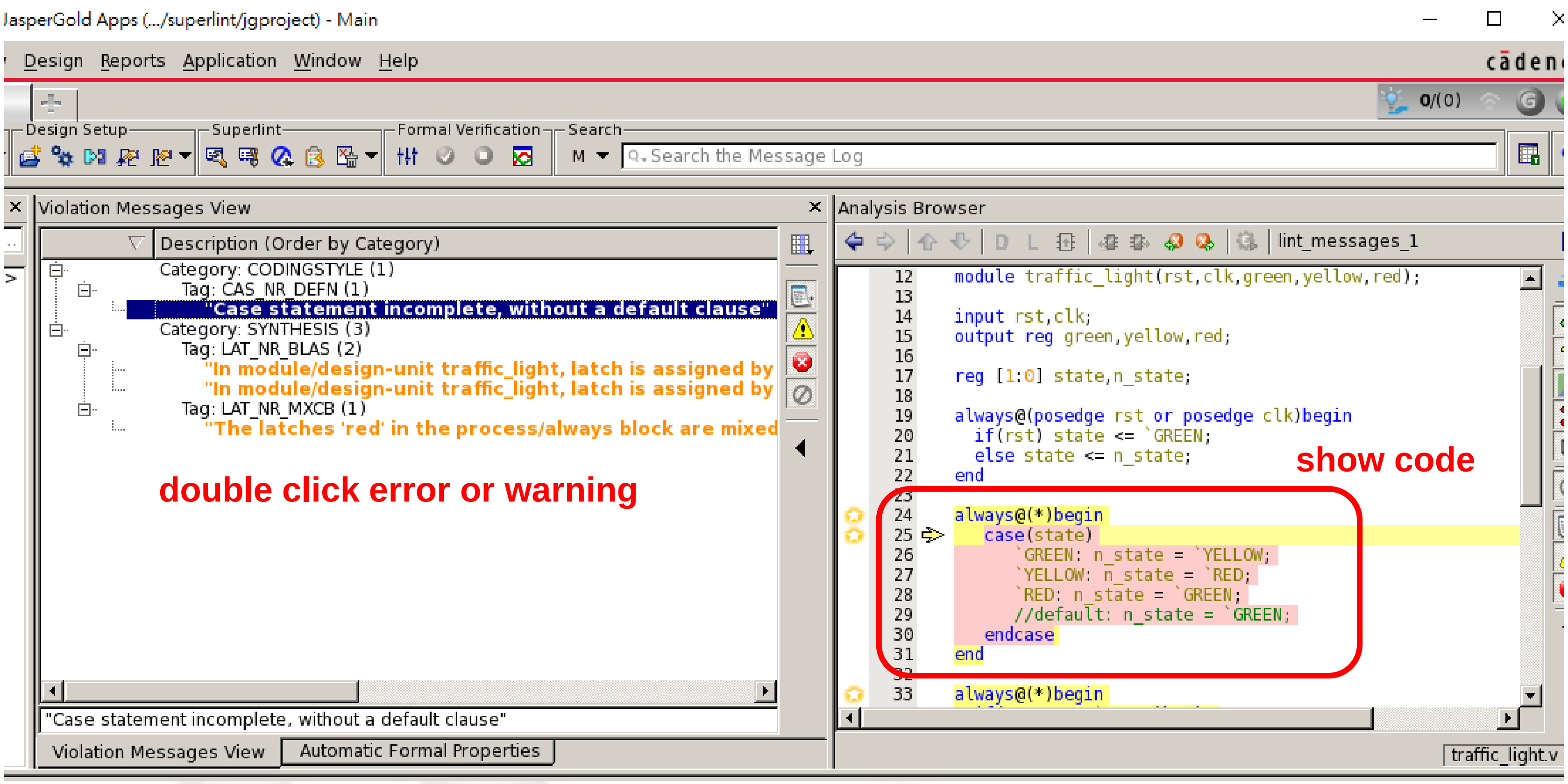
Error & Warning Information(1/3)



```
Description (Order by Category)
Category: CODINGSTYLE (1)
  Tag: CAS_NR_DEFN (1)
    "Case statement incomplete, without a default clause"
Category: SYNTHESIS (3)
  Tag: LAT_NR_BLAS (2)
    "In module/design-unit traffic_light, latch is assigned by
    "In module/design-unit traffic_light, latch is assigned by
  Tag: LAT_NR_MXCB (1)
    "The latches 'red' in the process/always block are mixed
```

```
module traffic_light(rst,clk,green,yellow,red);

input rst,clk;
output reg green,yellow,red;

reg [1:0] state,n_state;

always@(posedge rst or posedge clk)begin
  if(rst) state <= `GREEN;
  else state <= n_state;
end

always@(*)begin
   case(state)
     `GREEN: n_state = `YELLOW;
     `YELLOW: n_state = `RED;
     `RED: n_state = `GREEN;
     //default: n_state = `GREEN;
   endcase
end

always@(*)begin
```

Error & Warning Information(2/3)


JasperGold Apps (.../superlint/jgproject) - Main
Design Reports Application Window Help
Design Setup
Superlint
Formal Verification
Search
Search the Message Log
Violation Messages View
Analysis Browser
lint_messages_1
Description (Order by Category)
Violation Messages View
Automatic Formal Properties
traffic_light.v
double click error or warning
show code

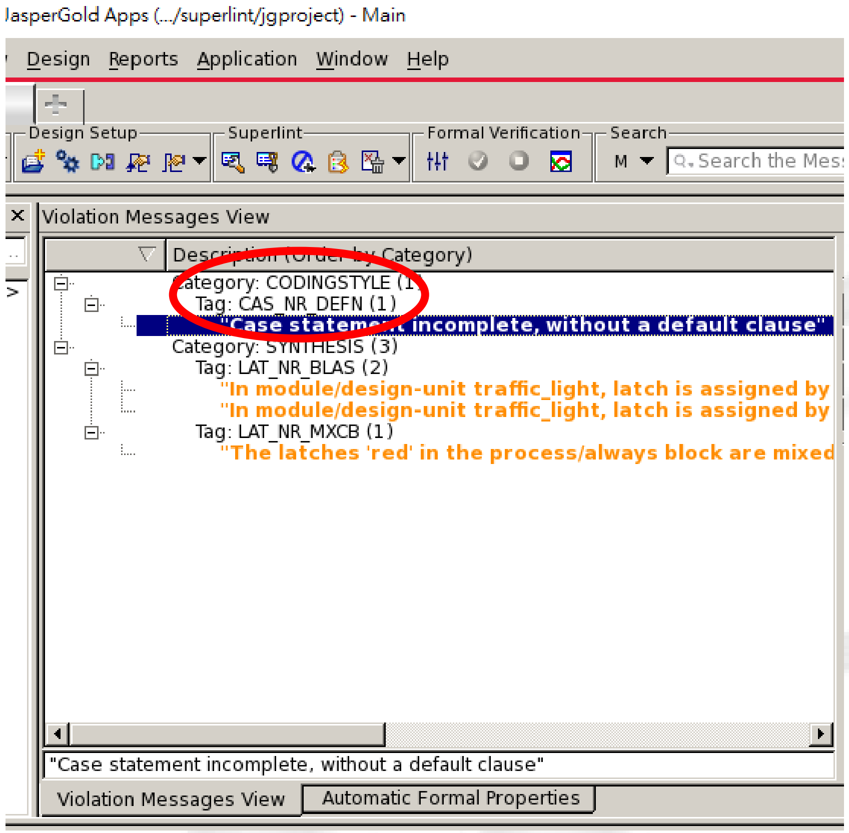
```
Category: CODINGSTYLE (1)
  Tag: CAS_NR_DEFN (1)
    "Case statement incomplete, without a default clause"
Category: SYNTHESIS (3)
  Tag: LAT_NR_BLAS (2)
    "In module/design-unit traffic_light, latch is assigned by
    "In module/design-unit traffic_light, latch is assigned by
  Tag: LAT_NR_MXCB (1)
    "The latches 'red' in the process/always block are mixed
```

