



臺灣大學

IC Design HW3 Tutorial

Bei-Sheng Su
Professor : Tzi-Dar Chiueh
2019/11/22



臺灣大學

Outline

- Introduction to Verilog
 - Module
 - Value & number
 - Data type
- Ncverilog simulation & nWave tool



臺灣大學

Introduction to Verilog

Module
Value & Number
Data Type



臺灣大學

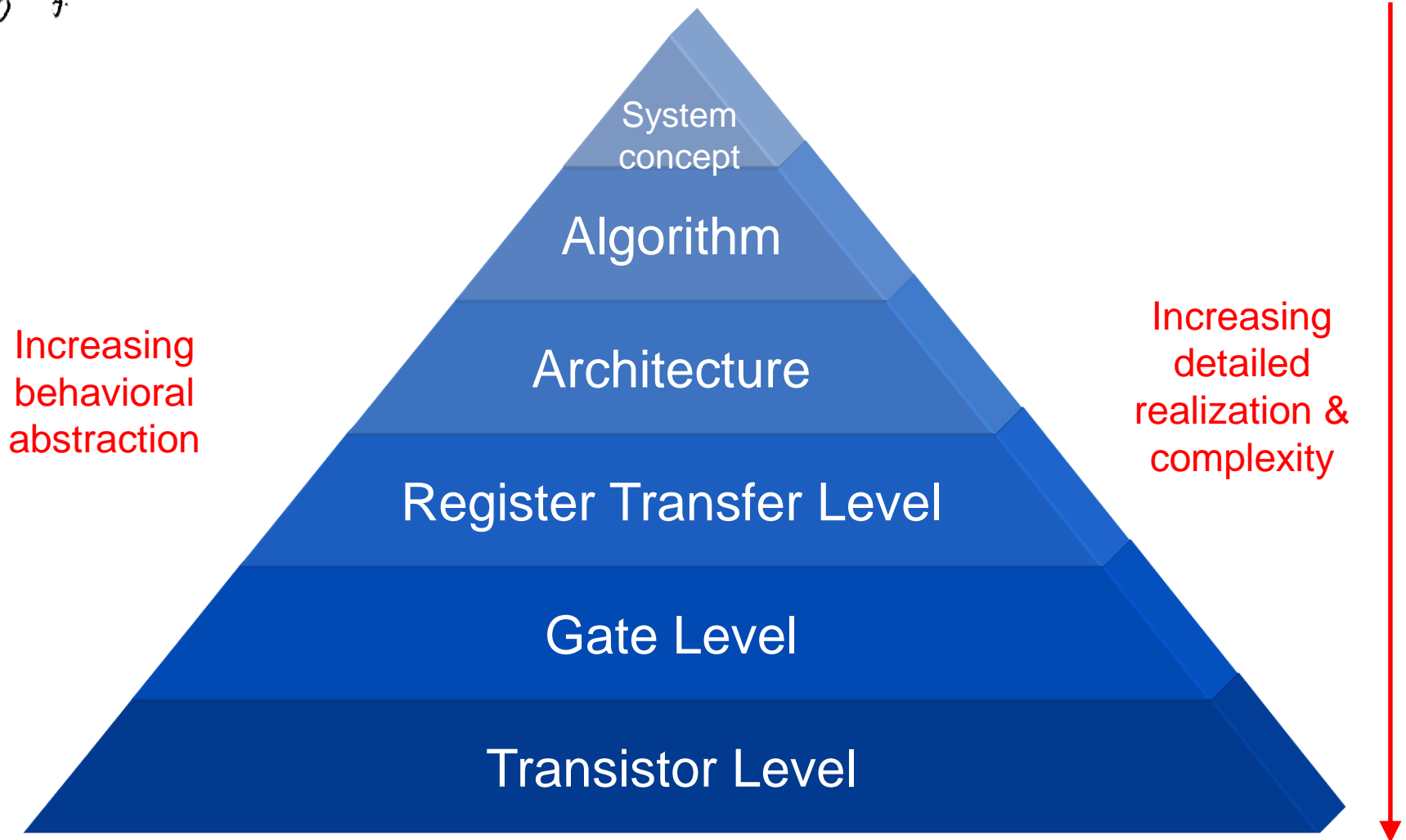
What is Verilog

- Verilog is a Hardware Description Language (HDL)
 - Describe digital electronic system at multiple levels of abstraction
 - Model the timing
 - Express the *concurrency* of the system operation
 - Test the system



臺灣大學

Behavioral Model of Circuits





臺灣大學

Verilog-Supported Levels of Abstraction

- Behavioral

- Structural and procedural like the C programming language
- Describe algorithm level and RTL level Verilog models

- Register Transfer Level (RTL)

- Describe the flow of data between registers and how a design process these data.

- Structural

- Describe gate-level and switch-level circuits.

