Switching Circuit
& Logic Design

Basic Logic Design Practicing - Quartus Schematic to Verilog

Switching Circuit
& Logic Design

Introduction to Gate Level Verilog HDL
--Convert Quartus II Schematic to Verilog

Lecturer: TA 謝明倫 (BL-421)
yans@media.ee.ntu.edu.tw

交換電路與邏輯設計課程 四班共同教學
Professors: 吳安宇 簡韶逸 盧奕璋 江介宏

Outline

- Introduction to Verilog HDL
  - Levels of abstraction
  - An example
  - Verilog design flow
- Trial: verilog sim
  1. .v design from .bdf
  2. .v testbench from .vwf
  3. Compile & simulate
  4. Verify & debug
- Summary
- Q&A and Ref.

For Combinational & Gate-level only

Verilog HDL - Comb. Gate Level design

Switching Circuit & Logic Design

What is Verilog HDL?

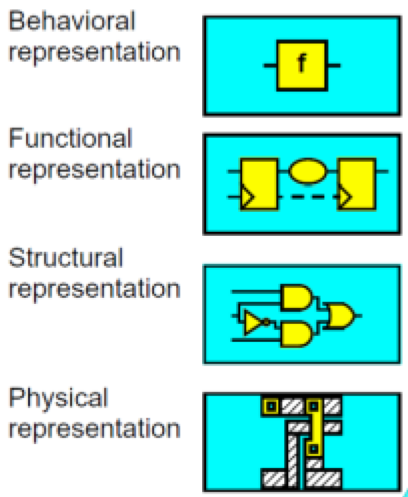
- Key features of Verilog
  - **Multiple levels of abstraction**
    - Behavioral
    - Functional (RTL:Register Transfer Level)
    - Structural (Gate-Level)
  - Model the timing of the system
  - Express the concurrency
  - Verify the design

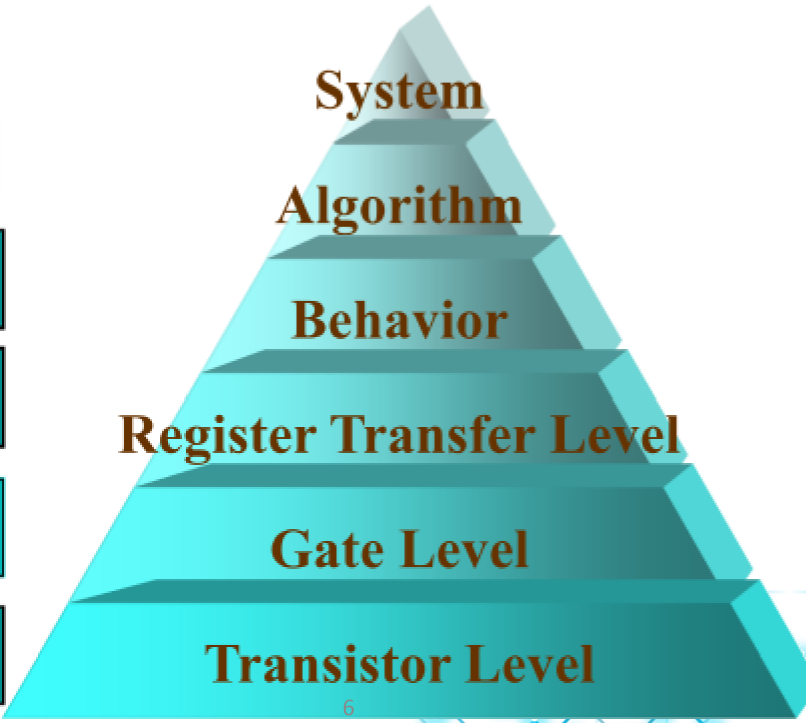
- Why not C++ ?
  - Not natively used for ckt design
  - Natively an imperative programming language
  - Ckt is parallel anytime anywhere
    - Every time every signals change their values, we need to trace them & interact with each other
- But be careful that our computer is still imperative
  - => All the parallel performance is "simulated"
  - Verilog is a language easy to "design HW" & "simulate it"

Levels of Abstraction

- Behavioral level
  - State the behavior or function of the design
  - Without the information of the architecture
- Functional level(Register transfer level)
  - Data flow between registers
  - Data processing
- Structural level
  - Logic gates
  - Interconnections

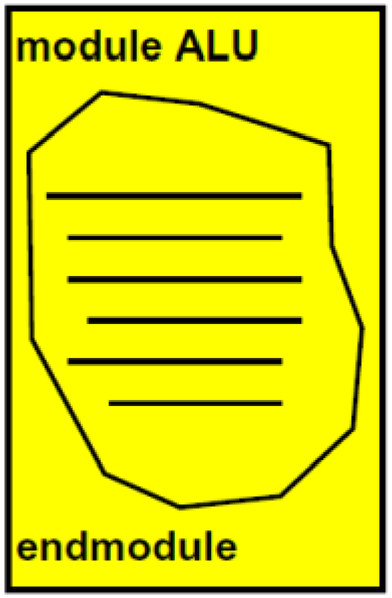
We are going to discuss


Behavioral representation
f
Functional representation
Structural representation
Physical representation



Verilog Architecture

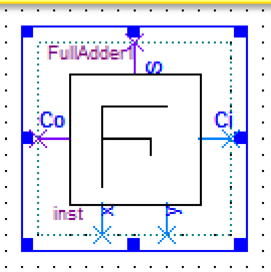
- module / endmodule
  - Basic building block
  - Can contain instances of other modules
  - All modules run concurrently
- Module ports
  - Input/output declaration
- Wire declaration
- Kernel hardware connection

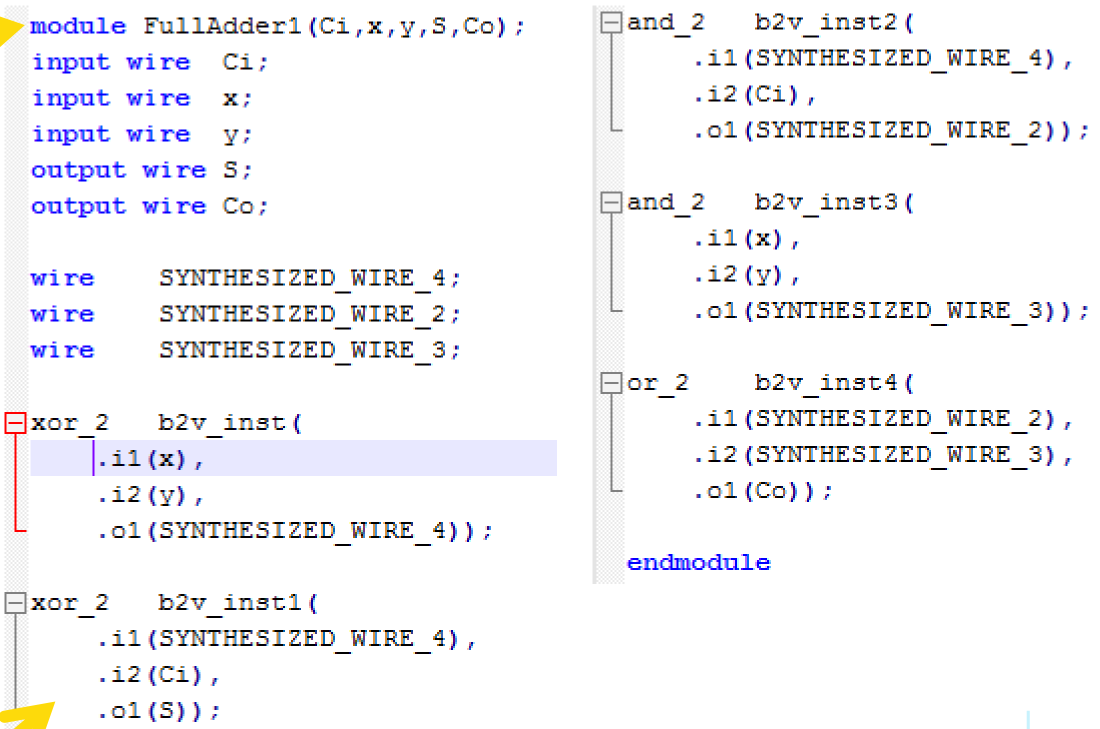


An Example - FullAdder

- Interconnections of logic elements

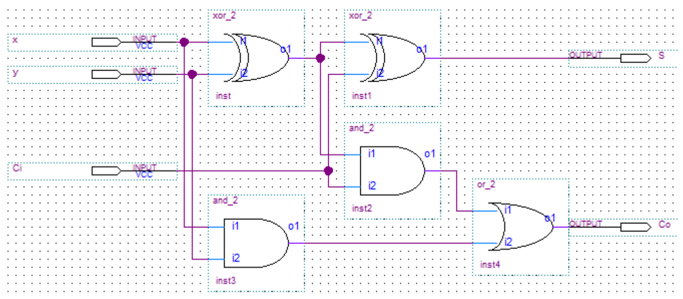
Module interface declaration





```
module FullAdder1(Ci,x,y,S,Co);
input wire  Ci;
input wire  x;
input wire  y;
output wire S;
output wire Co;

wire    SYNTHESIZED_WIRE_4;
wire    SYNTHESIZED_WIRE_2;
wire    SYNTHESIZED_WIRE_3;

xor_2   b2v_inst(
    .i1(x),
    .i2(y),
    .o1(SYNTHESIZED_WIRE_4));

xor_2   b2v_inst1(
    .i1(SYNTHESIZED_WIRE_4),
    .i2(Ci),
    .o1(S));
```

```
and_2   b2v_inst2(
    .i1(SYNTHESIZED_WIRE_4),
    .i2(Ci),
    .o1(SYNTHESIZED_WIRE_2));

and_2   b2v_inst3(
    .i1(x),
    .i2(y),
    .o1(SYNTHESIZED_WIRE_3));

or_2    b2v_inst4(
    .i1(SYNTHESIZED_WIRE_2),
    .i2(SYNTHESIZED_WIRE_3),
    .o1(Co));

endmodule
```



Descripting the interconnections

Gate Level Verilog Coding

- module declaration & I/O interface:
  - module name → like C++ class name

```
module FullAdder4(
input  wire [3:0] A,
input  wire [3:0] B,
output wire [4:0] S
);
```

```
module FullAdder4(A,B,S);
input  wire [3:0] A;
input  wire [3:0] B;
output wire [4:0] S;
```

These 2 format are equivalent
You can skip the word "wire" at I/O pin

- Wire declaration
  - Array of Wires

```
input  wire [3:0] A;
input  wire [3:0] B;
output wire [4:0] S;
```

```
wire    SYNTHESIZED_WIRE_0;
wire    SYNTHESIZED_WIRE_1;
wire    SYNTHESIZED_WIRE_2;
wire    SYNTHESIZED_WIRE_3;
```