"Case statement incomplete, without a default clause"

```
module traffic_light(rst,clk,green,yellow,red);

input rst,clk;
output reg green,yellow,red;

reg [1:0] state,n_state;

always@(posedge rst or posedge clk)begin
  if(rst) state <= `GREEN;
  else state <= n_state;
end

always@(*)begin
   case(state)
      `GREEN: n_state = `YELLOW;
      `YELLOW: n_state = `RED;
      `RED: n_state = `GREEN;
      //default: n_state = `GREEN;
   endcase
end

always@(*)begin
```

Error & Warning Information(3/3)

➢ **Let's check >> jaspergold_superlint_reference.pdf**


JasperGold Apps (.../superlint/jgproject) - Main
Design Reports Application Window Help
Design Setup
Superlint
Formal Verification
Search
Search the Mess
Violation Messages View
Description (Order by Category)
Violation Messages View
Automatic Formal Properties

```
Category: CODINGSTYLE (1)
    Tag: CAS_NR_DEFN (1)
        "Case statement incomplete, without a default clause"
Category: SYNTHESIS (3)
    Tag: LAT_NR_BLAS (2)
        "In module/design-unit traffic_light, latch is assigned by
        "In module/design-unit traffic_light, latch is assigned by
    Tag: LAT_NR_MXCB (1)
        "The latches 'red' in the process/always block are mixed
```

"Case statement incomplete, without a default clause"

CAS_NR_DEFN

Short Message: *Case statement incomplete, %s.*

```
always@(*)begin
   case(state)
      `GREEN: n_state = `YELLOW;
      `YELLOW: n_state = `RED;
      `RED: n_state = `GREEN;
      //default: n_state = `GREEN;
   endcase
end
```

Superlint Coverage

- Count the number of total lines

```
>> wc –l filename
```

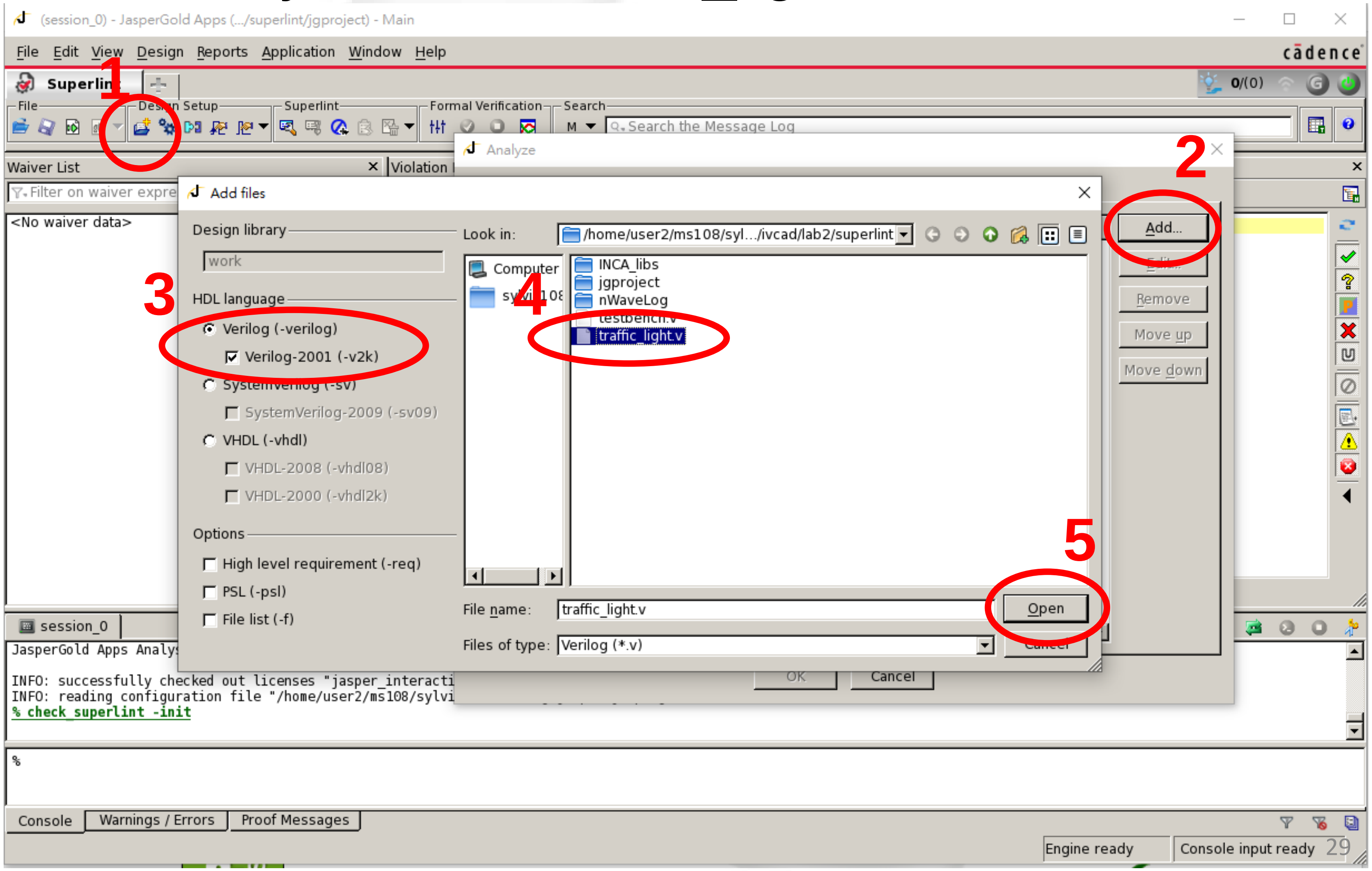
- $\boldsymbol{Coverage = \left(1 - \frac{warning\ lines}{total\ lines}\right) \times 100\%}$

Appendix

GUI of Superlint

Import Design

- >> analyze -v2k ./traffic_light.v



Design library: work

HDL language
- [✓ Verilog (-verilog)]
  - [✓] Verilog-2001 (-v2k)
- SystemVerilog (-sv)
  - SystemVerilog-2009 (-sv09)
- VHDL (-vhdl)
  - VHDL-2008 (-vhdl08)
  - VHDL-2000 (-vhdl2k)

Options
- High level requirement (-req)
- PSL (-psl)
- File list (-f)