High



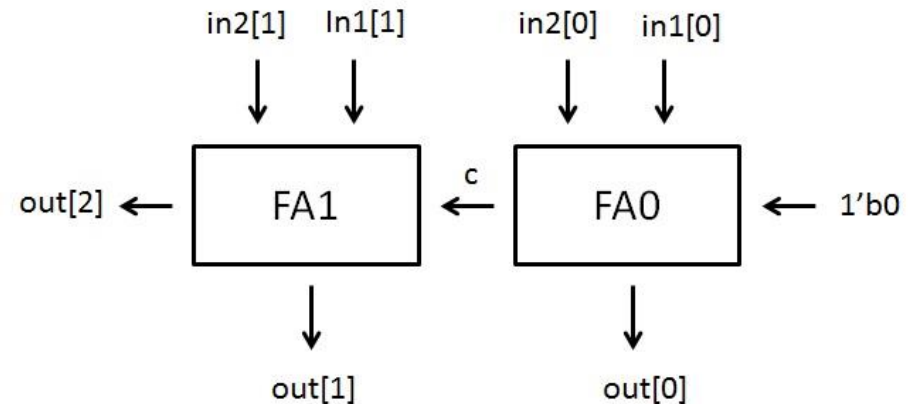
Low



臺灣大學

The Verilog Module

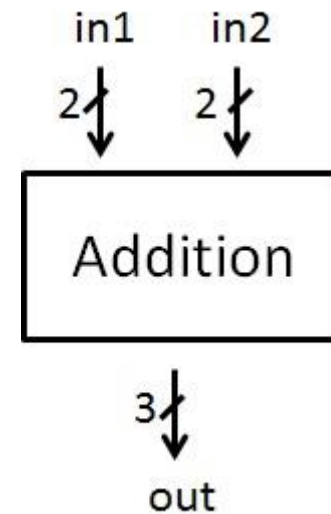
- Basic building blocks .
- Begin with **module**,
end with **endmodule**



```
module <module name>(<port lists>);
```

```
//module description
```

```
endmodule
```

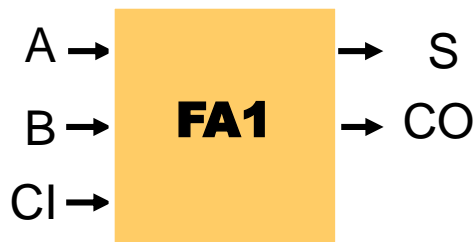




臺灣大學

Module Ports

- Modules communicate through ports
 - input
 - output
 - inout (bidirectional)



```
module FA1(CO,S,A,B,CI);  
    output S,CO;  
    input A,B,CI;    //module description  
endmodule
```




臺灣大學

4-Value Logic System

- 0
 - zero, false, low
- 1
 - one, true, high
- Z
 - high impedance, floating
- X
 - unknown, occurs at un-initialized storage elements or un-resolvable logic conflicts



臺灣大學

Value and Number

- `[[<size>]'<radix>]<value>`
 - Size
 - The size in bits
 - Default size is 32 bits
 - Radix
 - b (binary), o (octal), d (decimal), h (hexadecimal)
 - Default radix is decimal
 - Value
 - Any legal number in selected radix, including “x” and “z”
- Radix and value are case-insensitive



臺灣大學

Value and Number - Examples

`4'b1001` // 4-bit binary
`5'd3` // 5-bit decimal
`12'h7ff` // 12-bit hexadecimal
`3'bx10` // 3-bit binary with unknown MSB
`4'b101x` // 4-bit binary with unknown LSB
`12'hx` // 12-bit unknown
`-8'd6` // phrase as `-(8'd6)`

- underline usage
 - `16'b0001_0101_0001_1111`
 - `32'h12ab_f001`
- X and Z is sign-extended
 - Ex. 12-bit a
 - `a = 'h x; // yields xxx`
 - `a = 'h 3x; // yields 03x`
 - `a = 'h 0x; // yields 00x`



臺灣大學

Data Type Classes

- **wire**
 - **wire** [MSB:LSB] *variables*;
 - input, inout, output are default to be wire.
 - Used to describe combinational circuit!
- **reg**
 - **reg** [MSB:LSB] *variables*
 - Used to describe combinational or sequential circuit.



臺灣大學

Assign a value to wire

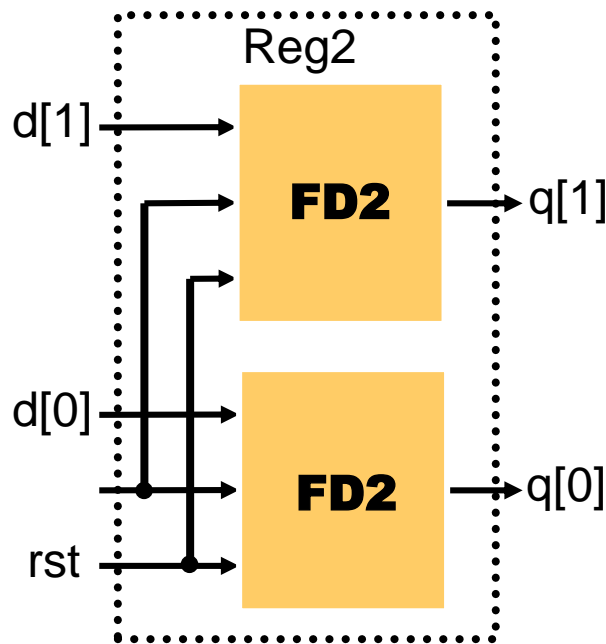
- wire
 - “assign “
wire a;
assign a = 1'b1;
 - Output port
wire a;
wire b;
assign b = 1'b0;
NOT n0(a, b);
- Every wire can be only assigned once!!!
wire a;
wire b;
assign b = 1'b0;
NOT n0(a, b);
assign a = 1'b0; //Wrong!!!