- Link elements & instantiation of modules

```
FullAdder1  b2v_inst(
    .x(A[3]),
    .y(B[3]),
    .Ci(SYNTHESIZED_WIRE_0),
    .Co(S[4]),
    .S(S[3]));
```

instance name

Means: Co pin link to wire S[4]

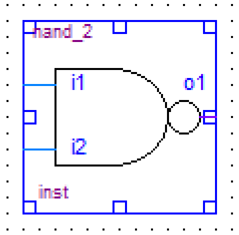
module name (class name)

- endmodule

An Example - NAND_2

- Descripting the behavior of logic elements

Module interface declaration



```
module nand_2(
input i1,
input i2,
output o1
);
    assign o1 = ~(i1&i2);
    specify
        (i1 => o1) = (3:3:4.5,3:3:5);
        (i2 => o1) = (3:3:4.5,3:3:5);
    endspecify
endmodule
```

Descripting the functional logic in higher level language (like C++)

Descripting the timing / delay information for simulation

(min T_{PLH} : avg T_{PLH} : max T_{PLH} , min T_{PHL} : avg T_{PHL} : max T_{PHL})

T_{PLH}: propagation delay from low to high

T_{PHL}: propagation delay from high to low

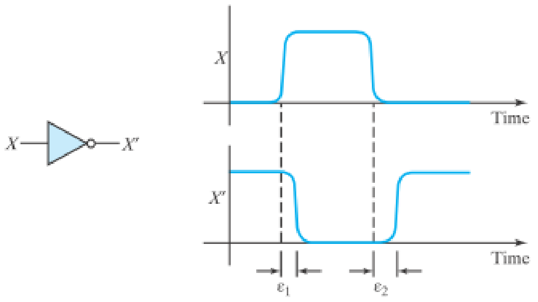


FIGURE 8-4 Propagation Delay in an Inverter

Verilog Coding - Comb.

- Operators

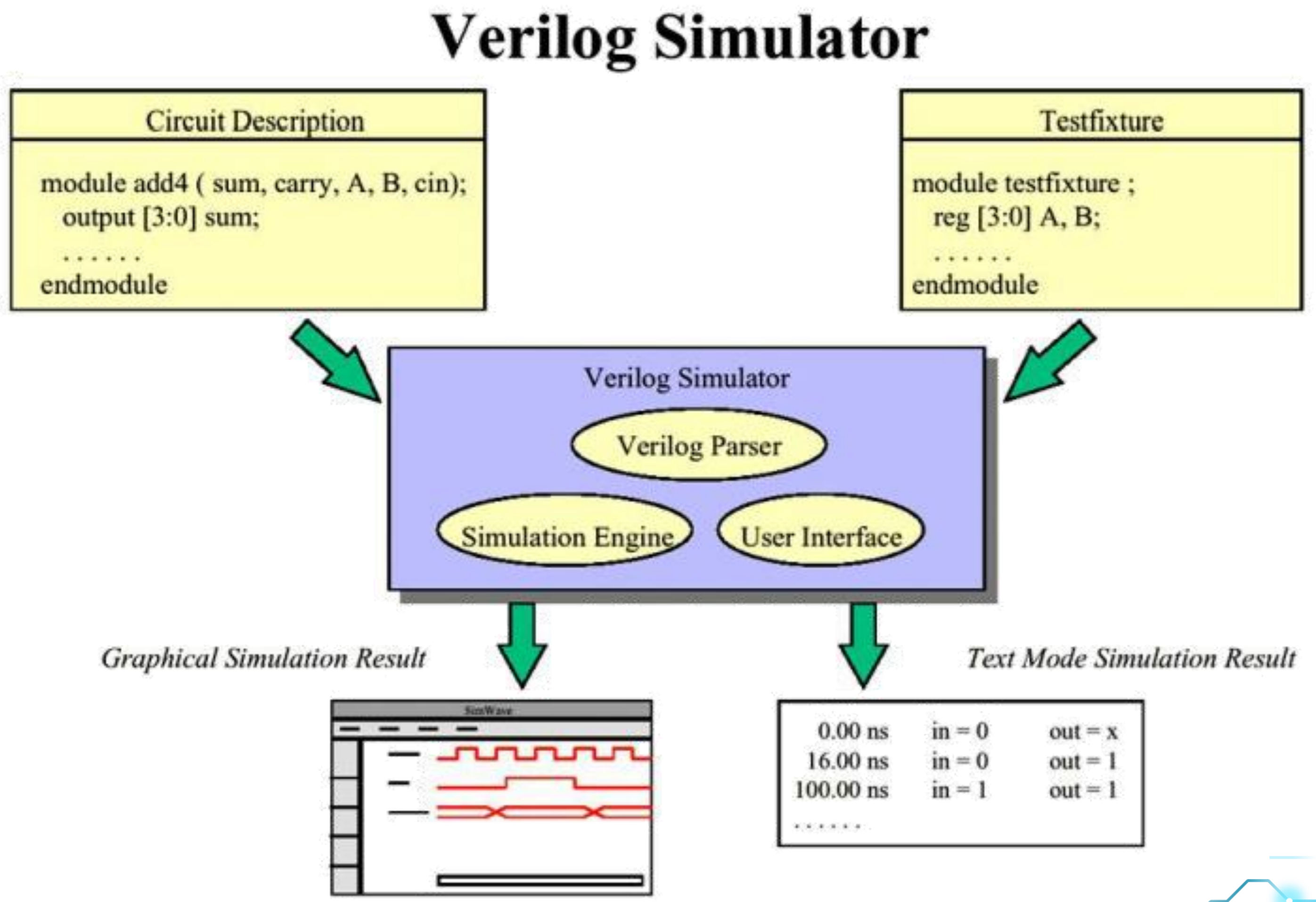
Arithmetic Operators	+, -, *, /, %
Relational Operators	<, <=, >, >=
Equality Operators	==, !=, ===, !==
Logical Operators	!, &&, ||
Bit-wise Operators	~, &, |, ^, ~^
Unary Reduction	&, ~&, |, ~|, ^, ~^
Shift Operators	>>, <<
Conditional Operators	?:
Concatenations	{}

Verilog Coding - Numbers

- Format: <size>'<base_format><number>
- <size> - decimal specification of bits count
- <base_format> - followed by arithmetic base of number
  - d or D – decimal (default if no base format given)
  - h or H – hexadecimal
  - o or O – octal
  - b or B – binary
- <number> - value given in base of base format
  - _ can be used for reading clarity
  - 0 extended
  - x and z are automatically extended

Format	Value	Representation
Decimal	110	**8'd110**
Binary	**01101110**	**8'b01101110**
Octal	**156**	**8'o156**
Hexadecimal	**6E**	**8'h6E**

Verilog Simulator



```
module add4 ( sum, carry, A, B, cin);
  output [3:0] sum;
  . . . . . .
endmodule
```

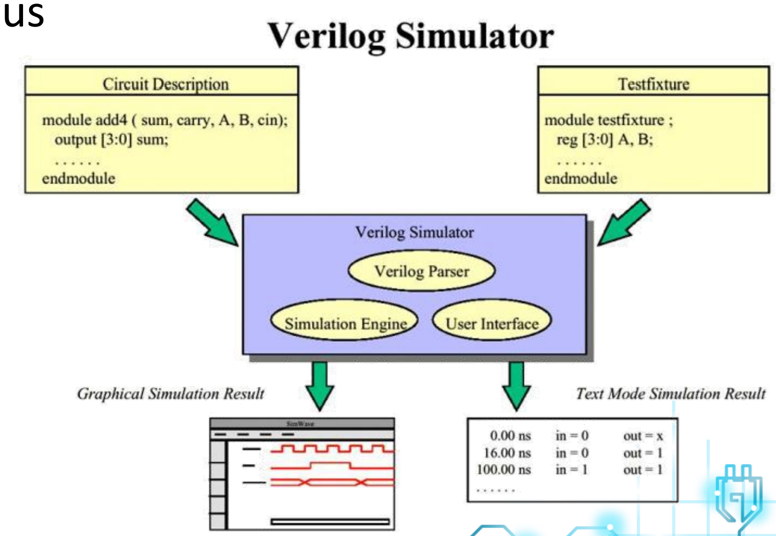
```
module testfixture ;
  reg [3:0] A, B;
  . . . . . .
endmodule
```

```
   0.00 ns    in = 0     out = x
  16.00 ns    in = 0     out = 1
 100.00 ns    in = 1     out = 1
 . . . . . .
```

Switching Circuit & Logic Design

Verilog Design Flow (Simple)