Look in: /home/user2/ms108/syl.../ivcad/lab2/superlint

File name: traffic_light.v

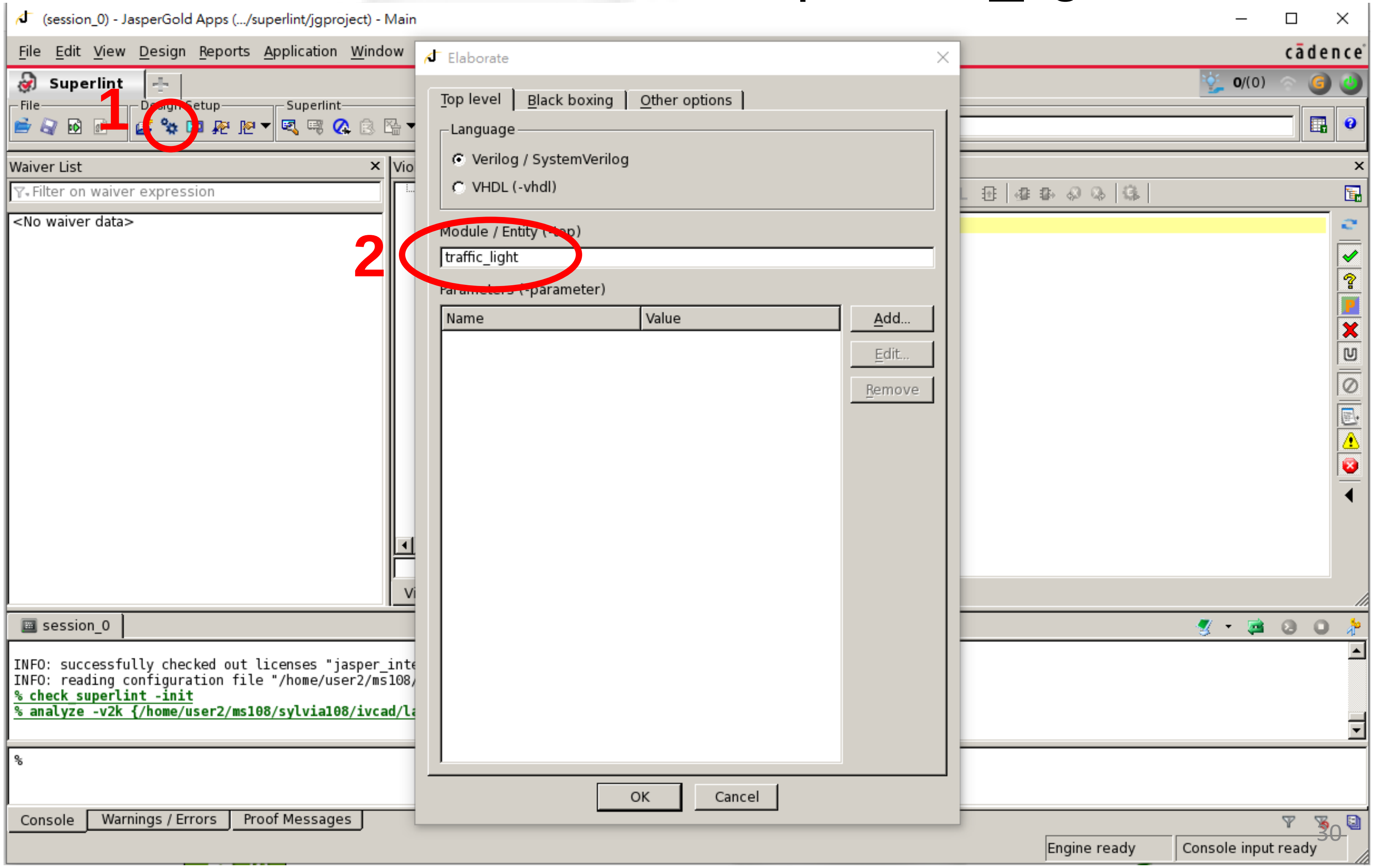
Files of type: Verilog (*.v)

```
JasperGold Apps Analy

INFO: successfully checked out licenses "jasper_interacti
INFO: reading configuration file "/home/user2/ms108/sylvi
% check_superlint -init
```

```
%
```

Elaborate (1/3)

- >> elaborate -bbox true -top traffic_light

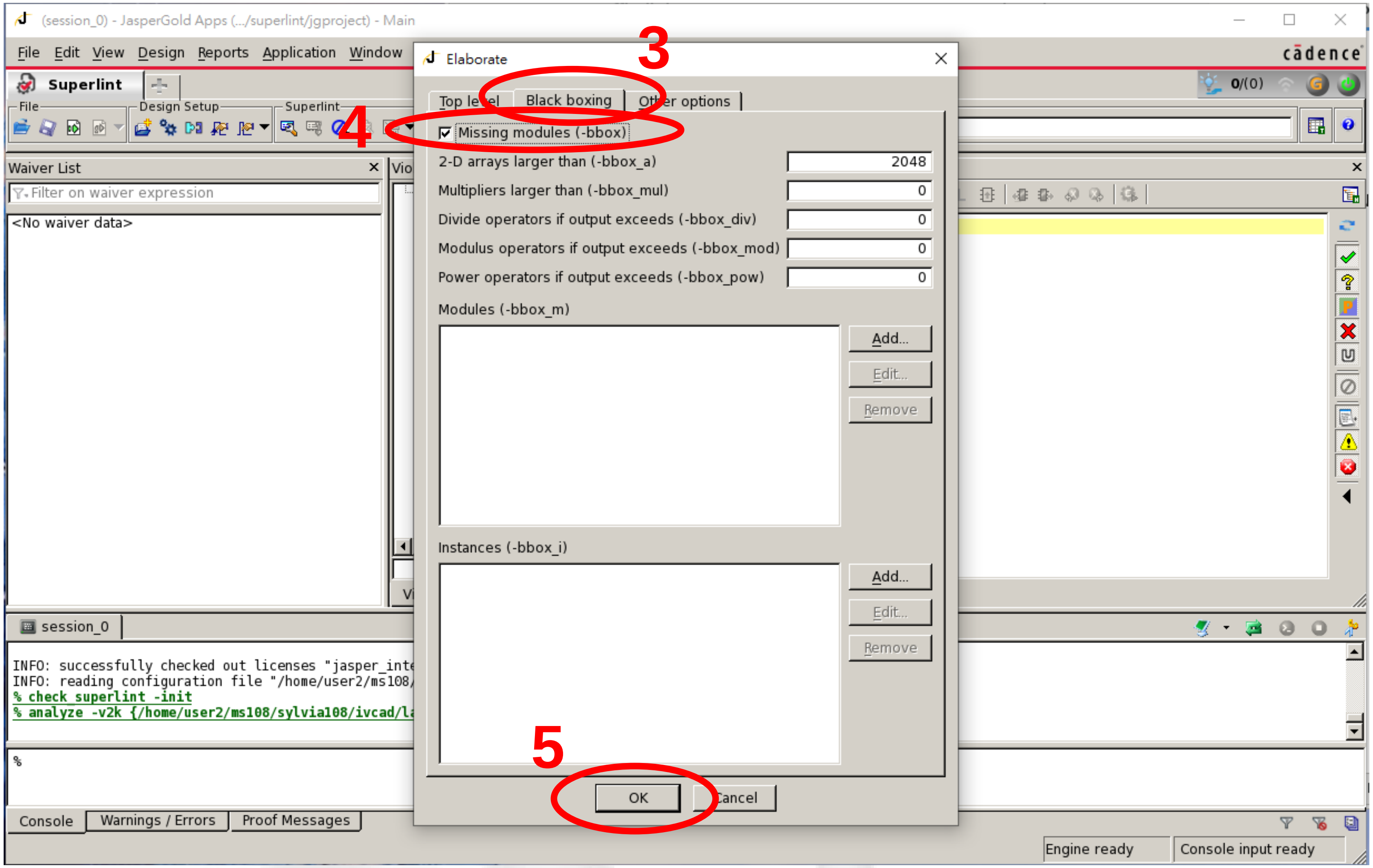


Language: [✓ Verilog / SystemVerilog] VHDL (-vhdl)