臺灣大學

Module Instances (1/2)

- Create a higher-level system by connecting lower-level components



```
module Reg2 (q, d, clk, rst);  
    output    [1:0] q;  
    input     [1:0] d;  
    input     clk, rst;  
    FD2 f0(q[0], d[0], clk, rst);  
    FD2 f1(.Q(q[1]), .D(d[1]),  
          .CLK(clk), .RESET(rst));  
endmodule
```

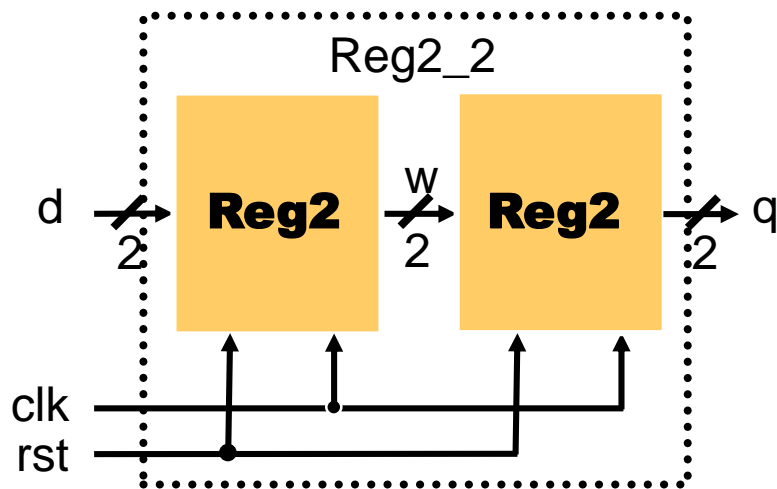
Module name

Instance name



臺灣大學

Module Instances (2/2)



```
module Reg2_2 (q, d, clk, rst);  
  
    output      [1:0] q;  
    input       [1:0] d;  
    input       clk, rst;  
  
    wire        [1:0] w;  
  
    Reg2 r0(w, d, clk, rst);  
    Reg2 r1(q, w, clk, rst);  
  
endmodule
```



臺灣大學

Net Concatenations

Representation	Meaning
$\{b[3:0], c[2:0]\}$	$\{b[3], b[2], b[1], b[0], c[2], c[1], c[0]\}$
$\{a, b[3:0], w, 3'b101\}$	$\{a, b[3], b[2], b[1], b[0], w, 1'b1, 1'b0, 1'b1\}$
$\{4\{w\}\}$	$\{w, w, w, w\}$
$\{b, \{3\{a, b\}\}\}$	$\{b, a, b, a, b, a, b\}$



Standard Cell Library (lib.v)

臺灣大學

- Choose what you need
- Compose your circuit according to I/O connections

```
module AN3(Z,A,B,C);
```

```
    output Z;  
    input A,B,C;
```

```
    // netlist
```

```
    and g1(Z,A,B,C);
```

```
    // specify block, declare local
```

```
    // timing constant
```

```
    specify
```

```
        // delay parameters
```

```
        specparam Tp_A_Z = 0.275;
```

```
        specparam Tp_B_Z = 0.275;
```

```
        specparam Tp_C_Z = 0.275;
```

```
        // path delay (full connection)
```

```
        ( A *> Z ) = ( Tp_A_Z );
```

```
        ( B *> Z ) = ( Tp_B_Z );
```

```
        ( C *> Z ) = ( Tp_C_Z );
```

```
    endspecify
```

```
endmodule
```



Standard Cell Library (lib.v) - 2/2

臺灣大學

- IV // not
- AN3
- AN4
- AN2
- EN // xnor
- EN3
- EO // xor
- EO3
- FA1 // full adder
- FD1 // DFF
- FD2
- ND2 // nand
- ND3
- ND4
- NR2 // nor
- NR3
- OR2 // or
- OR3
- OR4
- HA1 // half adder
- MUX21H // 2-to-1 MUX



臺灣大學

Notification

- In this HW, all the logic operation **MUST** consist of standard cell. You can **NOT** use logic operators.

~~wire a, b, c;
assign a = b & c;~~

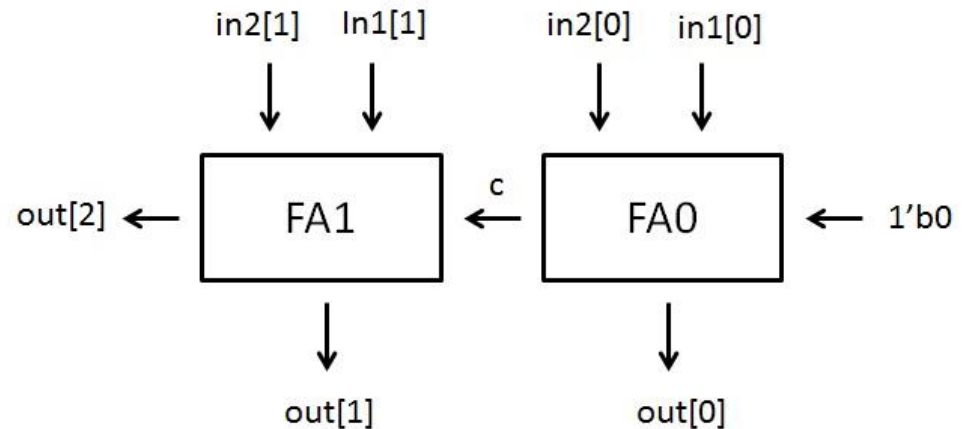
wire a, b, c;
AN2 an(a, b, c);



