Debugging Tips

Outline

- Module Instantiation
- Print out signal
- Waveform
- Breakpoint

Outline

- Module Instantiation
- Print out signal
- Waveform
- Breakpoint

Module Instantiation

- Parameter value assignment by order
  - ProgramCounter PC(

```
                    clk_i,
                    rst_i,
                    pc_in,
                    pc_out);
```

- Parameter value assignment by name
  - ProgramCounter PC(

```
                   .clk_i(clk_i),
                   .pc_in_i(pc_in),
                   .rst_i(rst_i),
                   .pc_out_o(pc_out));
```

Outline

- Module Instantiation
- Print out signal
- Waveform
- Breakpoint

Values of signals

- $monitor ("%0dns :\$monitor: a=%b b=%b" , $stime, a, b);
  - Print parameters every time as one of its parameters changes.
- $display ("%0dns :\$display: a=%b b=%b" , $stime, a, b);
  - Like printf in C, only print parameters once.

Values of signals

- $fwrite(fp, "%0dns :\$fwrite : a=%b b=%b\n", $stime, a, b);
  - Like fprintf in C.
  - Used with $fopen and $fclose.
  - $fdisplay is similar with $fwrite but append “\n” automatically

Values of signals

```
ProgramCounter PC(
        .clk_i(clk_i),
        .rst_i (rst_i),
        .pc_in_i(pc_in) ,
        .pc_out_o(pc_out));
```

- $display(“a = %d, b = %d”,
  PC.pc_in_i, PC.pc_out_o);

- $monitor(“a = %d, b = %d”,
  PC.pc_in_i, PC.pc_out_o);

Values of signals

- $monitor
  - 1ns :$monitor: a=0 b=1
- $display
  - 2ns :$display: a=1 b=0
- $fwrite
  - In a certain text file.
  - 1ns :$fwrite : a=0 b=1

Comparison

- $display displays the result of simulation only when the display task occurs in your code.
- $monitor continuously MONITORS its variables, when a variable changes its value, monitor displays the results.
- $fwrite writes data into a text file.

Outline

- Module Instantiation
- Print out signal
- Waveform
- Breakpoint

Waveform

- Add following code:

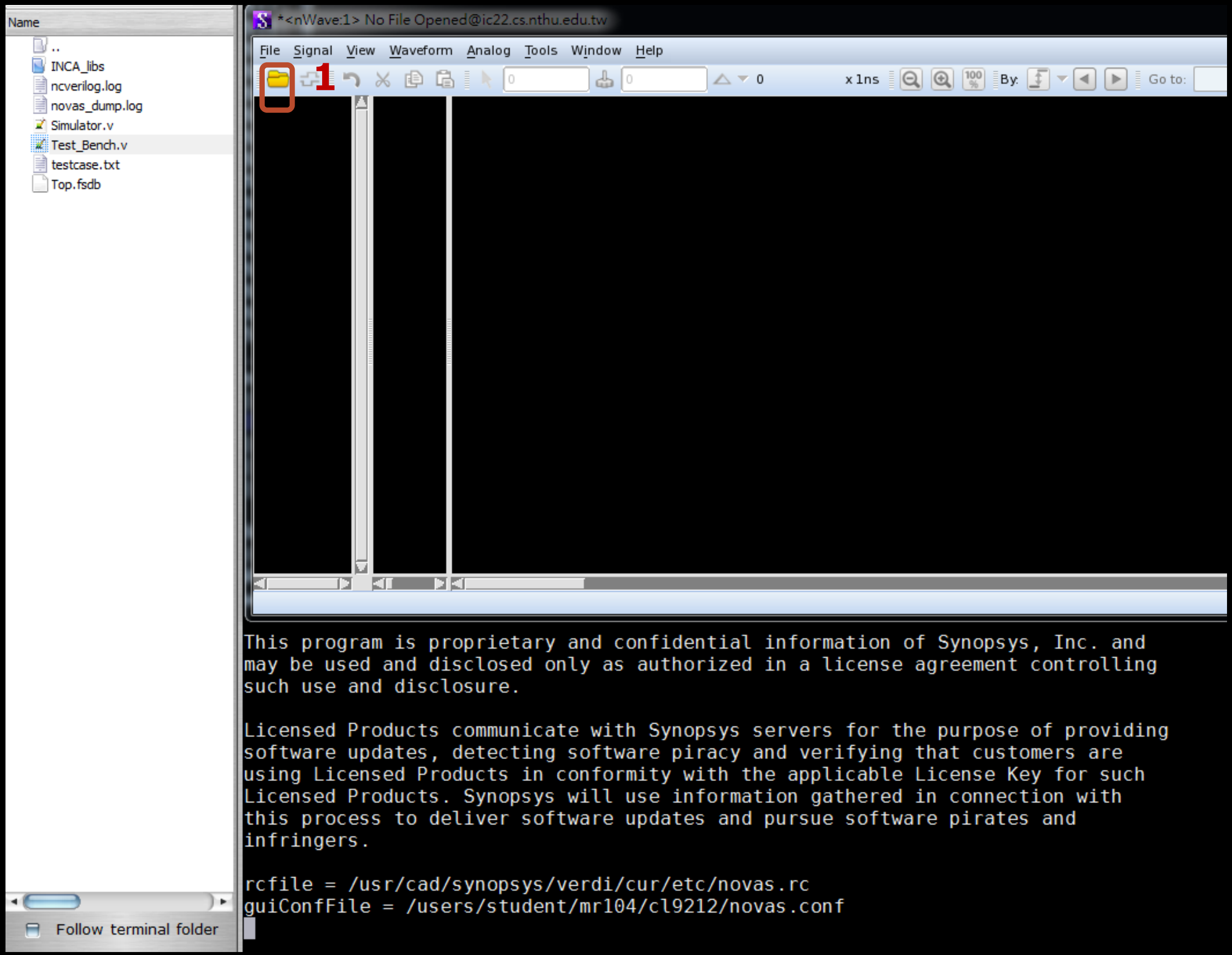
```
initial begin
        $fsdbDumpfile("Top.fsdb");
        /*waveform file*/
        $fsdbDumpvars(0, "+mda");
        /*also dump 2D register*/
end
```

Waveform

- Make sure you connect workstation with ssh –X icXX
  - -X: set IP used to display gui to your computer
- Execute NC Verilog with parameter "+access+r"
  - $ ncverilog Simulator.v Test_Bench.v +access+r
- Use nWave to view waveform