Module / Entity (-top): traffic_light

```
INFO: successfully checked out licenses "jasper_inte
INFO: reading configuration file "/home/user2/ms108/
% check_superlint -init
% analyze -v2k {/home/user2/ms108/sylvia108/ivcad/la
```

Elaborate (2/3)

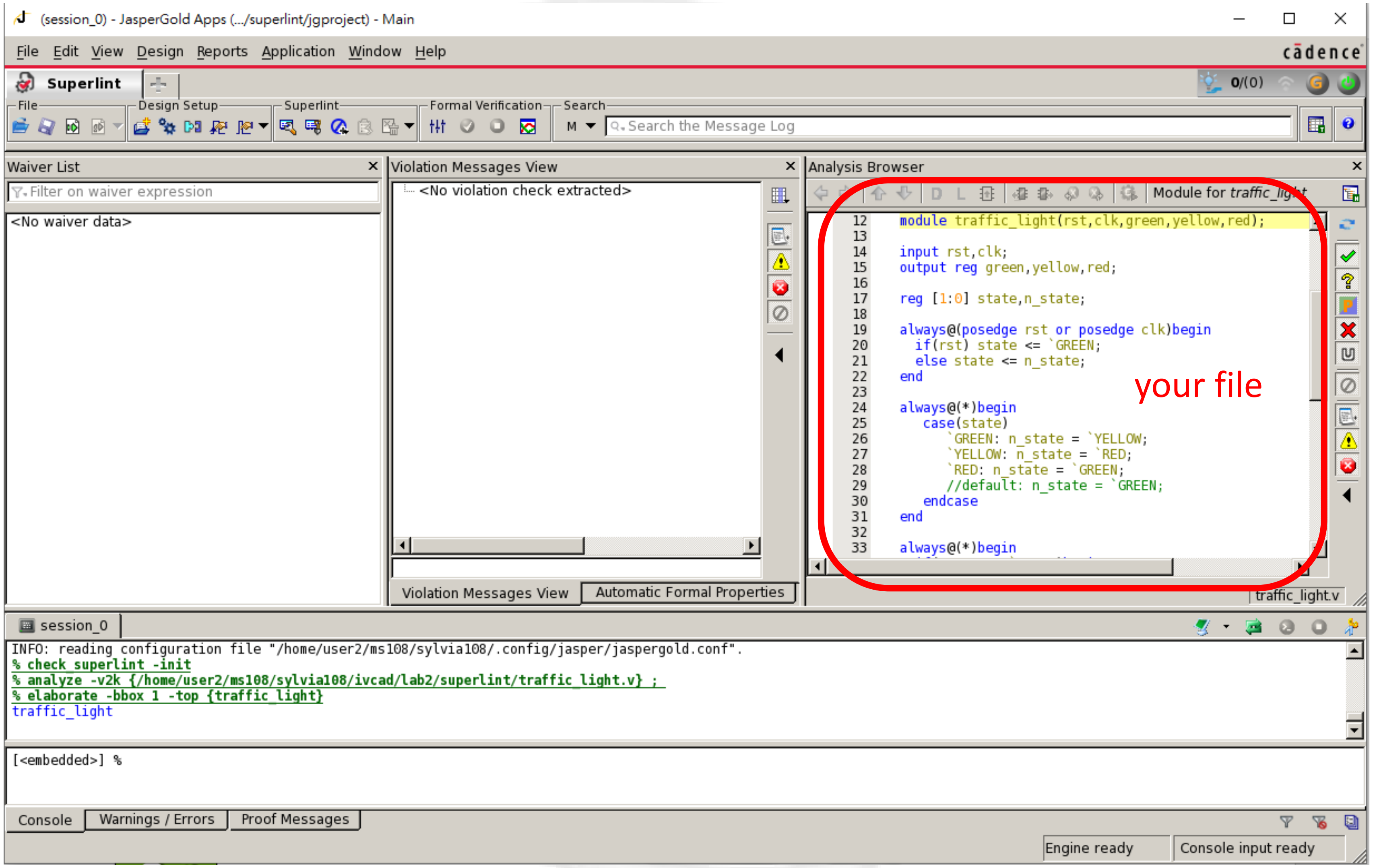


☑ Missing modules (-bbox)

2-D arrays larger than (-bbox_a)	2048
Multipliers larger than (-bbox_mul)	0
Divide operators if output exceeds (-bbox_div)	0
Modulus operators if output exceeds (-bbox_mod)	0
Power operators if output exceeds (-bbox_pow)	0

```
INFO: successfully checked out licenses "jasper_inte
INFO: reading configuration file "/home/user2/ms108/
% check_superlint -init
% analyze -v2k {/home/user2/ms108/sylvia108/ivcad/l
```

Elaborate (3/3)

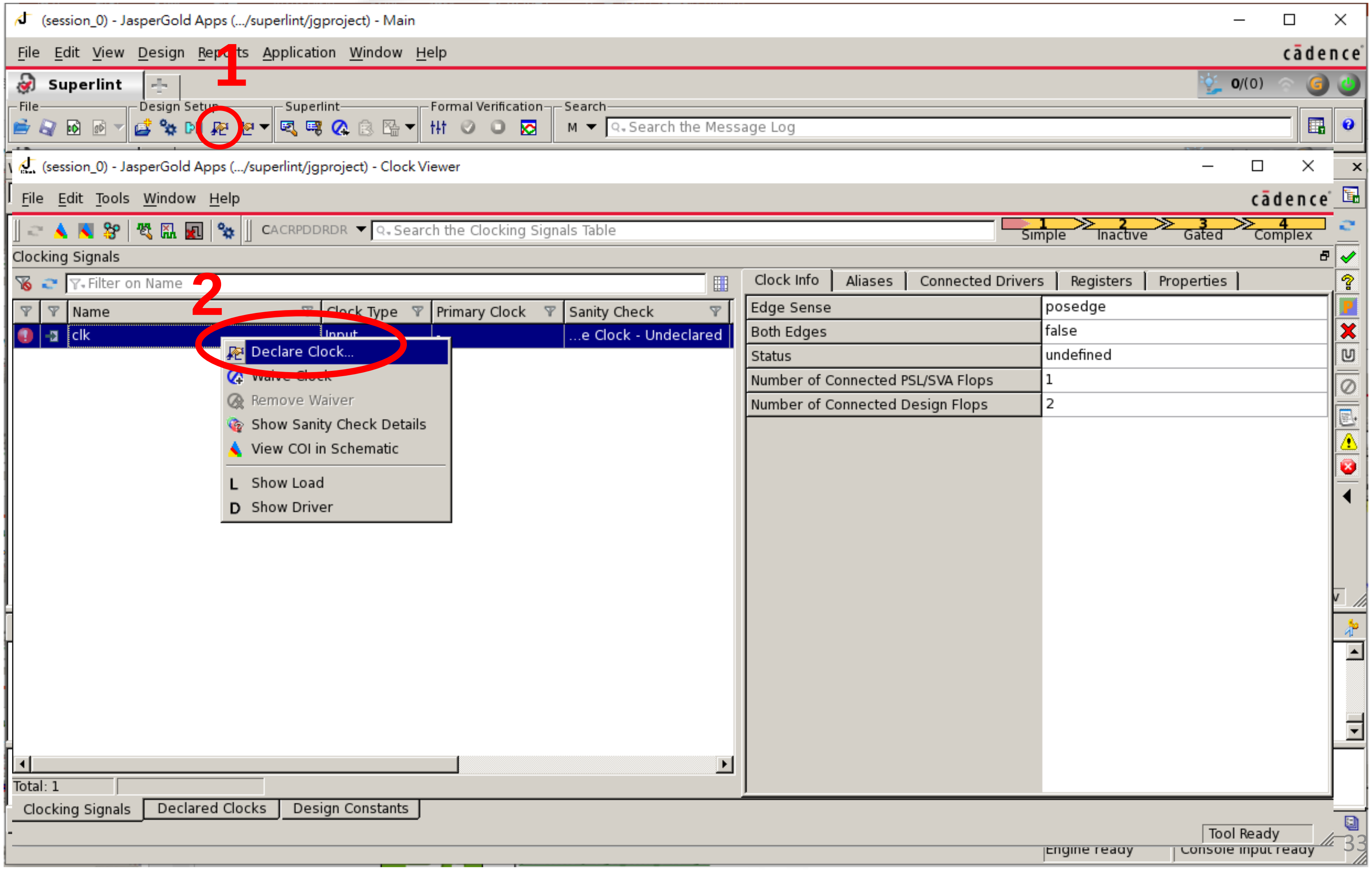


```
module traffic_light(rst,clk,green,yellow,red);

input rst,clk;
output reg green,yellow,red;

reg [1:0] state,n_state;

always@(posedge rst or posedge clk)begin
  if(rst) state <= `GREEN;
  else state <= n_state;
end

always@(*)begin
   case(state)
      `GREEN: n_state = `YELLOW;
      `YELLOW: n_state = `RED;
      `RED: n_state = `GREEN;
      //default: n_state = `GREEN;
   endcase
end

always@(*)begin
```

```
INFO: reading configuration file "/home/user2/ms108/sylvia108/.config/jasper/jaspergold.conf".
% check_superlint -init
% analyze -v2k {/home/user2/ms108/sylvia108/ivcad/lab2/superlint/traffic_light.v} ;
% elaborate -bbox 1 -top {traffic_light}
traffic_light
[<embedded>] %
```