臺灣大學

Example

```
module ADDER (out, in1, in2);  
    output [2:0] out;  
    input [1:0] in1, in2;  
    wire c;
```

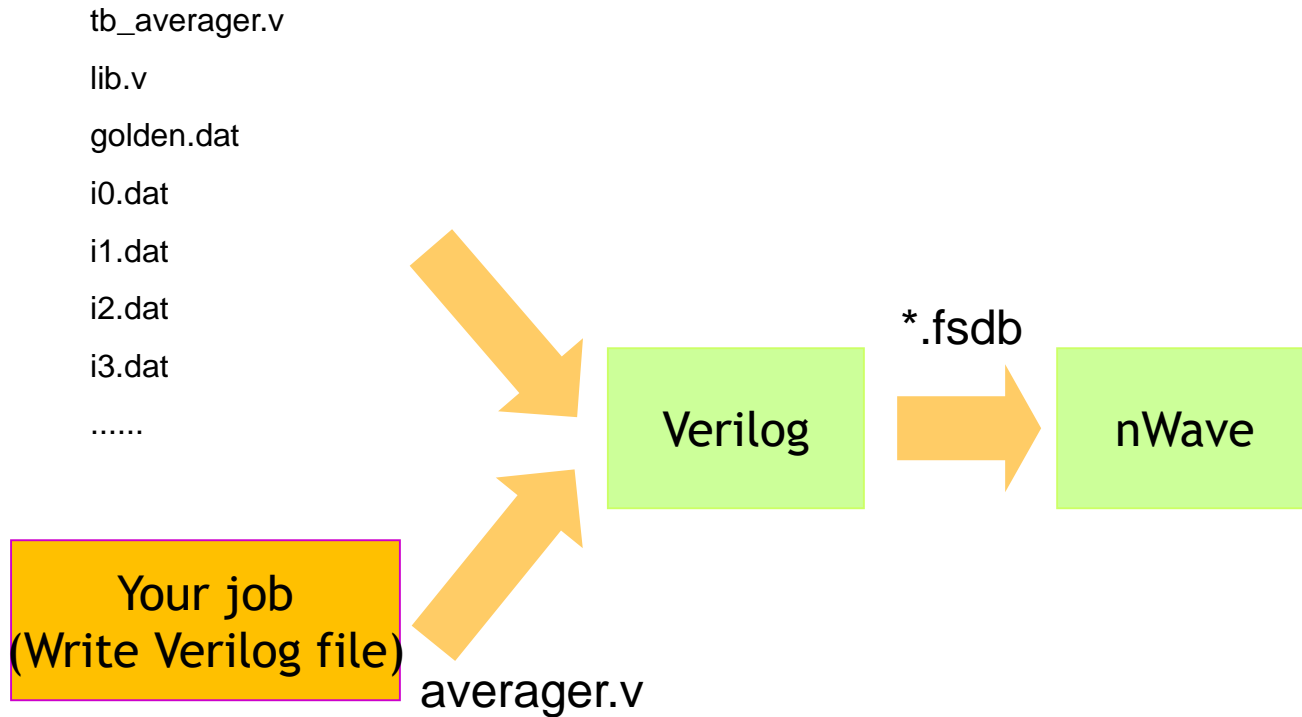


```
    FA1 fa0(c, out[0], in1[0], in2[0], 1'b0);  
    FA1 fa1(out[2], out[1], in1[1], in2[1], c);  
endmodule
```