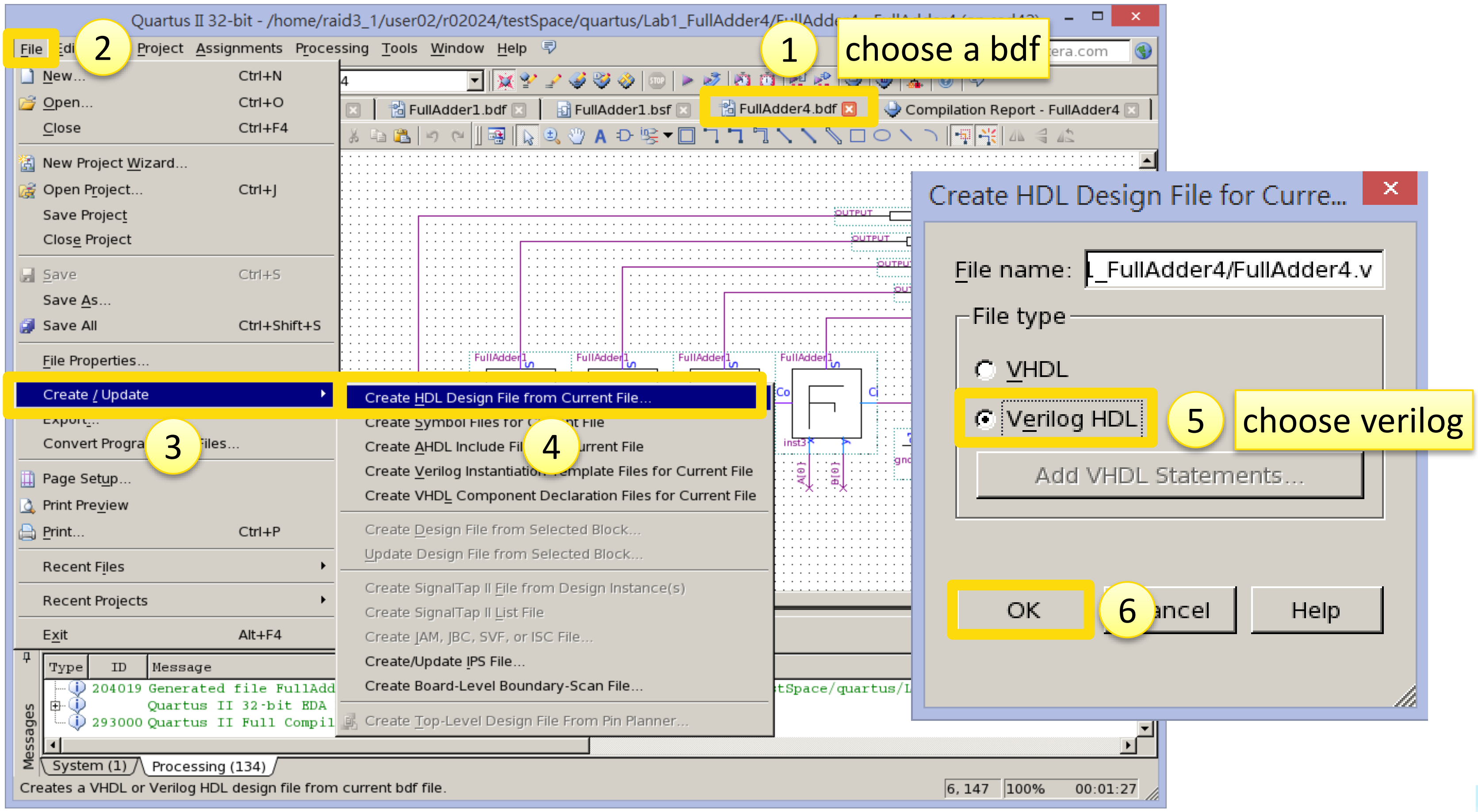
- Design from schematic to verilog:
  1. Prepare verilog design files
     - Convert all .bdf files to verilog by Quartus
  2. Prepare verilog testbench
     - Testbench: set inputs and check outputs
     - Convert a .vwf file to verilog by Quartus
  3. Compilation & Simulation
  4. Look over the result & debug



Verilog Simulation by Ourselves

Switching Circuit & Logic Design

Generate Verilog Design File from Schematic



Quartus II 32-bit - /home/raid3_1/user02/r02024/testSpace/quartus/Lab1_FullAdder4/FullAdde

File Edit Project Assignments Processing Tools Window Help

FullAdder1.bdf | FullAdder1.bsf | FullAdder4.bdf | Compilation Report - FullAdder4

New...	Ctrl+N
Open...	Ctrl+O
Close	Ctrl+F4
New Project Wizard...	
Open Project...	Ctrl+J
Save Project	
Close Project	
Save	Ctrl+S
Save As...	
Save All	Ctrl+Shift+S
File Properties...	
Create / Update	▸
Export...	
Convert Programming Files...	
Page Setup...	
Print Preview	
Print...	Ctrl+P
Recent Files	▸
Recent Projects	▸
Exit	Alt+F4

Create HDL Design File from Current File...
Create Symbol Files for Current File
Create AHDL Include Files for Current File
Create Verilog Instantiation Template Files for Current File
Create VHDL Component Declaration Files for Current File
Create Design File from Selected Block...
Update Design File from Selected Block...
Create SignalTap II File from Design Instance(s)
Create SignalTap II List File
Create JAM, JBC, SVF, or ISC File...
Create/Update IPS File...
Create Board-Level Boundary-Scan File...
Create Top-Level Design File From Pin Planner...

Create HDL Design File for Curre...

File name: _FullAdder4/FullAdder4.v

File type

VHDL [✓ Verilog HDL]

Add VHDL Statements...

OK | Cancel | Help

Type	ID	Message
	204019	Generated file FullAdd
		Quartus II 32-bit EDA
	293000	Quartus II Full Compil

System (1) | Processing (134)

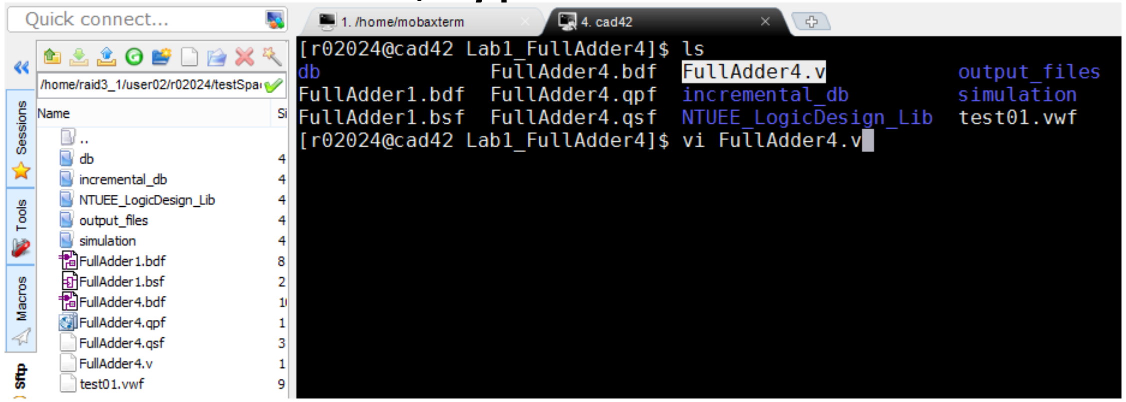
Creates a VHDL or Verilog HDL design file from current bdf file.

6, 147 | 100% | 00:01:27

Switching Circuit & Logic Design

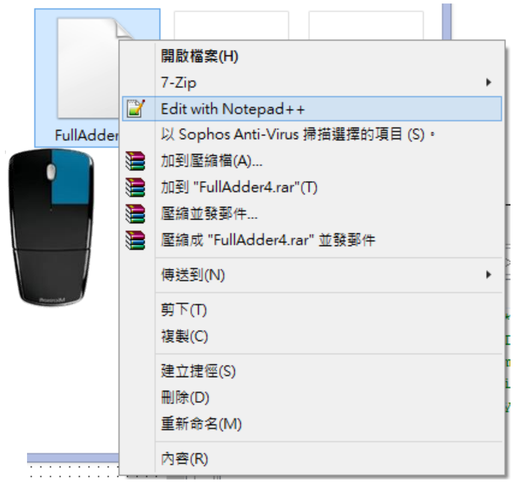
Modify Verilog Design File

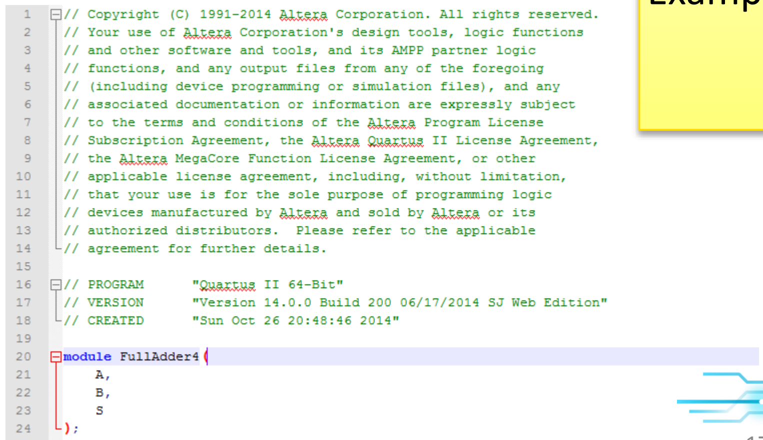
- For workstation, type: vi Fulladder4.v



```
[r02024@cad42 Lab1_FullAdder4]$ ls
db              FullAdder4.bdf   FullAdder4.v           output_files
FullAdder1.bdf  FullAdder4.qpf   incremental_db         simulation
FullAdder1.bsf  FullAdder4.qsf   NTUEE_LogicDesign_Lib  test01.vwf
[r02024@cad42 Lab1_FullAdder4]$ vi FullAdder4.v
```

- For Windows, you can use Notepad++



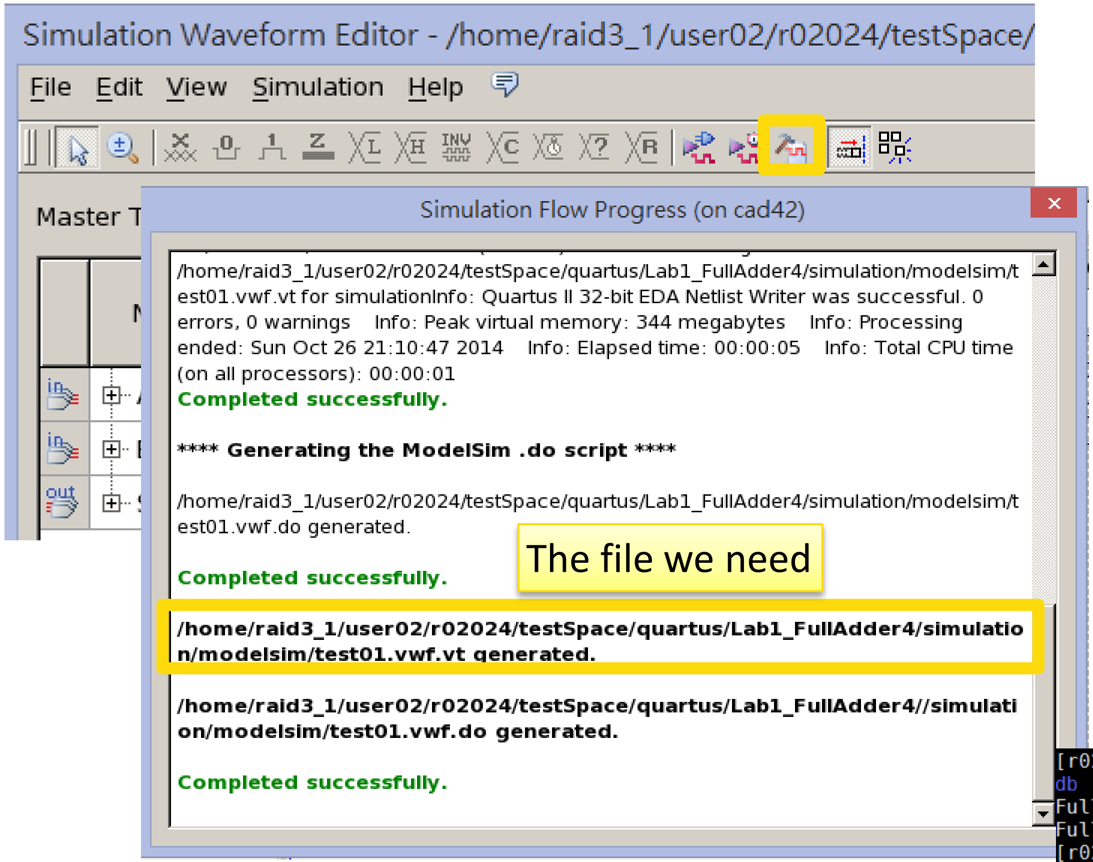


```
// Copyright (C) 1991-2014 Altera Corporation. All rights reserved.
// Your use of Altera Corporation's design tools, logic functions
// and other software and tools, and its AMPP partner logic
// functions, and any output files from any of the foregoing
// (including device programming or simulation files), and any
// associated documentation or information are expressly subject
// to the terms and conditions of the Altera Program License
// Subscription Agreement, the Altera Quartus II License Agreement,
// the Altera MegaCore Function License Agreement, or other
// applicable license agreement, including, without limitation,
// that your use is for the sole purpose of programming logic
// devices manufactured by Altera and sold by Altera or its
// authorized distributors.  Please refer to the applicable
// agreement for further details.

// PROGRAM      "Quartus II 64-Bit"
// VERSION      "Version 14.0.0 Build 200 06/17/2014 SJ Web Edition"
// CREATED      "Sun Oct 26 20:48:46 2014"

module FullAdder4 (
    A,
    B,
    S
);
```