Waveform

- $nWave &

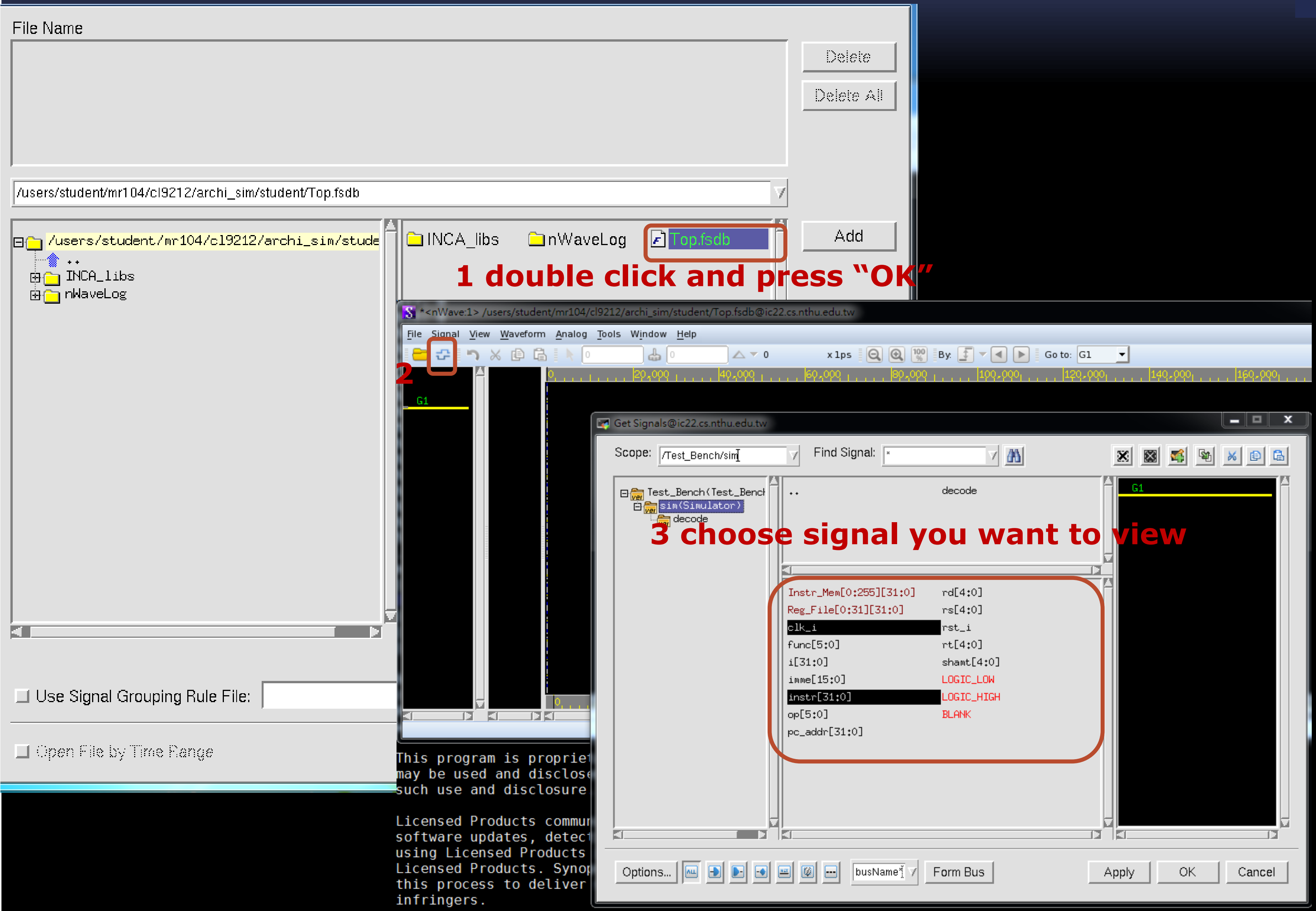


```
This program is proprietary and confidential information of Synopsys, Inc. and
may be used and disclosed only as authorized in a license agreement controlling
such use and disclosure.

Licensed Products communicate with Synopsys servers for the purpose of providing
software updates, detecting software piracy and verifying that customers are
using Licensed Products in conformity with the applicable License Key for such
Licensed Products. Synopsys will use information gathered in connection with
this process to deliver software updates and pursue software pirates and
infringers.

rcfile = /usr/cad/synopsys/verdi/cur/etc/novas.rc
guiConfFile = /users/student/mr104/cl9212/novas.conf
```

Waveform



File Name

/users/student/mr104/cl9212/archi_sim/student/Top.fsdb

/users/student/mr104/cl9212/archi_sim/stude

- ..
- INCA_libs
- nWaveLog

INCA_libs nWaveLog Top.fsdb

1 double click and press “OK”

□ Use Signal Grouping Rule File:

□ Open File by Time Range

*<nWave:1> /users/student/mr104/cl9212/archi_sim/student/Top.fsdb@ic22.cs.nthu.edu.tw