Set Clock (1/3)

- Open clock viewer -> Right click -> Declare Clock

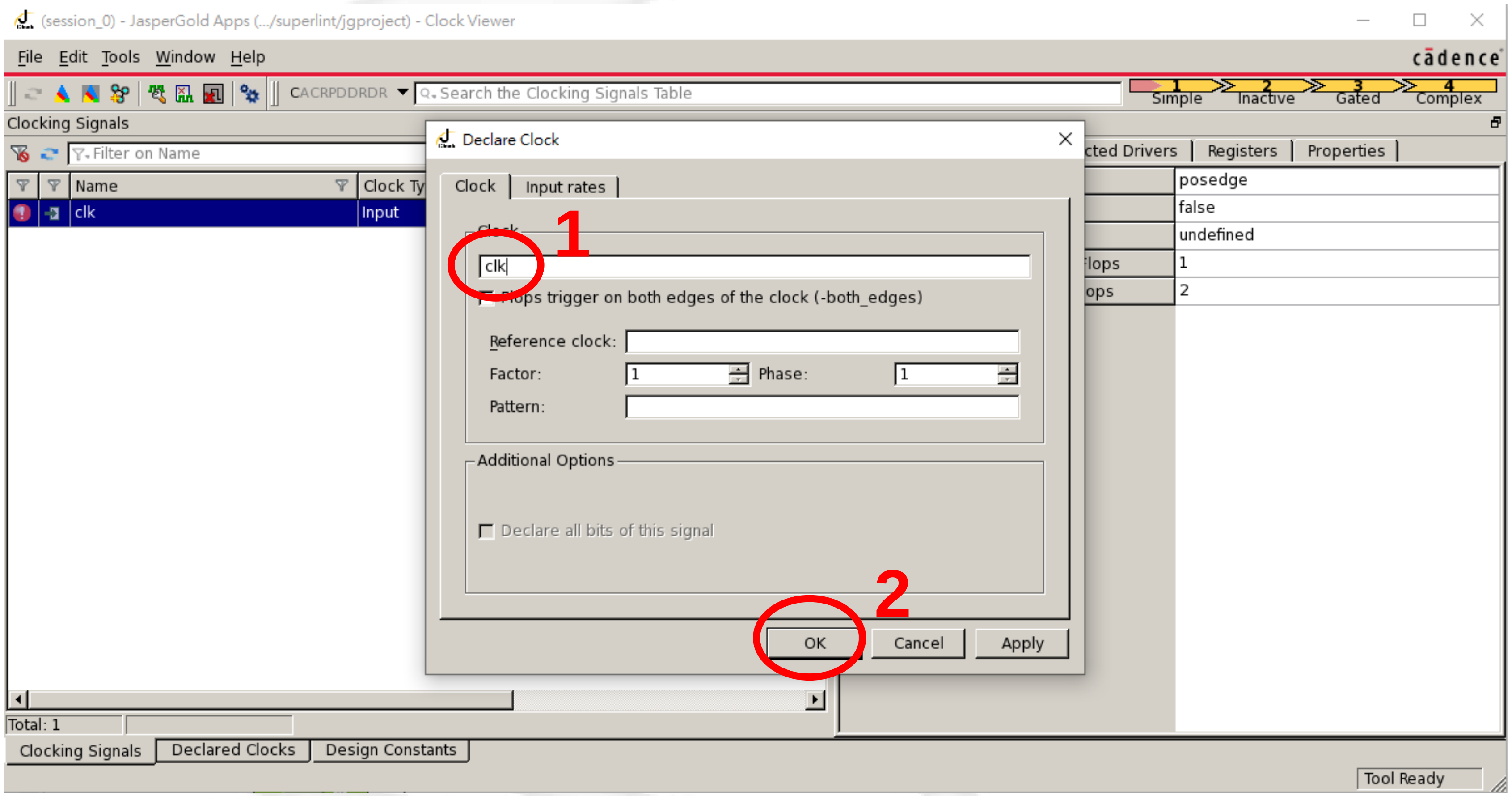

(session_0) - JasperGold Apps (.../superlint/jgproject) - Main
File Edit View Design Reports Application Window Help
1
Superlint
(session_0) - JasperGold Apps (.../superlint/jgproject) - Clock Viewer
File Edit Tools Window Help
Search the Clocking Signals Table
Clocking Signals
2
Declare Clock...
Waive Clock
Remove Waiver
Show Sanity Check Details
View COI in Schematic
Show Load
Show Driver
Clocking Signals
Declared Clocks
Design Constants

Name	Clock Type	Primary Clock	Sanity Check
clk	Input	-	...e Clock - Undeclared

Edge Sense	posedge
Both Edges	false
Status	undefined
Number of Connected PSL/SVA Flops	1
Number of Connected Design Flops	2