If you want to run Verilog simulation outside Quartus (by other compiler), you need to transform all .bdf files into .v files.
(.bsf not needed)
Example here, needs to have:
FullAdder4.v,
FullAdder1.v

Switching Circuit & Logic Design

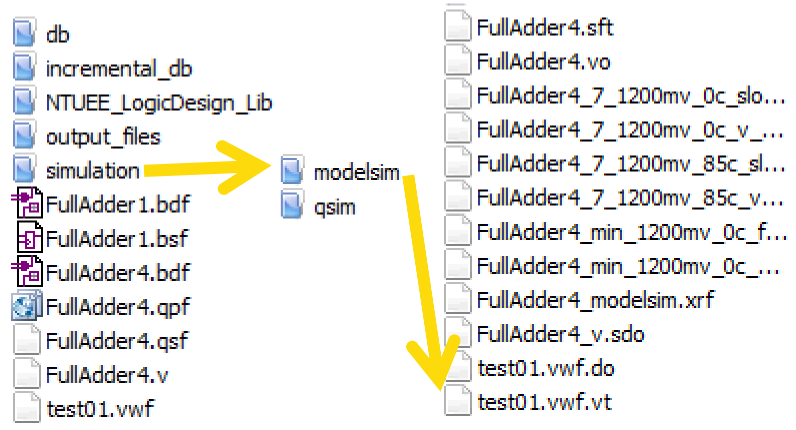
Generate Verilog Testbench from .vwf



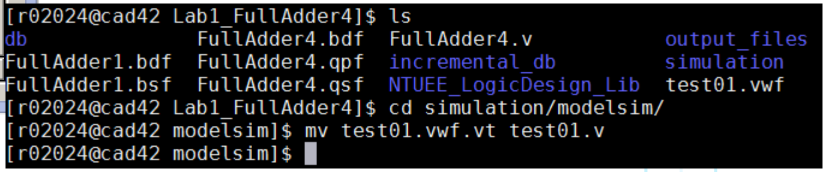
Simulation Waveform Editor - /home/raid3_1/user02/r02024/testSpace/
File Edit View Simulation Help
Master T
Simulation Flow Progress (on cad42)

```
/home/raid3_1/user02/r02024/testSpace/quartus/Lab1_FullAdder4/simulation/modelsim/t
est01.vwf.vt for simulationInfo: Quartus II 32-bit EDA Netlist Writer was successful. 0
errors, 0 warnings    Info: Peak virtual memory: 344 megabytes    Info: Processing
ended: Sun Oct 26 21:10:47 2014    Info: Elapsed time: 00:00:05    Info: Total CPU time
(on all processors): 00:00:01
Completed successfully.

**** Generating the ModelSim .do script ****

/home/raid3_1/user02/r02024/testSpace/quartus/Lab1_FullAdder4/simulation/modelsim/t
est01.vwf.do generated.

Completed successfully.

/home/raid3_1/user02/r02024/testSpace/quartus/Lab1_FullAdder4/simulatio
n/modelsim/test01.vwf.vt generated.

/home/raid3_1/user02/r02024/testSpace/quartus/Lab1_FullAdder4//simulati
on/modelsim/test01.vwf.do generated.

Completed successfully.
```

The file we need



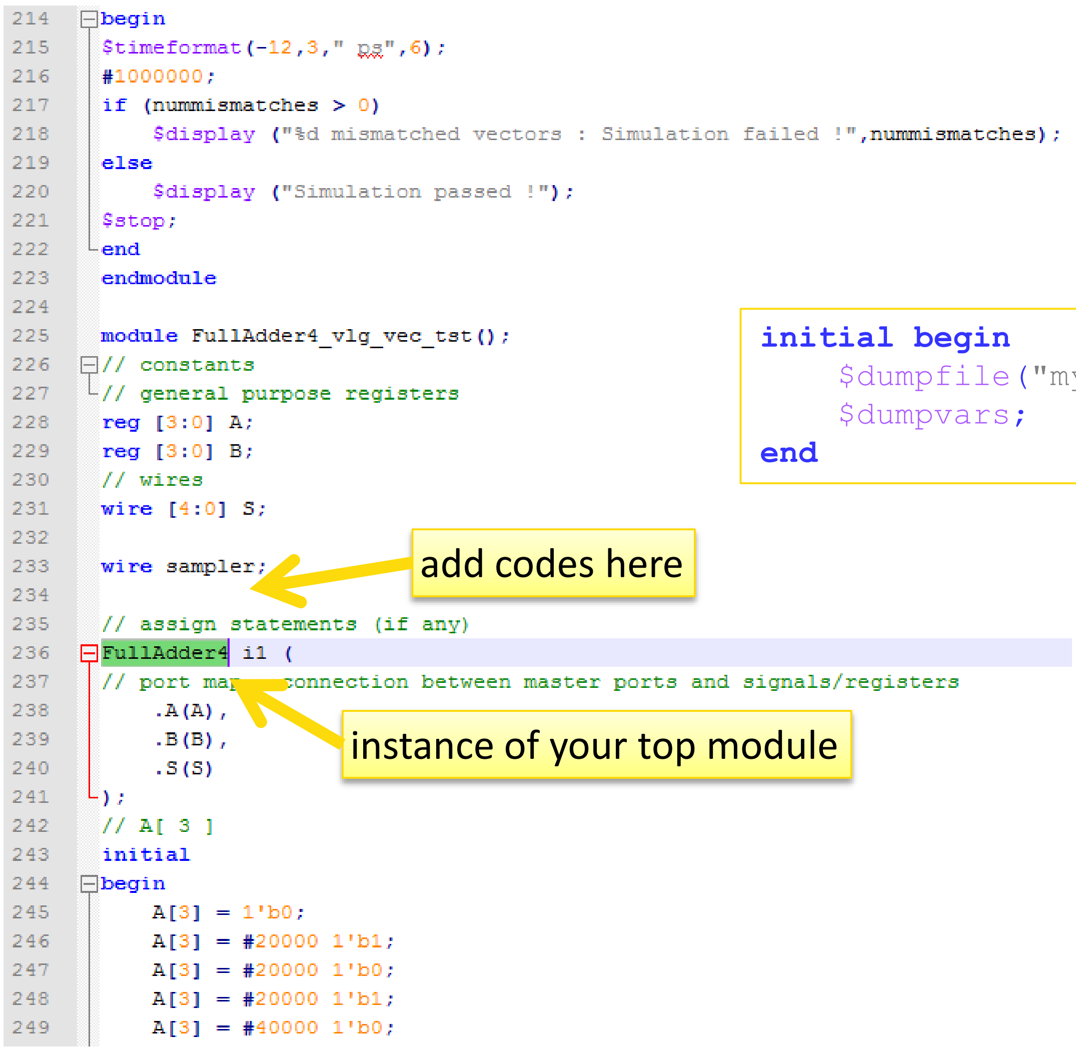
Find this file, rename it as .v file
Then open it



```
[r02024@cad42 Lab1_FullAdder4]$ ls
db              FullAdder4.bdf  FullAdder4.v           output_files
FullAdder1.bdf  FullAdder4.qpf  incremental_db         simulation
FullAdder1.bsf  FullAdder4.qsf  NTUEE_LogicDesign_Lib  test01.vwf
[r02024@cad42 Lab1_FullAdder4]$ cd simulation/modelsim/
[r02024@cad42 modelsim]$ mv test01.vwf.vt test01.v
[r02024@cad42 modelsim]$ 
```

Switching Circuit & Logic Design

Add some codes in testbench



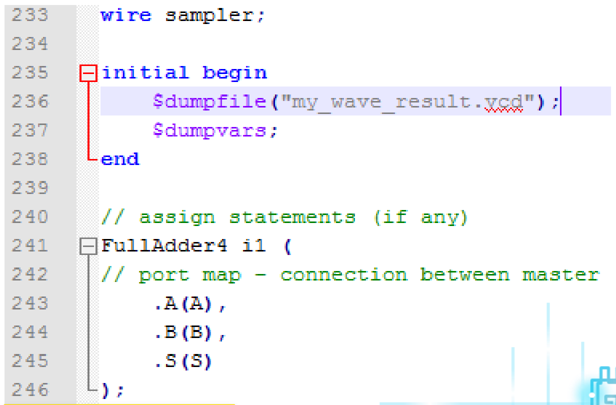
```
begin
$timeformat(-12,3," ps",6);
#1000000;
if (nummismatches > 0)
    $display ("%d mismatched vectors : Simulation failed !",nummismatches);
else
    $display ("Simulation passed !");
$stop;
end
endmodule

module FullAdder4_vlg_vec_tst();
// constants
// general purpose registers
reg [3:0] A;
reg [3:0] B;
// wires
wire [4:0] S;

wire sampler;

// assign statements (if any)
FullAdder4 i1 (
// port map - connection between master ports and signals/registers
    .A(A),
    .B(B),
    .S(S)
);
// A[ 3 ]
initial
begin
    A[3] = 1'b0;
    A[3] = #20000 1'b1;
    A[3] = #20000 1'b0;
    A[3] = #20000 1'b1;
    A[3] = #40000 1'b0;
```

add codes here

instance of your top module

```
initial begin
    $dumpfile("my_wave_result.vcd");
    $dumpvars;
end
```