2

Get Signals@ic22.cs.nthu.edu.tw

Scope: /Test_Bench/sim

Find Signal: *

- Test_Bench(Test_Bench
  - sim(Simulator)
    - decode

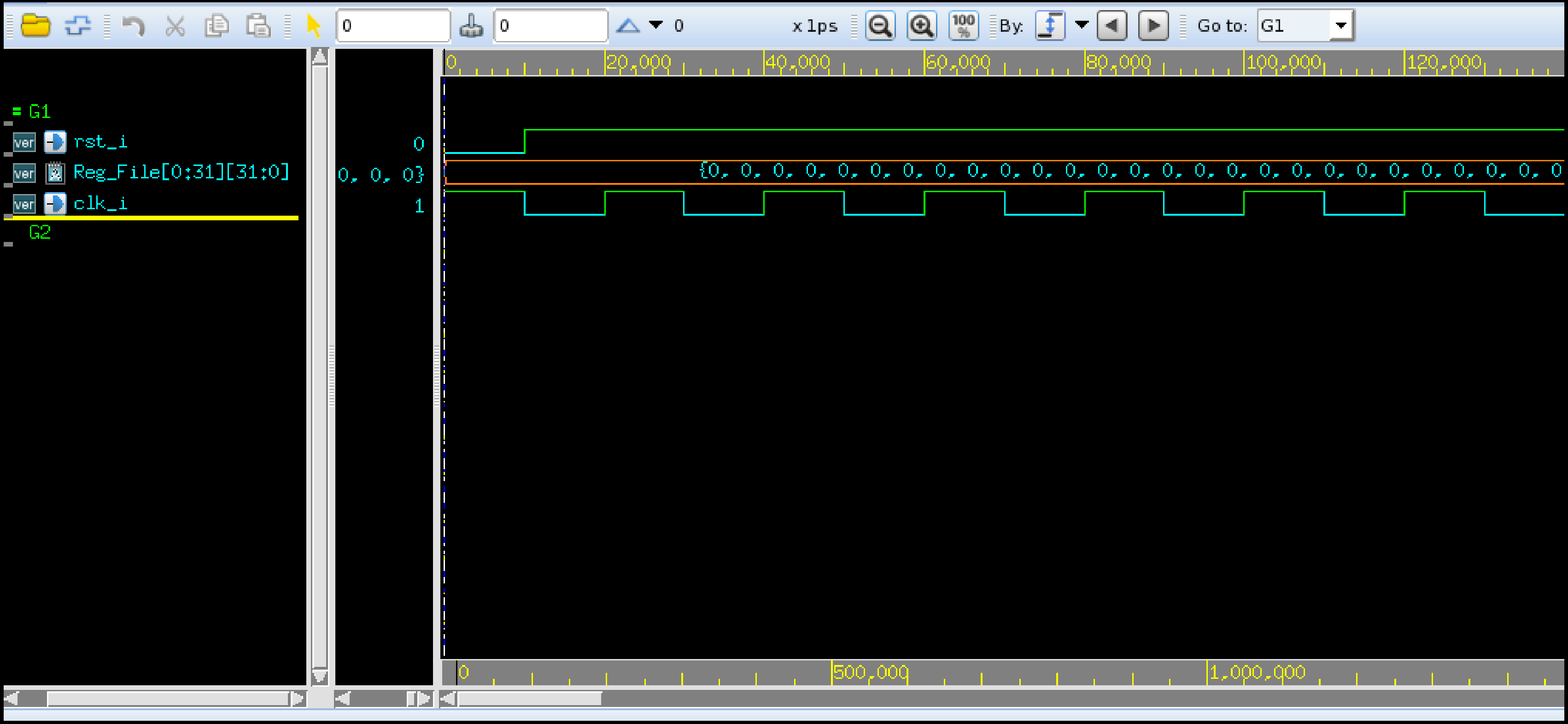
3 choose signal you want to view

```
Instr_Mem[0:255][31:0]   rd[4:0]
Reg_File[0:31][31:0]     rs[4:0]
clk_i                    rst_i
func[5:0]                rt[4:0]
i[31:0]                  shamt[4:0]
imme[15:0]               LOGIC_LOW
instr[31:0]              LOGIC_HIGH
op[5:0]                  BLANK
pc_addr[31:0]
```

```
This program is propriet
may be used and disclose
such use and disclosure

Licensed Products commun
software updates, detect
using Licensed Products
Licensed Products. Synop
this process to deliver
infringers.
```