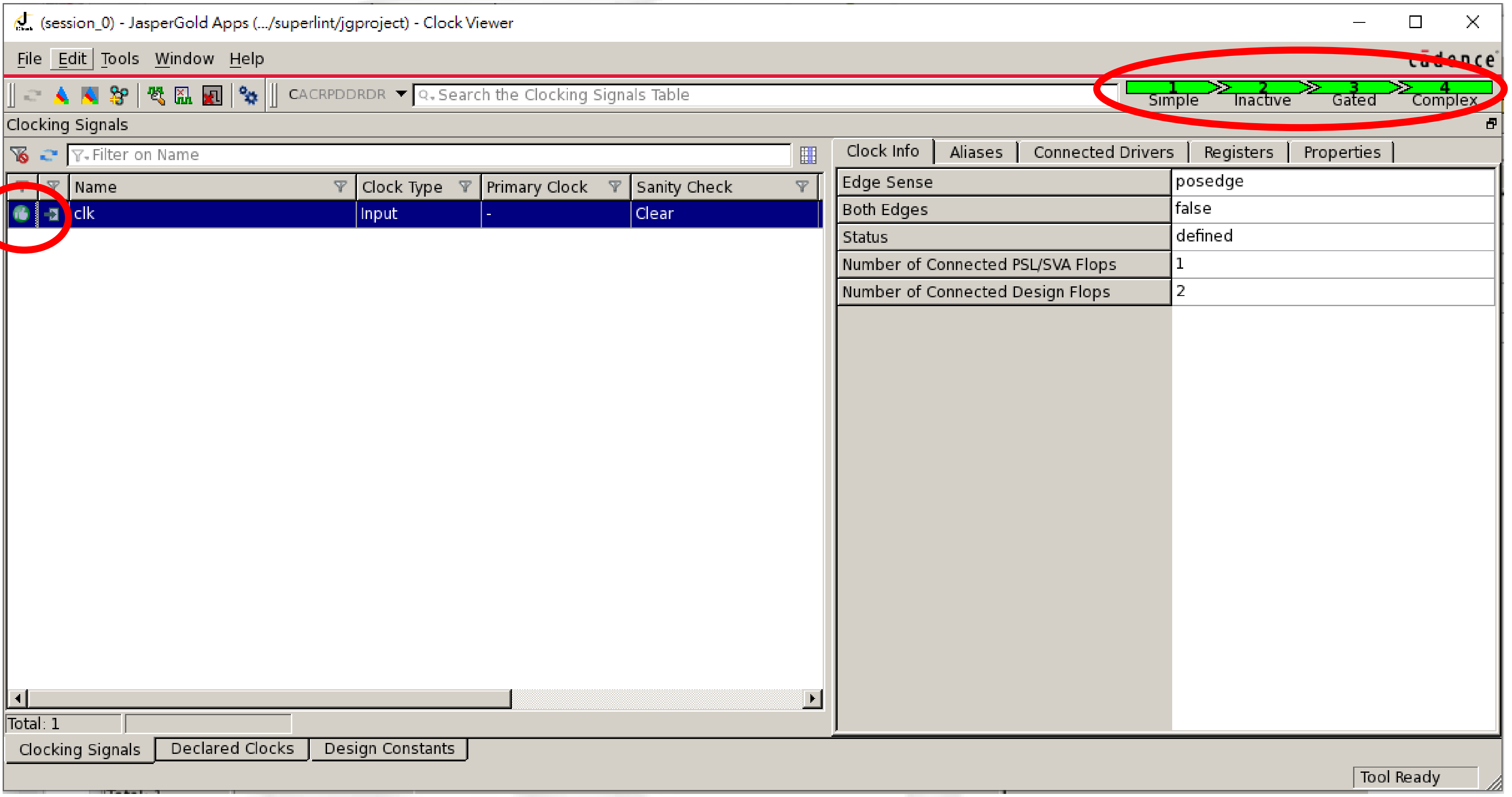
Set Clock (2/3)

- Type your clock name (>> clock clk)



Set Clock (3/3)

- Clock is valid

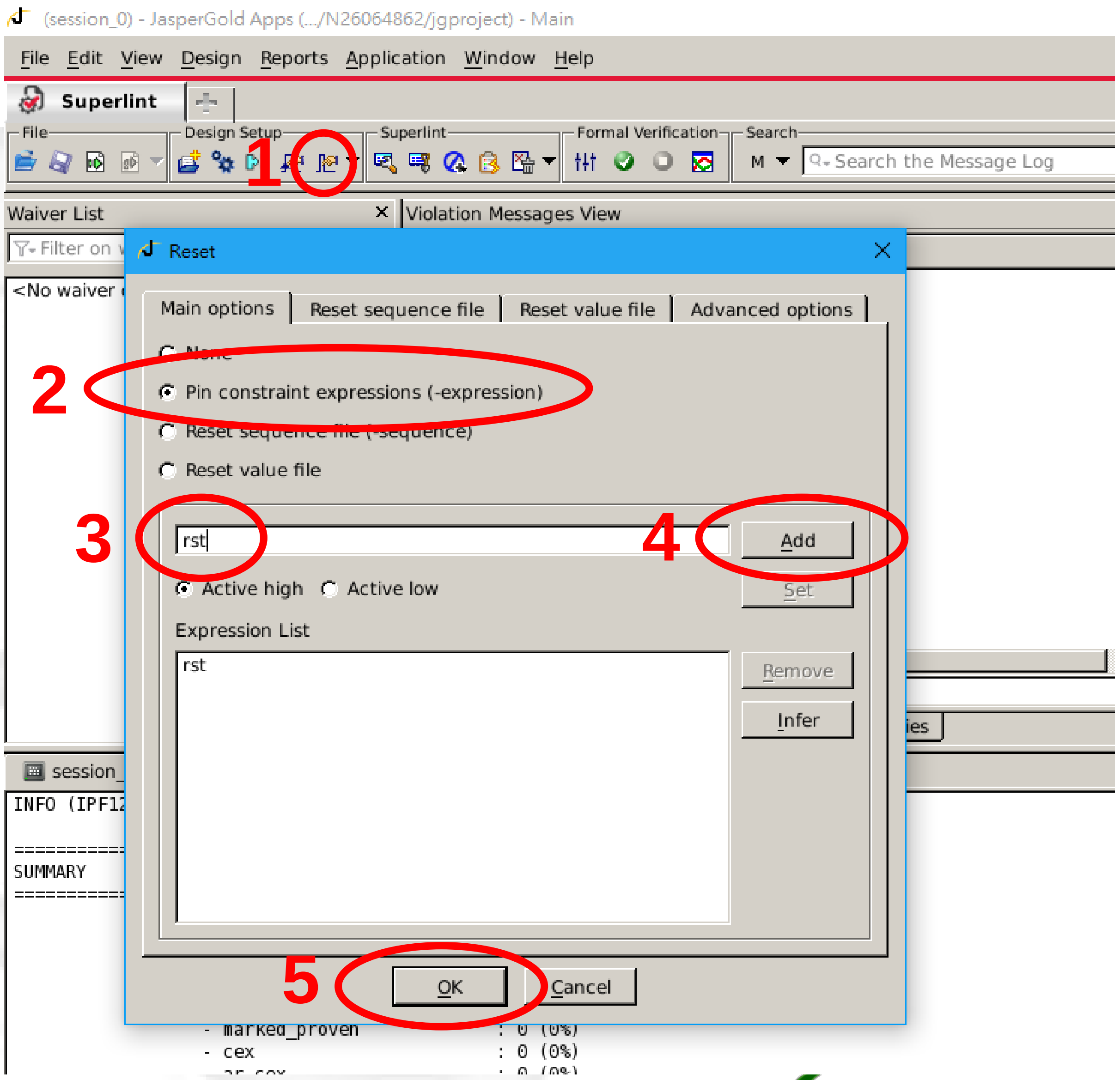


Name	Clock Type	Primary Clock	Sanity Check
clk	Input	-	Clear

Edge Sense	posedge
Both Edges	false
Status	defined
Number of Connected PSL/SVA Flops	1
Number of Connected Design Flops	2