臺灣大學

Flow chart

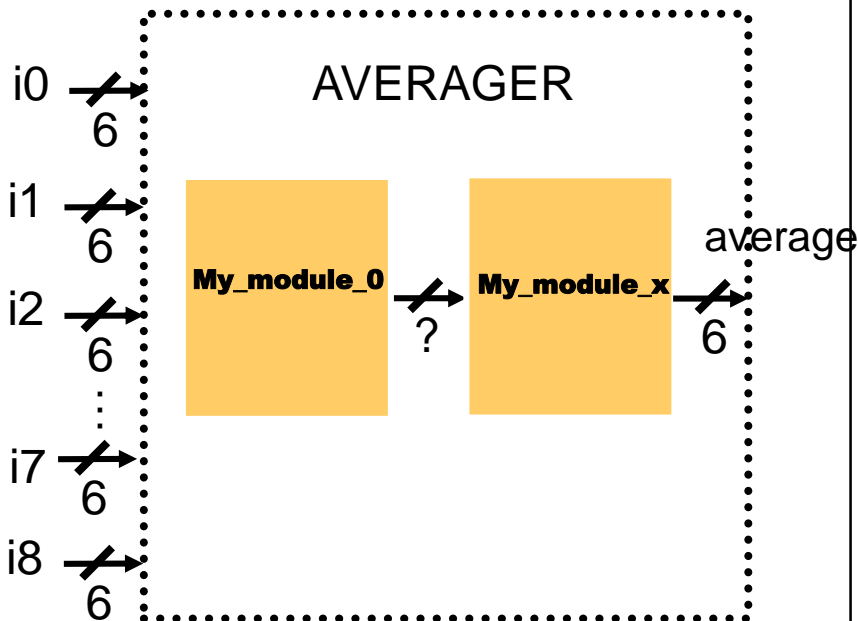


All these files should be placed under the same folder



臺灣大學

Your Job



```
module averager(average, i0, i1, i2, i3,
i4, i5, i6, i7, i8);

    input  [5:0] i0, i1, i2, i3, i4 , i5, i6, i7, i8;
    output [5:0] average;

    // Write your design here

    wire      [?:0] ...;

    My_module_0 M0(?, ?, ... , ?);

    .....
    My_module_x Mx(?, ?, ... , ?);

endmodule
```



臺灣大學

Ncverilog Simulation & nWave Tool

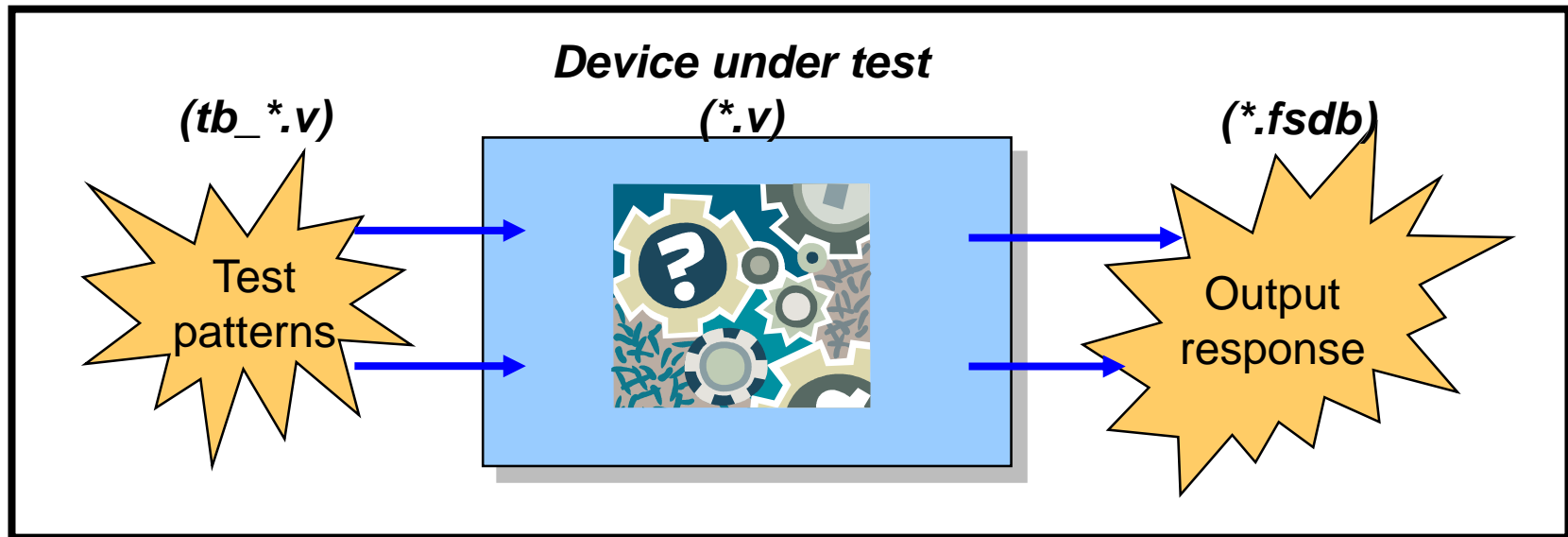


Test and verification your circuit

臺灣大學

- By applying input patterns and observing output responses

Testbench





臺灣大學

Compile and debug (1/3)

- Source
 - `source /usr/cadence/cshrc`
- Include the testbench & lib.v files to run simulation
 - `ncverilog +access+r tb_averager.v averager.v lib.v`



臺灣大學

Compile and debug (2/3)

```
[r07001@cad29 ~/HW3_2019]$ ncverilog +access+r tb averager.v averager.v lib.v
ncverilog: 15.20-s039: (c) Copyright 1995-2017 Cadence Design Systems, Inc.
file: tb_averager.v
    module worklib.tb_averager:v
        errors: 0, warnings: 0
file: averager.v
```

```
-----
START!!! Simulation Start .....
```

```
-----
[CORRECT] Your output in      0 ~      999 is correct.
[CORRECT] Your output in    1000 ~    1999 is correct.
[CORRECT] Your output in    2000 ~    2999 is correct.
[CORRECT] Your output in    3000 ~    3999 is correct.
[CORRECT] Your output in    4000 ~    4999 is correct.
[CORRECT] Your output in    5000 ~    5999 is correct.
[CORRECT] Your output in    6000 ~    6999 is correct.
[CORRECT] Your output in    7000 ~    7999 is correct.
[CORRECT] Your output in    8000 ~    8999 is correct.
[CORRECT] Your output in    9000 ~    9999 is correct.
-----
```

```
Your score in correctness part is 40.Accuracy is 100.00 %
Congratulations!!! All data have been generated successfully!
```

```
----- The test result is ..... PASS -----
```

```
Your average latency is 5.958700 ns
```

```
Simulation complete via $finish(1) at time 52671 NS + 0
./tb_averager.v:167                               $finish;
ncsim> exit
```