These code will ask compiler to record all signals for you to see



```
wire sampler;

initial begin
    $dumpfile("my_wave_result.vcd");
    $dumpvars;
end

// assign statements (if any)
FullAdder4 i1 (
// port map - connection between master
    .A(A),
    .B(B),
    .S(S)
);
```

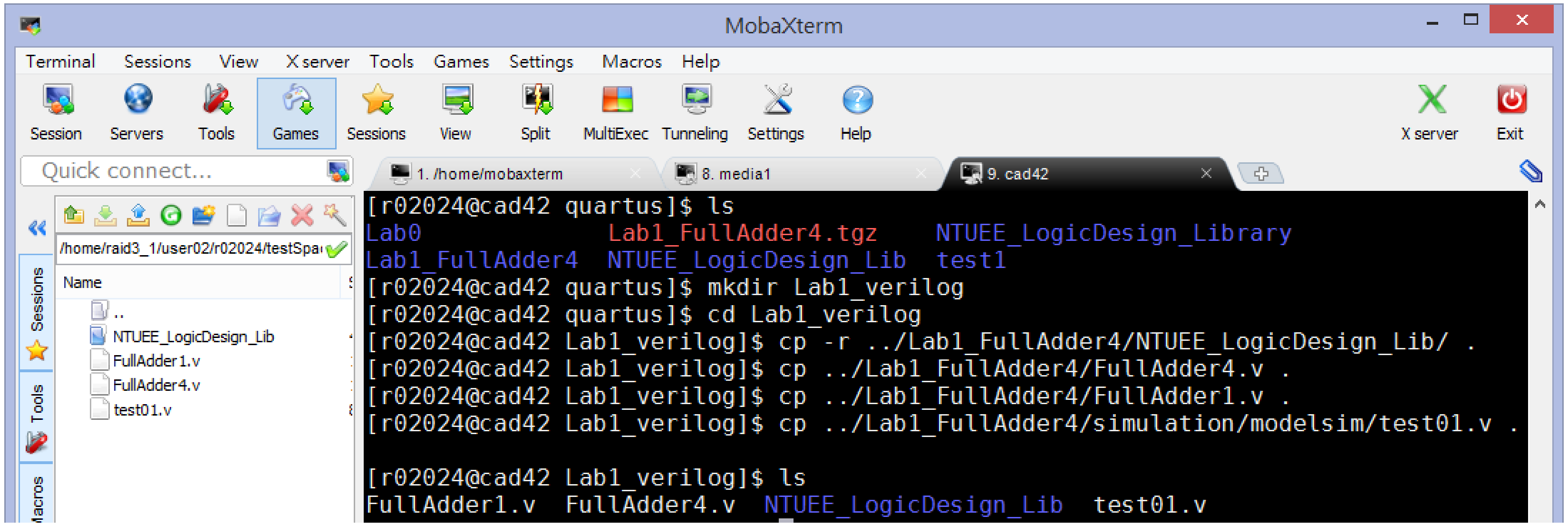
save this file

Simulation by Verilog Compiler

Switching Circuit & Logic Design

Files Prepare

- Make a directory and put files we need together


MobaXterm
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Settings Help X server Exit
Quick connect...
1. /home/mobaxterm
8. media1
9. cad42
/home/raid3_1/user02/r02024/testSpa
Name
..
NTUEE_LogicDesign_Lib
FullAdder1.v
FullAdder4.v
test01.v
Sessions
Tools
Macros

```
[r02024@cad42 quartus]$ ls
Lab0                Lab1_FullAdder4.tgz    NTUEE_LogicDesign_Library
Lab1_FullAdder4  NTUEE_LogicDesign_Lib  test1
[r02024@cad42 quartus]$ mkdir Lab1_verilog
[r02024@cad42 quartus]$ cd Lab1_verilog
[r02024@cad42 Lab1_verilog]$ cp -r ../Lab1_FullAdder4/NTUEE_LogicDesign_Lib/ .
[r02024@cad42 Lab1_verilog]$ cp ../Lab1_FullAdder4/FullAdder4.v .
[r02024@cad42 Lab1_verilog]$ cp ../Lab1_FullAdder4/FullAdder1.v .
[r02024@cad42 Lab1_verilog]$ cp ../Lab1_FullAdder4/simulation/modelsim/test01.v .

[r02024@cad42 Lab1_verilog]$ ls
FullAdder1.v  FullAdder4.v  NTUEE_LogicDesign_Lib  test01.v
```

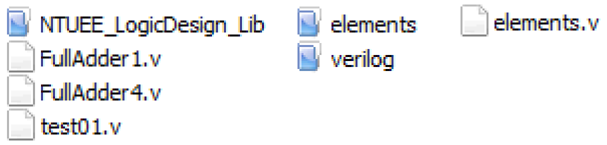
```
ls
mkdir Lab1_verilog
cd Lab1_verilog
cp -r ../Lab1_FullAdder4/NTUEE_LogicDesign_Lib/ .
cp ../Lab1_FullAdder4/FullAdder4.v .
cp ../Lab1_FullAdder4/FullAdder1.v .
cp ../Lab1_FullAdder4/simulation/modelsim/test01.v .
ls
```

Run Simulation

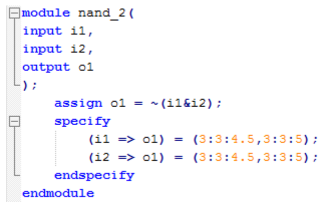
- source some tool:

```
source ~cvsd/cvsd.cshrc
source ~cvsd/verdi.cshrc
```

If you have problem sourcing verdi, please try:
source /usr/spring_soft/CIC/verdi.cshrc



```
NTUEE_LogicDesign_Lib    elements    elements.v
FullAdder1.v             verilog
FullAdder4.v
test01.v
```

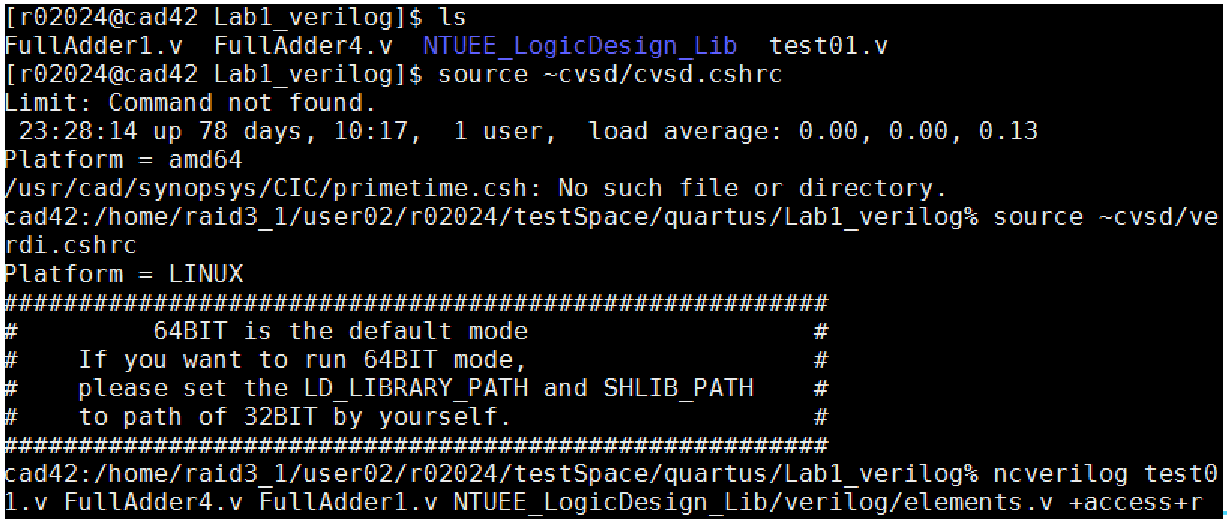


```
module nand_2(
input i1,
input i2,
output o1
);
    assign o1 = ~(i1&i2);
    specify
        (i1 => o1) = (3:3:4.5,3:3:5);
        (i2 => o1) = (3:3:4.5,3:3:5);
    endspecify
endmodule
```

- Run:

```
ncverilog test01.v FullAdder4.v FullAdder1.v
NTUEE_LogicDesign_Lib/verilog/elements.v +access+r
```

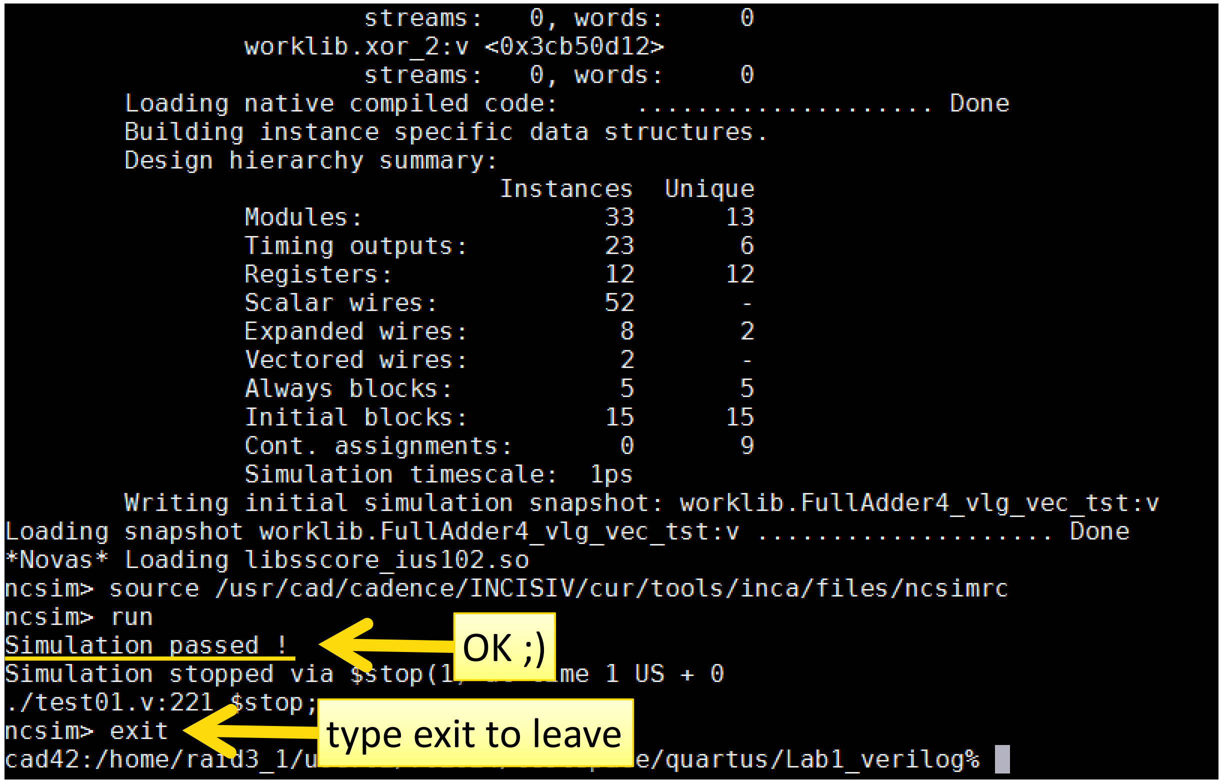
in one line, no “\n”



```
[r02024@cad42 Lab1_verilog]$ ls
FullAdder1.v  FullAdder4.v  NTUEE_LogicDesign_Lib  test01.v
[r02024@cad42 Lab1_verilog]$ source ~cvsd/cvsd.cshrc
Limit: Command not found.
 23:28:14 up 78 days, 10:17,  1 user,  load average: 0.00, 0.00, 0.13
Platform = amd64
/usr/cad/synopsys/CIC/primetime.csh: No such file or directory.
cad42:/home/raid3_1/user02/r02024/testSpace/quartus/Lab1_verilog% source ~cvsd/ve
rdi.cshrc
Platform = LINUX
#######################################################
#         64BIT is the default mode                   #
#    If you want to run 64BIT mode,                   #
#    please set the LD_LIBRARY_PATH and SHLIB_PATH    #
#    to path of 32BIT by yourself.                    #
#######################################################
cad42:/home/raid3_1/user02/r02024/testSpace/quartus/Lab1_verilog% ncverilog test0
1.v FullAdder4.v FullAdder1.v NTUEE_LogicDesign_Lib/verilog/elements.v +access+r
```