Waveform



G1
ver rst_i 0
ver Reg_File[0:31][31:0] 0, 0, 0}
ver clk_i 1
G2

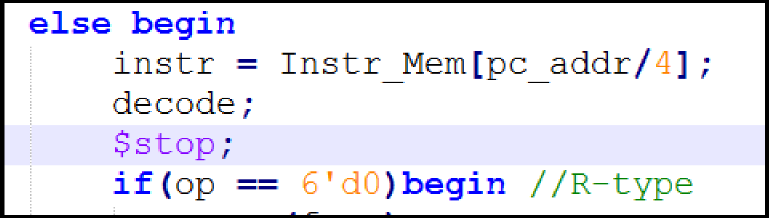
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Outline

- Module Instantiation
- Print out signal
- Waveform
- Breakpoint

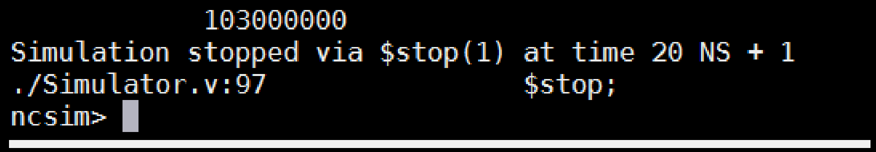
Breakpoint

- Insert “$stop;” where you want to set breakpoint



```
else begin
    instr = Instr_Mem[pc_addr/4];
    decode;
    $stop;
    if(op == 6'd0)begin //R-type
```

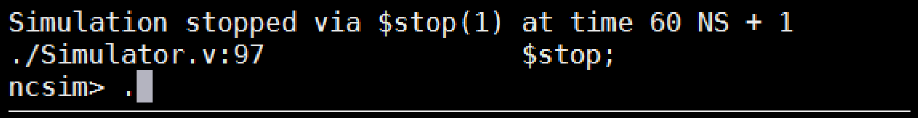
- It will stop simulation when it encounter $stop



```
            103000000
Simulation stopped via $stop(1) at time 20 NS + 1
./Simulator.v:97                $stop;
ncsim> 
```

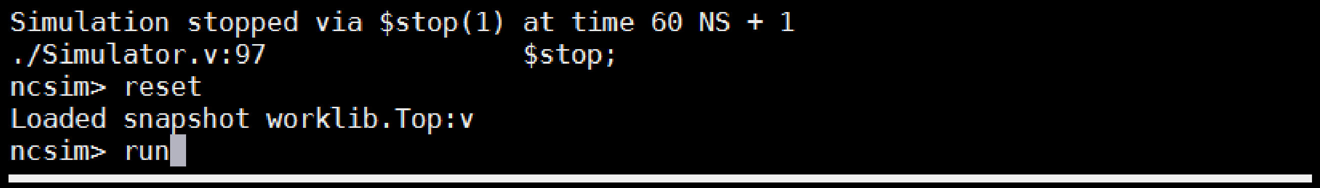
Breakpoint

- Continue simulation by “.” or “run”



```
Simulation stopped via $stop(1) at time 60 NS + 1
./Simulator.v:97                $stop;
ncsim> .
```

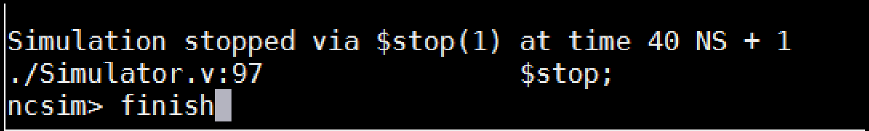
- You can also restart from the beginning by “reset” and type “.” or “run” to start simulation



```
Simulation stopped via $stop(1) at time 60 NS + 1
./Simulator.v:97                $stop;
ncsim> reset
Loaded snapshot worklib.Top:v
ncsim> run
```

Breakpoint

- "finish" or "exit" to exit simulation



```
Simulation stopped via $stop(1) at time 40 NS + 1
./Simulator.v:97                  $stop;
ncsim> finish
```