臺灣大學

Compile and debug (3/3)

```
[r07001@cad29 ~/HW3_2019]$ ncverilog +access+r tb averager.v averager.v lib.v
ncverilog: 15.20-s039: (c) Copyright 1995-2017 Cadence Design Systems, Inc.
file: tb_averager.v
    module worklib.tb_averager:v
        errors: 0, warnings: 0
file: averager.v
```

```
[ERROR] The golden in      268 is 0x3d , but your output is 0x3c
[ERROR] The golden in      276 is 0x0d , but your output is 0x0c
[ERROR] The golden in      280 is 0x3d , but your output is 0x3c
[ERROR] The golden in      282 is 0x01 , but your output is 0x00
```

There are more than 100 errors!! So the following error message will be pruned!!

```
[ERROR]Your output in      0 ~      999 has      396 errors.
[ERROR]Your output in    1000 ~    1999 has      407 errors.
[ERROR]Your output in    2000 ~    2999 has      399 errors.
[ERROR]Your output in    3000 ~    3999 has      415 errors.
[ERROR]Your output in    4000 ~    4999 has      370 errors.
[ERROR]Your output in    5000 ~    5999 has      412 errors.
[ERROR]Your output in    6000 ~    6999 has      388 errors.
[ERROR]Your output in    7000 ~    7999 has      377 errors.
[ERROR]Your output in    8000 ~    8999 has      400 errors.
[ERROR]Your output in    9000 ~    9999 has      412 errors.
```

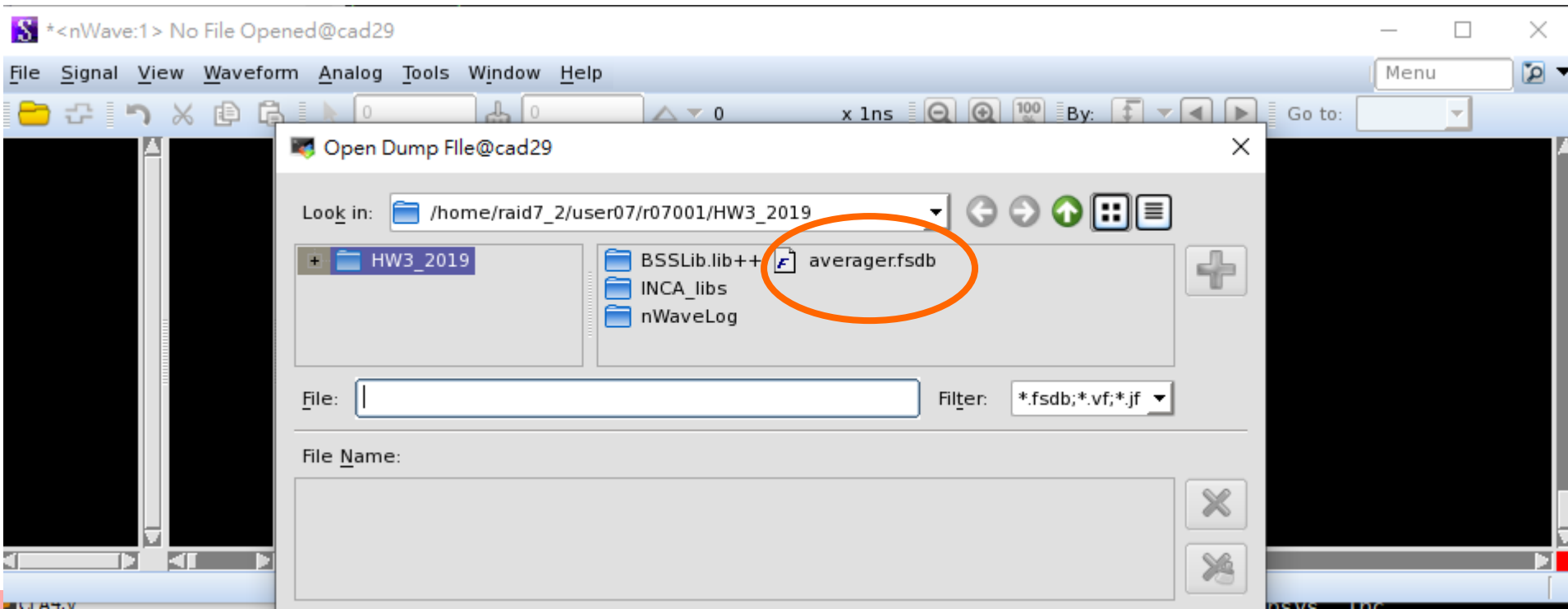
Error!!! There is something wrong with your code ...!
There are 3976 errors ! Your score in correctness part is 24.10. Accuracy is 60.24 %
-----The test result isFAIL -----