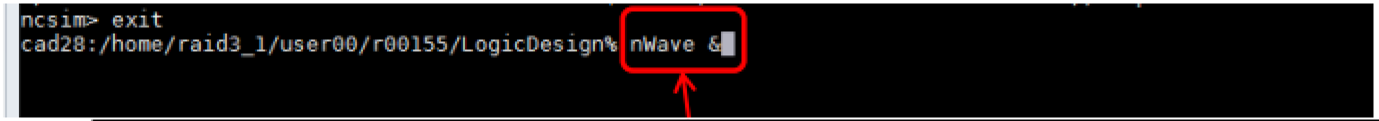
Switching Circuit & Logic Design

End of Simulation



```
                        streams:   0, words:     0
                worklib.xor_2:v <0x3cb50d12>
                        streams:   0, words:     0
        Loading native compiled code:     .................... Done
        Building instance specific data structures.
        Design hierarchy summary:
                                 Instances  Unique
                Modules:                33      13
                Timing outputs:         23       6
                Registers:              12      12
                Scalar wires:           52       -
                Expanded wires:          8       2
                Vectored wires:          2       -
                Always blocks:           5       5
                Initial blocks:         15      15
                Cont. assignments:       0       9
                Simulation timescale:  1ps
        Writing initial simulation snapshot: worklib.FullAdder4_vlg_vec_tst:v
Loading snapshot worklib.FullAdder4_vlg_vec_tst:v .................... Done
*Novas* Loading libsscore_ius102.so
ncsim> source /usr/cad/cadence/INCISIV/cur/tools/inca/files/ncsimrc
ncsim> run
Simulation passed !
Simulation stopped via $stop(1            me 1 US + 0
./test01.v:221 $stop;
ncsim> exit
cad42:/home/raid3_1/u                          e/quartus/Lab1_verilog%
```

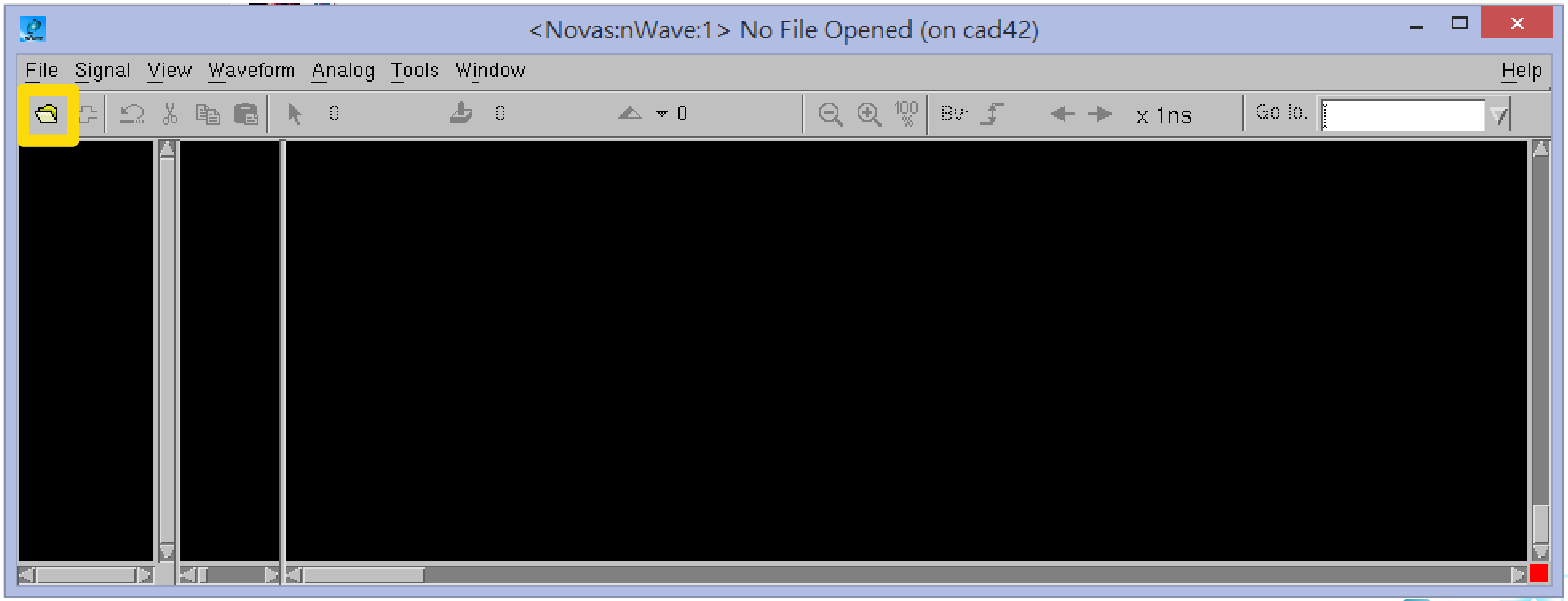
You can use nWave to see results



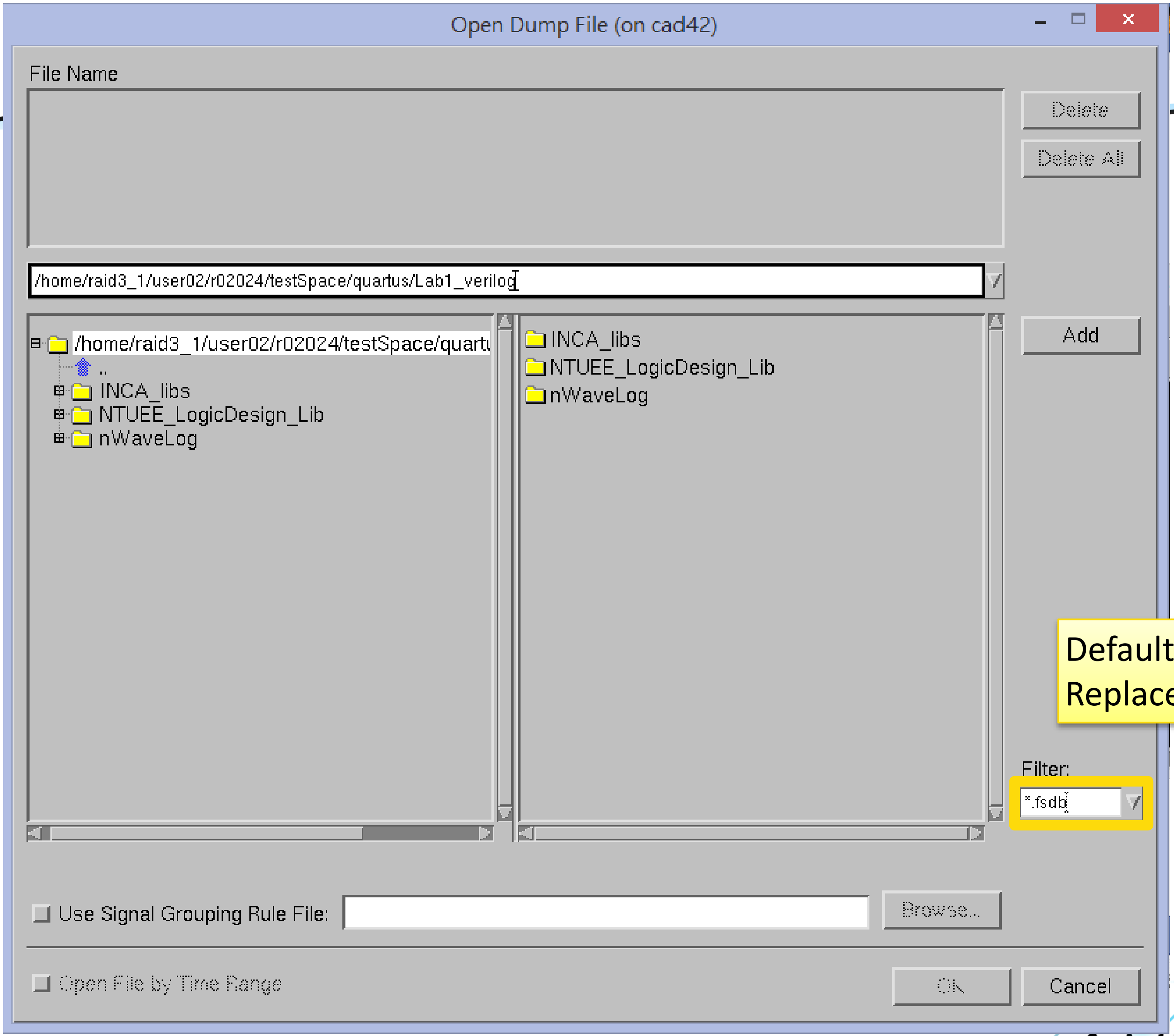
```
ncsim> exit
cad28:/home/raid3_1/user00/r00155/LogicDesign% nWave &
```

nWave & : open the tool “nWave”

- “&” means open in background mode



<Novas:nWave:1> No File Opened (on cad42)
File Signal View Waveform Analog Tools Window Help
x 1ns
Go to:



/home/raid3_1/user02/r02024/testSpace/quartus/Lab1_verilog

```
/home/raid3_1/user02/r02024/testSpace/quartu
  ..
  INCA_libs
  NTUEE_LogicDesign_Lib
  nWaveLog
```

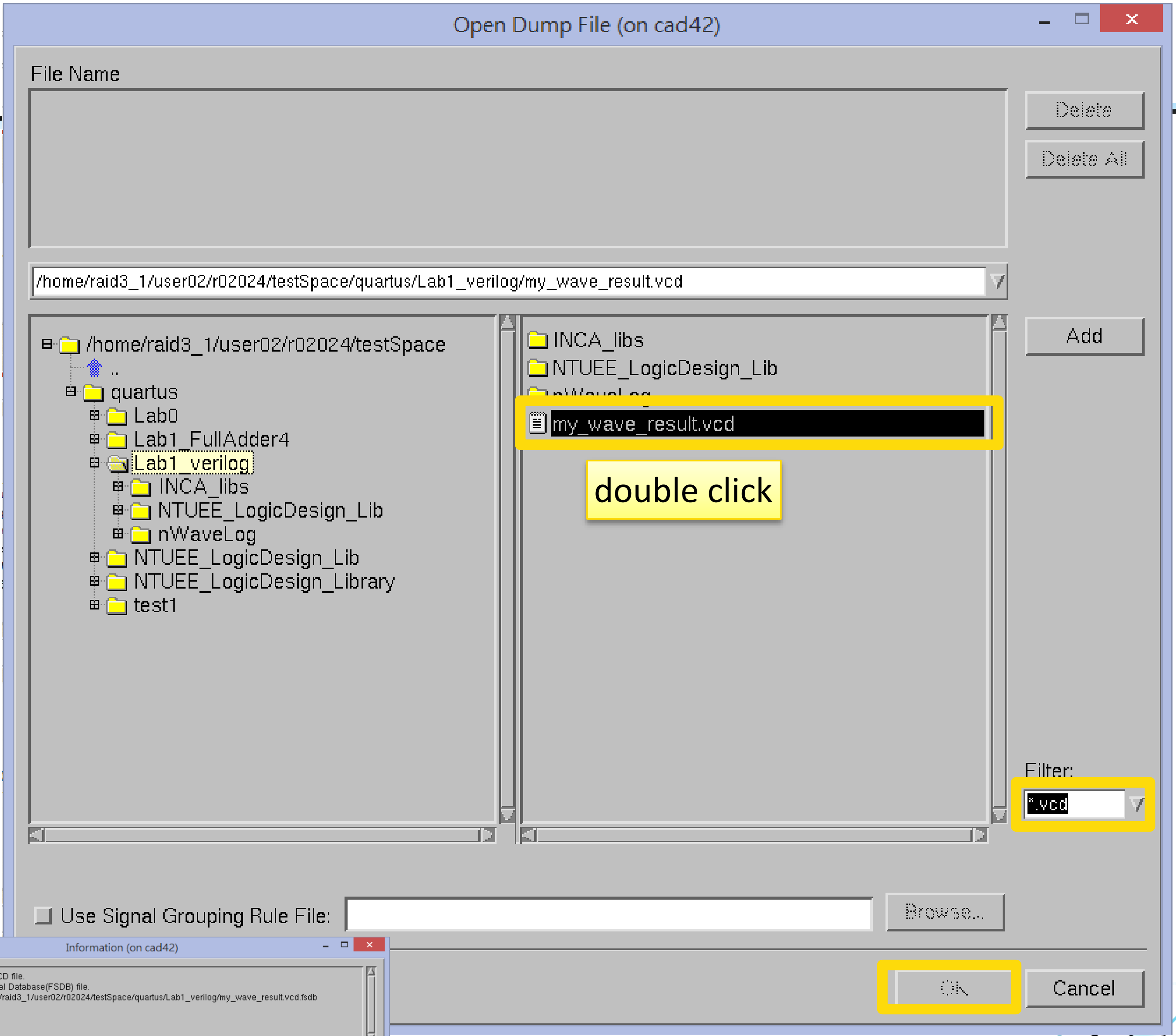
```
INCA_libs
NTUEE_LogicDesign_Lib
nWaveLog
```

Filter: *.fsdb

☐ Use Signal Grouping Rule File:

☐ Open File by Time Range

Default is ***.fsdb**
Replace it with ***.vcd**


Open Dump File (on cad42)

File Name

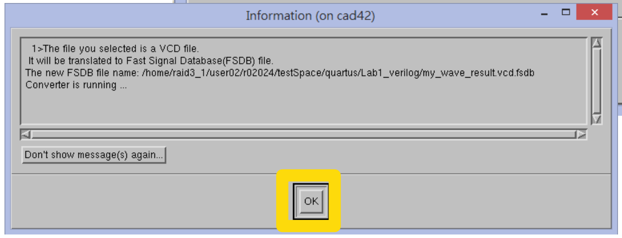
```
/home/raid3_1/user02/r02024/testSpace/quartus/Lab1_verilog/my_wave_result.vcd
```

```
⊟ /home/raid3_1/user02/r02024/testSpace
    ..
  ⊟ quartus
    ⊞ Lab0
    ⊞ Lab1_FullAdder4
    ⊟ Lab1_verilog
      ⊞ INCA_libs
      ⊞ NTUEE_LogicDesign_Lib
      ⊞ nWaveLog
    ⊞ NTUEE_LogicDesign_Lib
    ⊞ NTUEE_LogicDesign_Library
    ⊞ test1
```

```
INCA_libs
NTUEE_LogicDesign_Lib
nWaveLog
my_wave_result.vcd
```

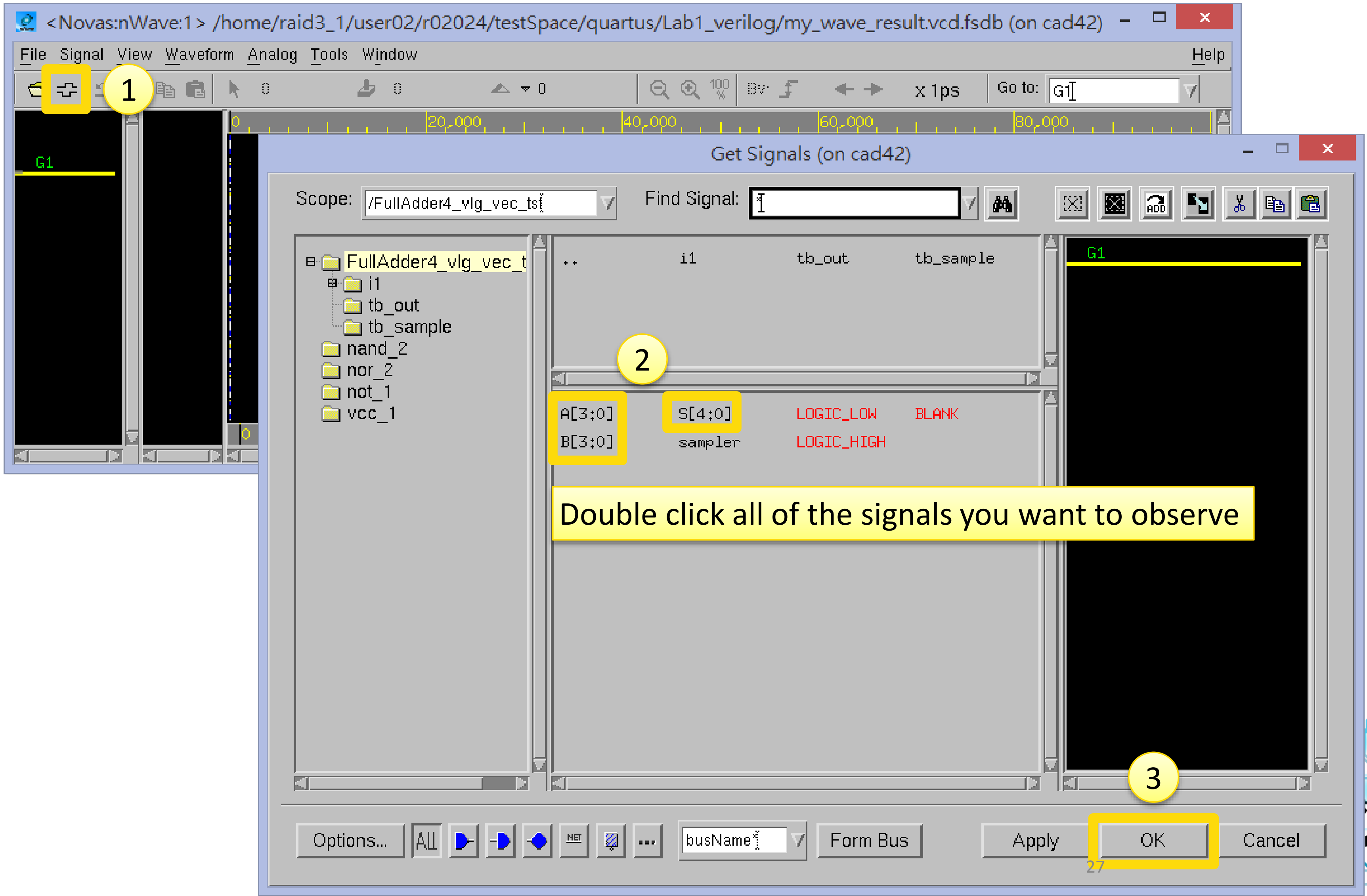
Filter: *.vcd

☐ Use Signal Grouping Rule File:


Information (on cad42)

```
 1>The file you selected is a VCD file.
 It will be translated to Fast Signal Database(FSDB) file.
The new FSDB file name: /home/raid3_1/user02/r02024/testSpace/quartus/Lab1_verilog/my_wave_result.vcd.fsdb
Converter is running ...
```

nWave



<Novas:nWave:1> /home/raid3_1/user02/r02024/testSpace/quartus/Lab1_verilog/my_wave_result.vcd.fsdb (on cad42)