Set Reset

- >> reset rst



- None
- [✓ Pin constraint expressions (-expression)]
- Reset sequence file (-sequence)
- Reset value file

rst

[✓ Active high] Active low

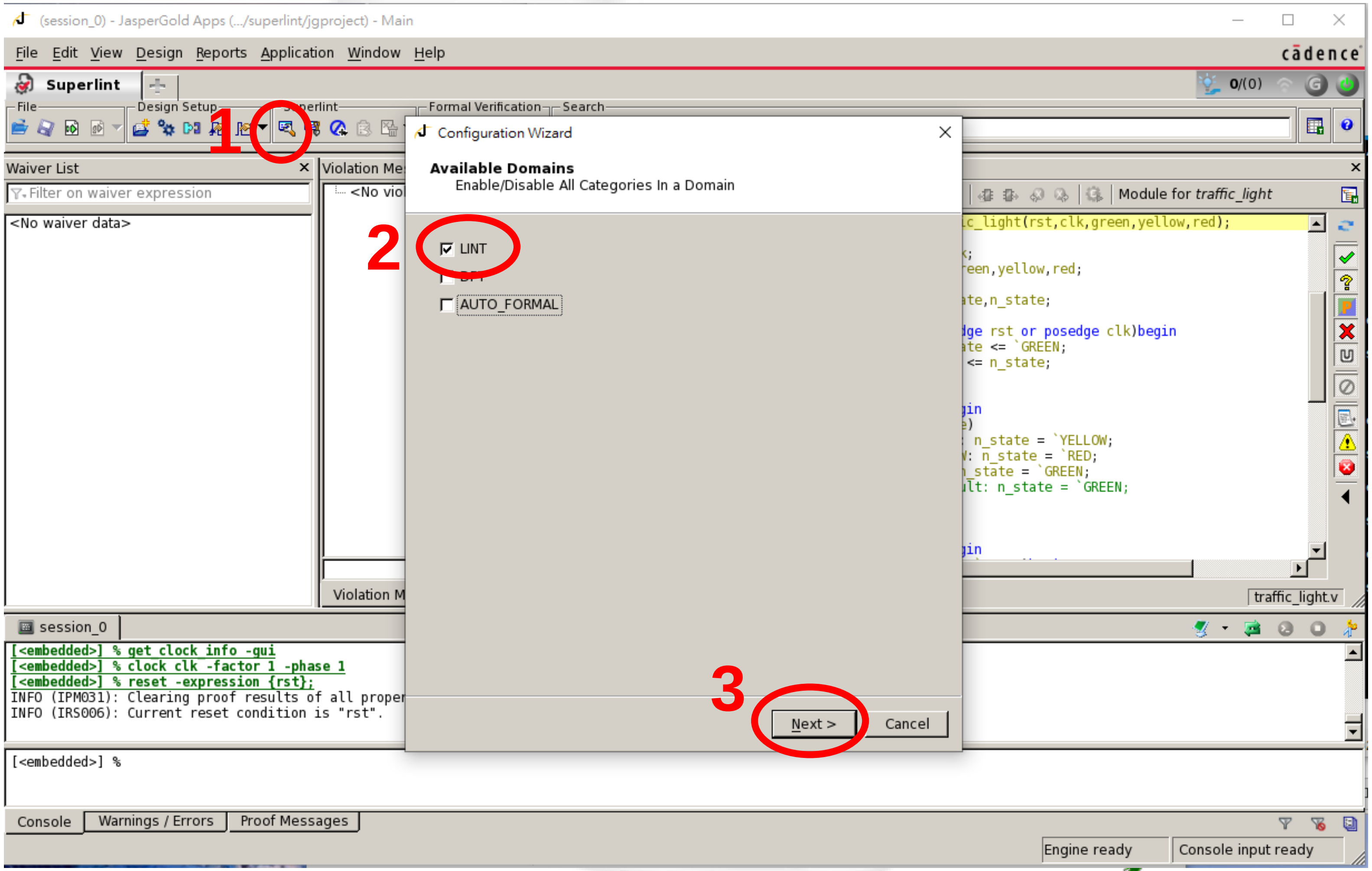
Expression List

rst

```
INFO (IPF12
==========
SUMMARY
==========
- marked_proven          : 0 (0%)
- cex                    : 0 (0%)
- ar_cex                 : 0 (0%)
```

Set Rule (1/2)

- Configure Superlint checks -> LINT



[✓ LINT]
AUTO_FORMAL

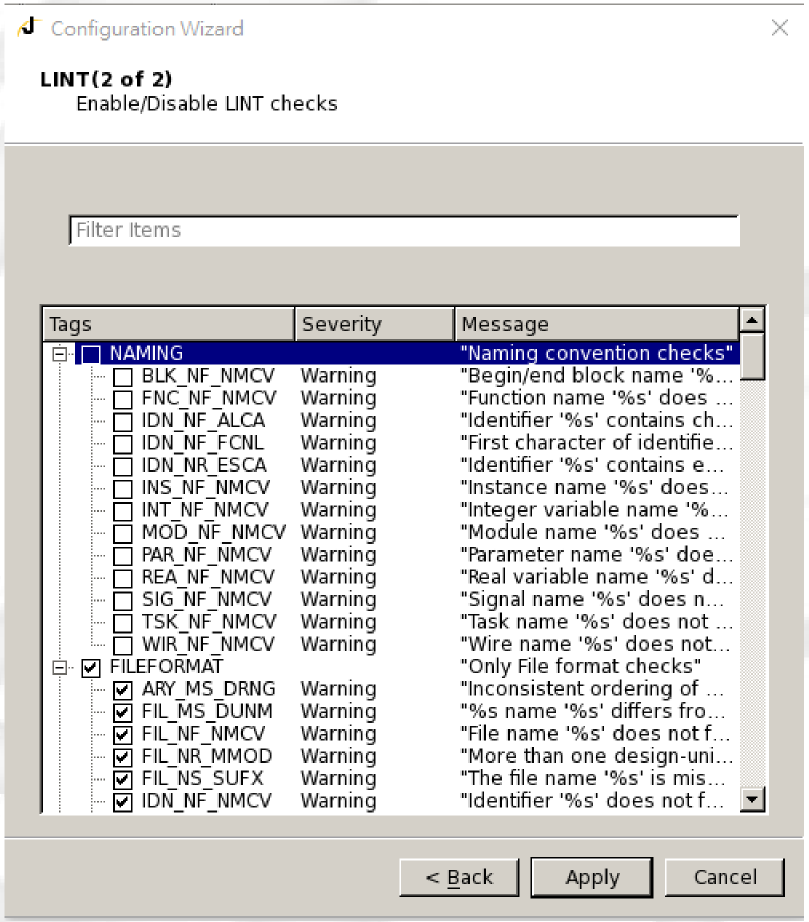
```
c_light(rst,clk,green,yellow,red);
k;
een,yellow,red;
ate,n_state;
dge rst or posedge clk)begin
ate <= `GREEN;
 <= n_state;
gin
e)
 n_state = `YELLOW;
W: n_state = `RED;
 _state = `GREEN;
ult: n_state = `GREEN;
gin
```

```
[<embedded>] % get_clock_info -gui
[<embedded>] % clock clk -factor 1 -phase 1
[<embedded>] % reset -expression {rst};
INFO (IPM031): Clearing proof results of all proper
INFO (IRS006): Current reset condition is "rst".
```

```
[<embedded>] %
```