臺灣大學

Execute nWave & Open *.fsdb

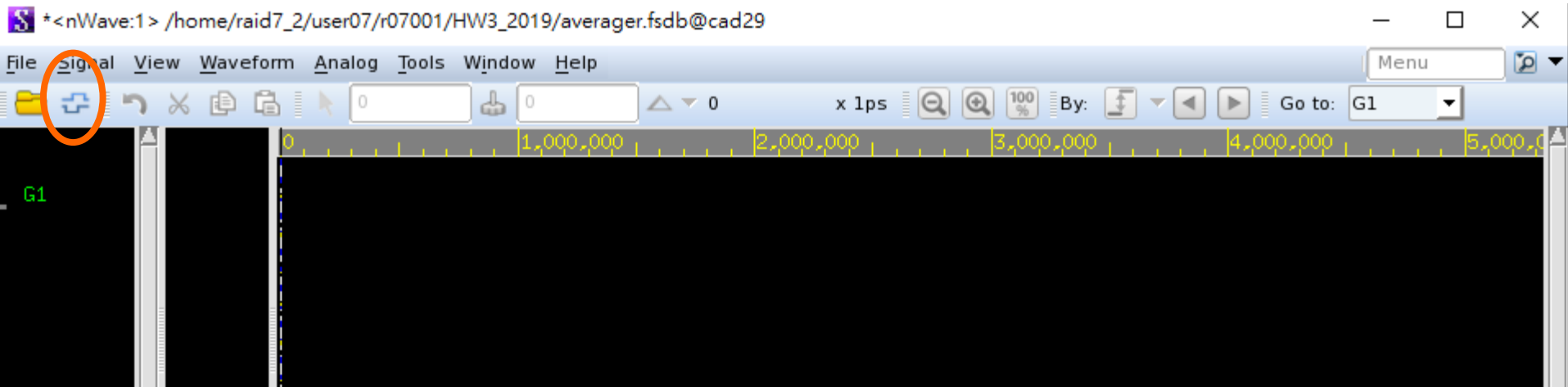
- Execute : *nWave* &
- Open waveform file:
 - File -> Open -> averager.fsdb