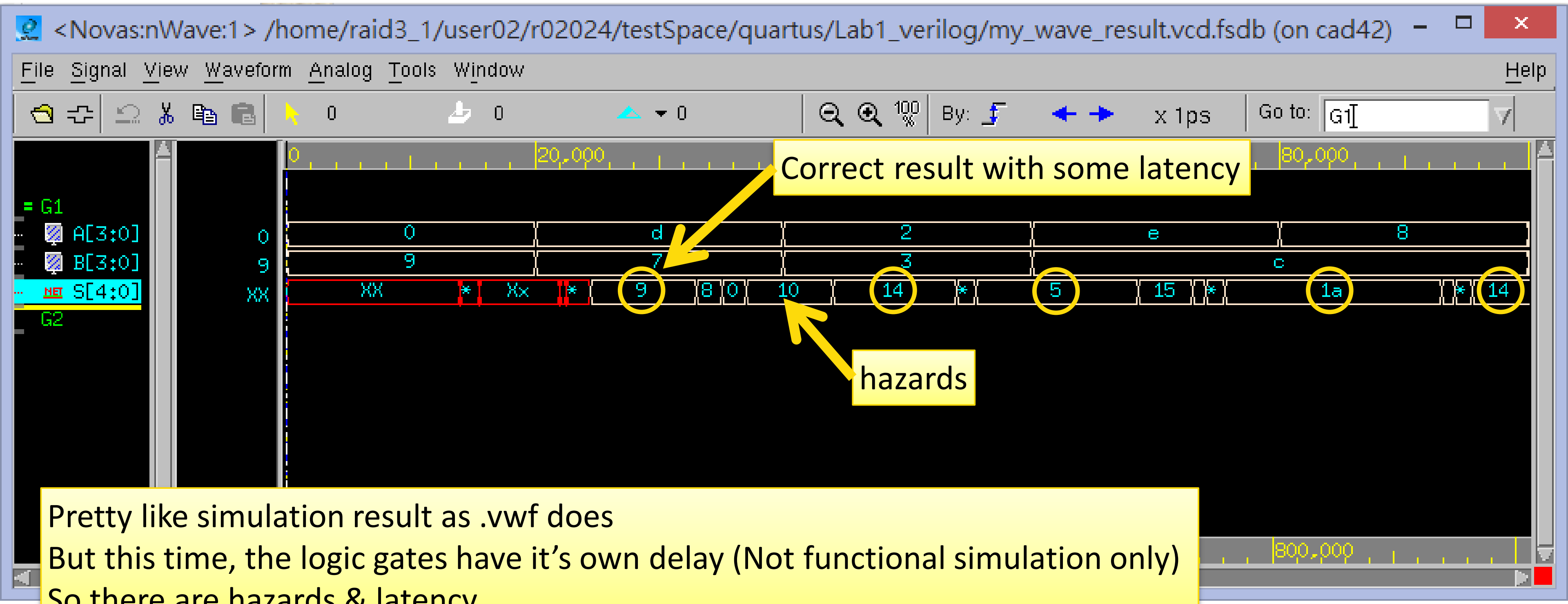
- FullAdder4_vlg_vec_t
  - i1
  - tb_out
  - tb_sample
- nand_2
- nor_2
- not_1
- vcc_1

```
..          i1          tb_out          tb_sample
```

```
A[3:0]      S[4:0]      LOGIC_LOW       BLANK
B[3:0]      sampler     LOGIC_HIGH
```

Double click all of the signals you want to observe

nWave

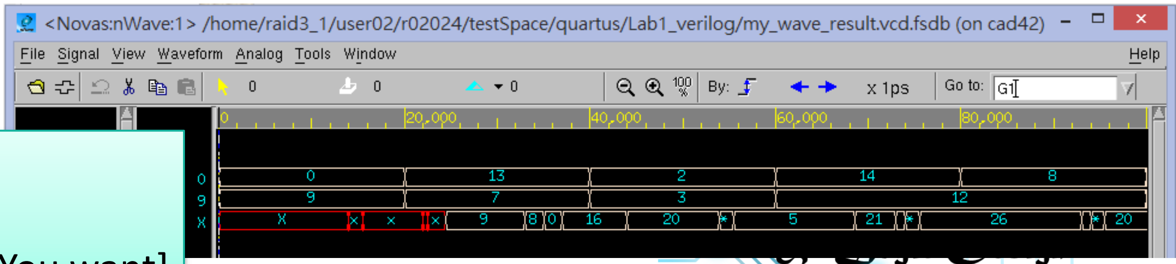


<Novas:nWave:1> /home/raid3_1/user02/r02024/testSpace/quartus/Lab1_verilog/my_wave_result.vcd.fsdb (on cad42)
File Signal View Waveform Analog Tools Window Help
By: x 1ps Go to: G1
G1
A[3:0]
B[3:0]
S[4:0]
G2
0
20,000
80,000
800,000

Correct result with some latency

hazards

Pretty like simulation result as .vwf does
But this time, the logic gates have it's own delay (Not functional simulation only)
So there are hazards & latency



<Novas:nWave:1> /home/raid3_1/user02/r02024/testSpace/quartus/Lab1_verilog/my_wave_result.vcd.fsdb (on cad42)
File Signal View Waveform Analog Tools Window Help
By: x 1ps Go to: G1
0
20,000
40,000
60,000
80,000

[Hint] Default format: Hex,
you can change format at:
Waveform→Signal Value Radix→[You want]

Verilog on Windows …

- You can use iVerilog
  - Not recommended
  - ncverilog on workstation is much more powerful
    - iVerilog is free, it has some tools that can act as nWave

Summary

Summary

Design module

- From .bdf

- Divide-and-Conquer
- Partition the whole design into several parts
  - Derive the architecture of each sub-module
- Make architecture figures before you write Verilog codes
  - Create hardware design in gate level
  - Connection of sub-modules

Test-bench

- From .vwf

- Feed input data and compare output values at right timing slots
- Usually describe in behavioral level
- Not real hardware, just like software programming (e.g. C/C++)

Switching Circuit & Logic Design

Summary

	Declaration	Implementation	Test or use it
Quartus Schematic Module	Module Block Symbol File with I/O port info. FullAdder1.bsf	Ckt Design Block diagram File FullAdder1.bdf	Wave Form test pattern & result test01.vwf
Verilog HDL Design	Module declaration & instantiation	Comb. Logic、interconnection、timing FullAdder1.v FullAdder4.v	Testbench test pattern & dump test01.v Testbench have an instance of top design
C++ Object	Object Class (header file) with data, interface MyClass.h	Object Class (source file) MyClass.cpp	instance of the class used in somewhere of the program main.cpp main.exe

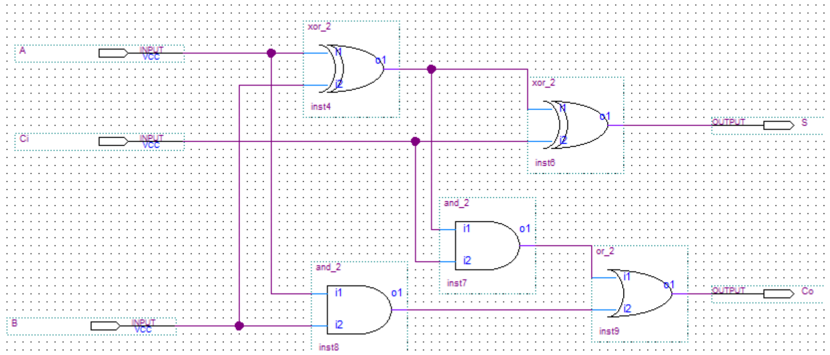
```
module nand_2(
 input i1,
 input i2,
 output o1
);
```

```
      assign o1 = ~(i1&i2);
      specify
          (i1 => o1) = (3:3:4.5,3:3:5);
          (i2 => o1) = (3:3:4.5,3:3:5);
      endspecify
endmodule
```

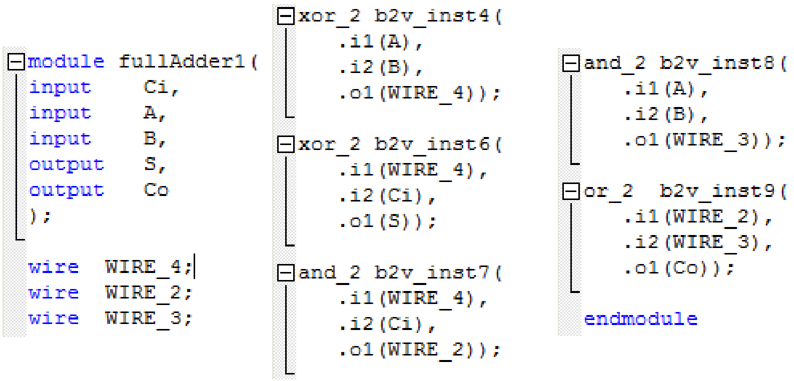
```
initial begin
    $dumpfile("my_wave_result.vcd");
    $dumpvars;
end
```

Schematic v.s. HDL

Schematic Design
(Block Diagram Design)



HDL Design
(Hardware description language)



```
module fullAdder1(
input     Ci,
input     A,
input     B,
output    S,
output    Co
);

wire    WIRE_4;
wire    WIRE_2;
wire    WIRE_3;

xor_2 b2v_inst4(
    .i1(A),
    .i2(B),
    .o1(WIRE_4));

xor_2 b2v_inst6(
    .i1(WIRE_4),
    .i2(Ci),
    .o1(S));

and_2 b2v_inst7(
    .i1(WIRE_4),
    .i2(Ci),
    .o1(WIRE_2));

and_2 b2v_inst8(
    .i1(A),
    .i2(B),
    .o1(WIRE_3));

or_2  b2v_inst9(
    .i1(WIRE_2),
    .i2(WIRE_3),
    .o1(Co));

endmodule
```

- **Visualizing**, easy to realize the hierarchical **architecture**
- Easy for beginners to catch up
- But hard to scale up for large designs
- Hard for programs to read

- Explicit, easy for programs (EDA tools) reading & processing
- Needs to learn a new computer language (HDL) first
- Easy to scale up for complex designs
- **Higher level logic design** (ex. RT level)

Switching Circuit & Logic Design

Better ask twice than lose you way once.

Q&A

Reference

Acknowledgement

- Authors of Basic Logic Design via Verilog HDL
  - Ver. 1: Chen-han Tsai
  - Ver. 2: Chih-hao Chao
  - Ver. 3: Xin-Yu Shi
  - Ver. 4: Bo-Yuan Peng
  - Ver. 5: Chieh-Chuan Chiu & Chieh-Chi Kao
  - Ver. 6: Yu-Hao Chen & Ming-Chun Hsiao
  - Ver. 7: Yu-Hao Chen & Cheng-Rung Tsai

Reference

- Textbook
  - Fundamentals of Logic Design, Charles H. Roth, Jr., Larry L. Kinney
- Dclab lecture:
  - Verilog Coding Guideline, 吳柏辰
  - My First FPGA for Altera DE2-115 Board, 吳柏辰
- CVSD lecture:
  - Computer-aided VLSI System Design Linux / Unix Tutorial, 陳滿蓉

Switching Circuit & Logic Design