Set Rule (2/2)

- Choose the rules you want to check



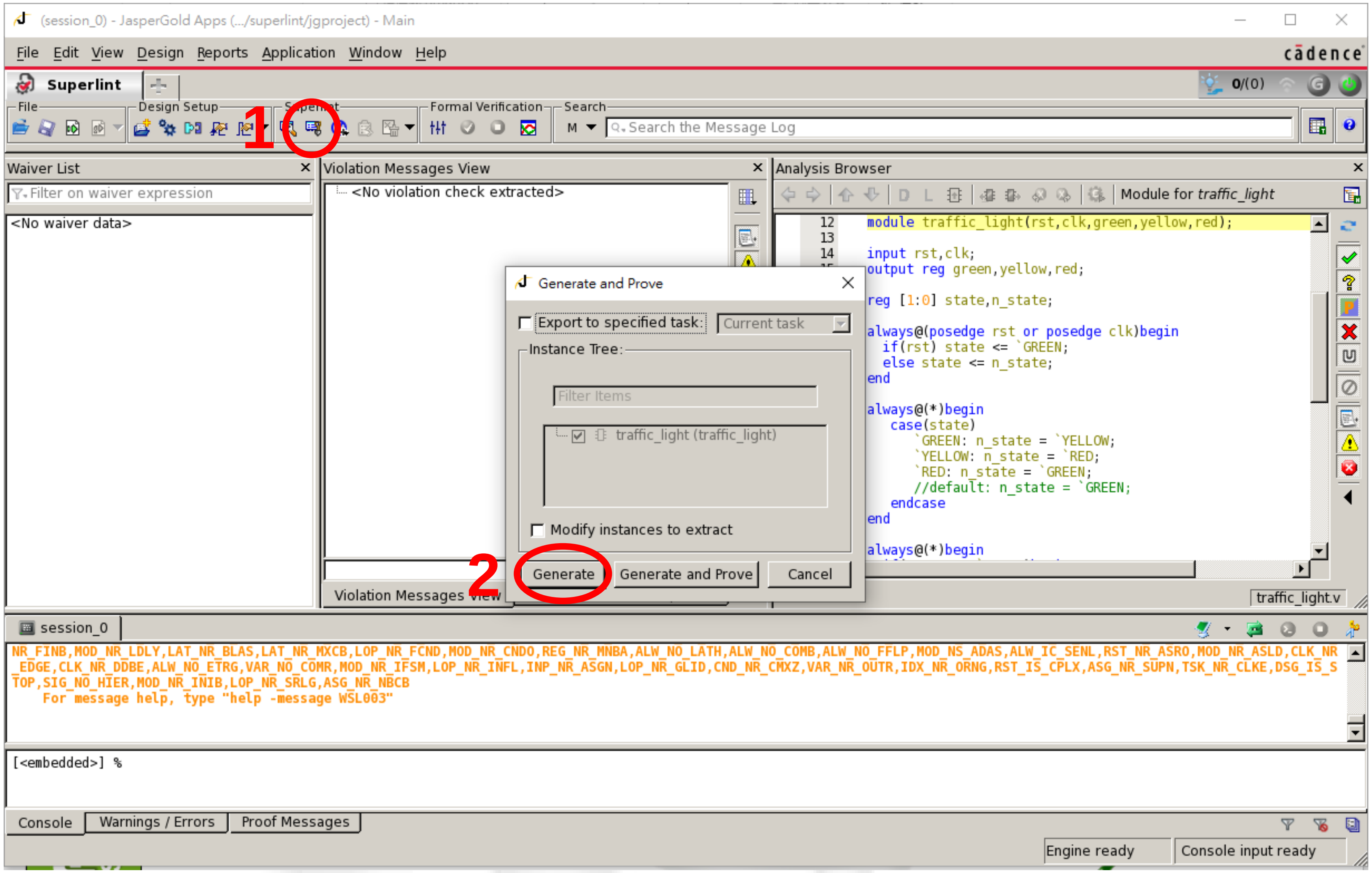
Configuration Wizard

LINT(2 of 2)
Enable/Disable LINT checks

Filter Items

Tags	Severity	Message
☐ NAMING		"Naming convention checks"
☐ BLK_NF_NMCV	Warning	"Begin/end block name '%…
☐ FNC_NF_NMCV	Warning	"Function name '%s' does …
☐ IDN_NF_ALCA	Warning	"Identifier '%s' contains ch…
☐ IDN_NF_FCNL	Warning	"First character of identifie…
☐ IDN_NR_ESCA	Warning	"Identifier '%s' contains e…
☐ INS_NF_NMCV	Warning	"Instance name '%s' does…
☐ INT_NF_NMCV	Warning	"Integer variable name '%…
☐ MOD_NF_NMCV	Warning	"Module name '%s' does …
☐ PAR_NF_NMCV	Warning	"Parameter name '%s' doe…
☐ REA_NF_NMCV	Warning	"Real variable name '%s' d…
☐ SIG_NF_NMCV	Warning	"Signal name '%s' does n…
☐ TSK_NF_NMCV	Warning	"Task name '%s' does not …
☐ WIR_NF_NMCV	Warning	"Wire name '%s' does not…
✓ FILEFORMAT		"Only File format checks"
✓ ARY_MS_DRNG	Warning	"Inconsistent ordering of …
✓ FIL_MS_DUNM	Warning	"%s name '%s' differs fro…
✓ FIL_NF_NMCV	Warning	"File name '%s' does not f…
✓ FIL_NR_MMOD	Warning	"More than one design-uni…
✓ FIL_NS_SUFX	Warning	"The file name '%s' is mis…
✓ IDN_NF_NMCV	Warning	"Identifier '%s' does not f…

Generate Superlint checks



```
module traffic_light(rst,clk,green,yellow,red);

input rst,clk;
output reg green,yellow,red;

reg [1:0] state,n_state;

always@(posedge rst or posedge clk)begin
  if(rst) state <= `GREEN;
  else state <= n_state;
end

always@(*)begin
   case(state)
      `GREEN: n_state = `YELLOW;
      `YELLOW: n_state = `RED;
      `RED: n_state = `GREEN;
      //default: n_state = `GREEN;
   endcase
end

always@(*)begin
```

```
NR_FINB,MOD_NR_LDLY,LAT_NR_BLAS,LAT_NR_MXCB,LOP_NR_FCND,MOD_NR_CNDO,REG_NR_MNBA,ALW_NO_LATH,ALW_NO_COMB,ALW_NO_FFLP,MOD_NS_ADAS,ALW_IC_SENL,RST_NR_ASRO,MOD_NR_ASLD,CLK_NR
_EDGE,CLK_NR_DDBE,ALW_NO_ETRG,VAR_NO_COMR,MOD_NR_IFSM,LOP_NR_INFL,INP_NR_ASGN,LOP_NR_GLID,CND_NR_CMXZ,VAR_NR_OUTR,IDX_NR_ORNG,RST_IS_CPLX,ASG_NR_SUPN,TSK_NR_CLKE,DSG_IS_S
TOP,SIG_NO_HIER,MOD_NR_INIB,LOP_NR_SRLG,ASG_NR_NBCB
    For message help, type "help -message WSL003"

[<embedded>] %
```