臺灣大學

Get signals



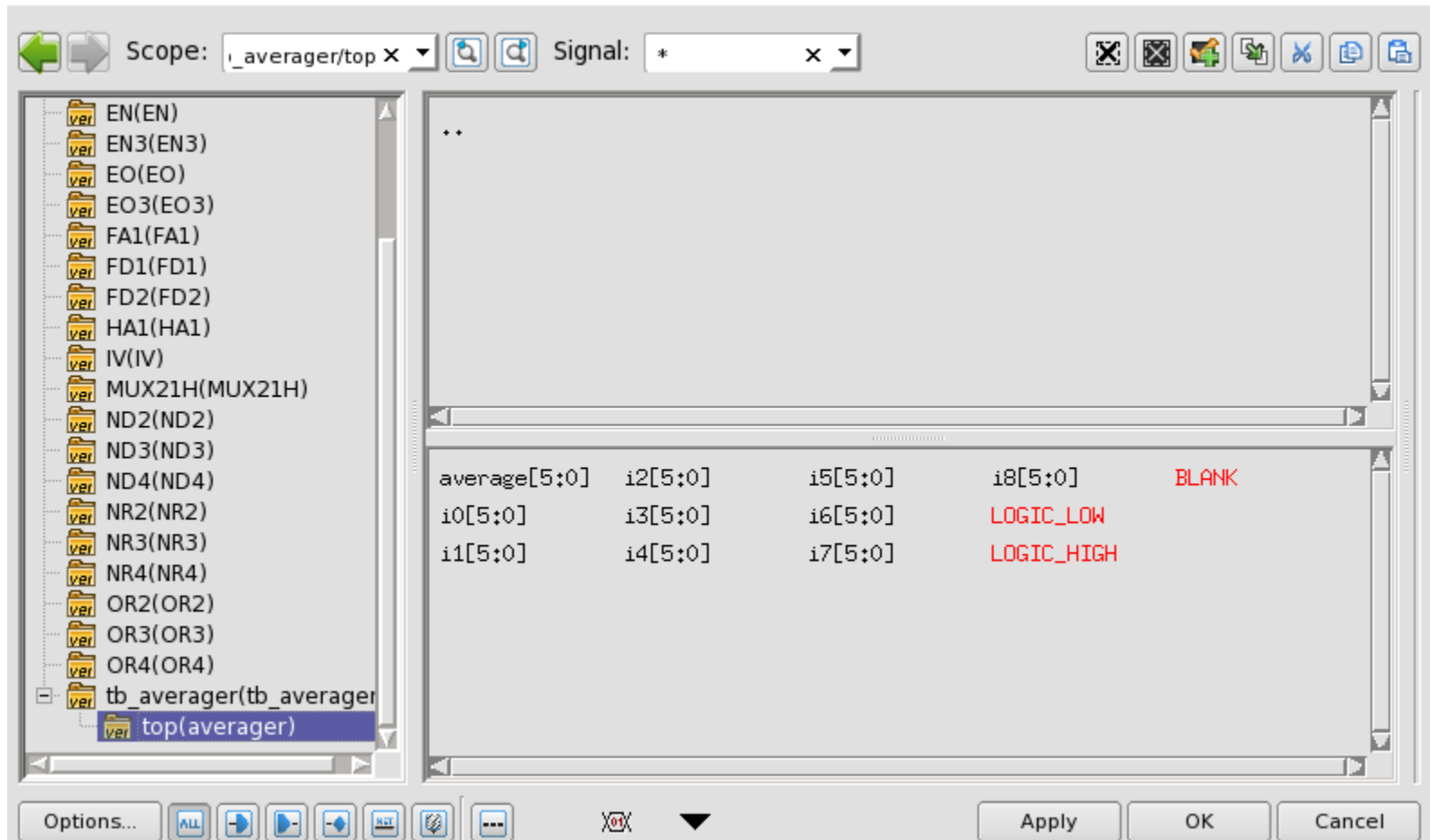


Find Top Module & Choose signals

臺灣大學

Get Signals@cad29

X





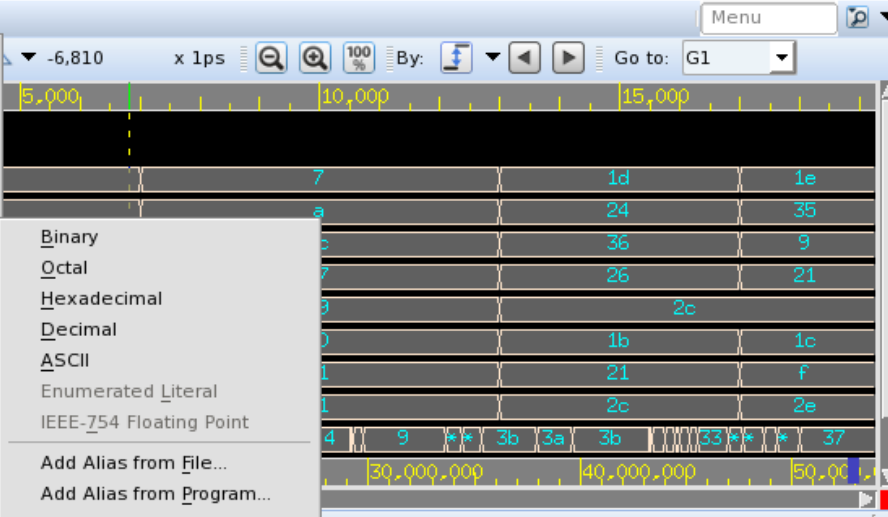
Change the radix to be decimal

臺灣大學

*<nWave:1> /home/raid7_2/user07/r07001/HW3_2019/averager.fsdb@cad29

File Signal View Waveform Analog Tools Window Help

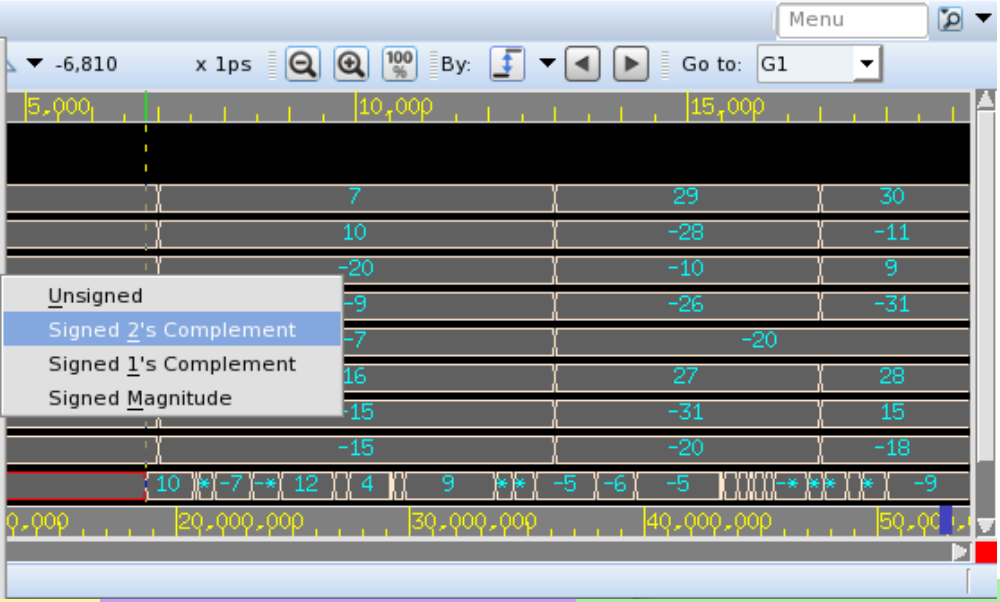
- Auto Update
- Spacing...
- Height...
- Fit Waveform Ctrl+L
- Color/Pattern... C
- Signal Value Radix
- Signal Value Notation
- Analog Waveform
- Digital Waveform
- Invert Waveform
- Property
- Classic Transaction
- Classic Message
- Go To



*<nWave:1> /home/raid7_2/user07/r07001/HW3_2019/averager.fsdb@cad29

File Signal View Waveform Analog Tools Window Help

- Auto Update
- Spacing...
- Height...
- Fit Waveform Ctrl+L
- Color/Pattern... C
- Signal Value Radix
- Signal Value Notation
- Analog Waveform
- Digital Waveform
- Invert Waveform
- Property
- Classic Transaction
- Classic Message
- Go To
- Set Search Value...





臺灣大學

Reminder

- Due to 2019/12/13 13:20
- Any further questions, please contact...
 - r07943001@ntu.edu.tw
 - sam0108w@gmail.com