

# Verilog 설계 언어 초급 (실습)

2018.8.7~8

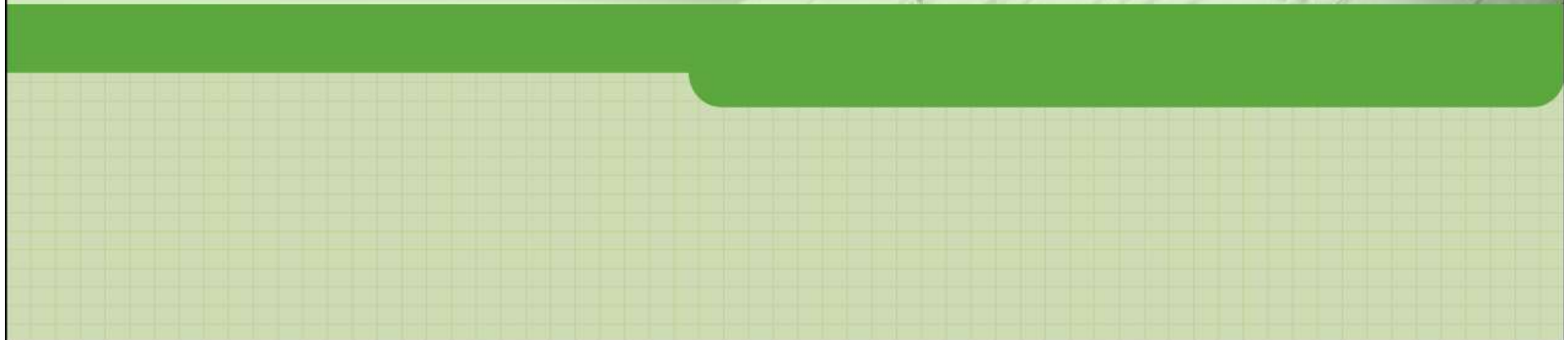
송재훈

# Contents

1. Unix 사용법
2. NC-Verilog를 이용한 simulation
3. Simvision을 이용한 파형 분석
4. Design Vision을 이용한 합성
5. Labs

**2016 IDEC**

# Unix 사용방법



## Unix 사용방법

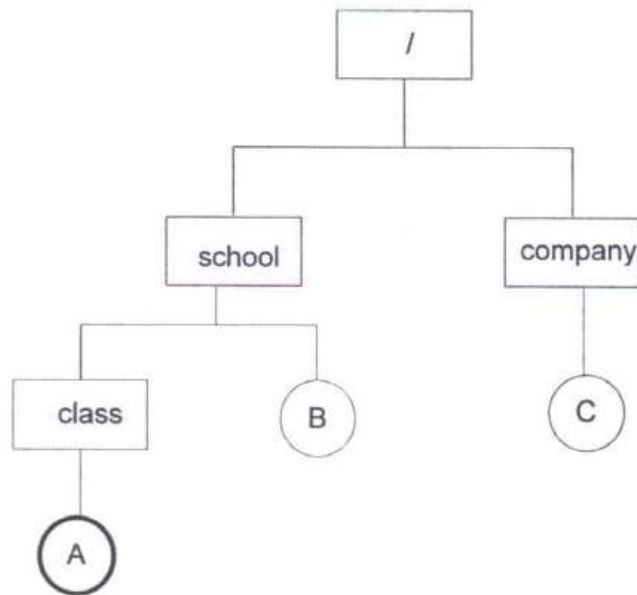
### • 파일 유틸리티

- ◆ `mv <oldFileName> <newFileName>`
  - 파일이나 디렉토리의 이름변경. 현재 파일의 디렉토리 변경
- ◆ `pwd`
  - 현재의 작업 디렉토리가 무엇인지 출력
- ◆ `cd <directoryName>`
  - 해당 디렉토리로 이동. 디렉토리 이름을 주어지지 않은 경우엔 홈 디렉토리로 이동
- ◆ `cp <oldFileName> <newFileName>` (파일 복사)
- ◆ `rmdir <directoryName>` (디렉토리 제거)
- ◆ `mkdir <directoryName>` (디렉토리를 만들기)
- ◆ `rm <fileName>` (파일 삭제)

## Unix 사용방법

### • 절대 경로와 상대 경로

◆ 현재 위치가 **class** 디렉토리에 있다고 가정할 때



절대경로	상대경로
/school/class/A	A
/school/B	../B
/company/C	../../company/C

### • 디렉토리 용보기: **ls [-a] [-l]**



## Unix 사용방법

- 텍스트 에디터 종류

- 1) 명령어 **vi**

- 2) 명령어 **gvim**

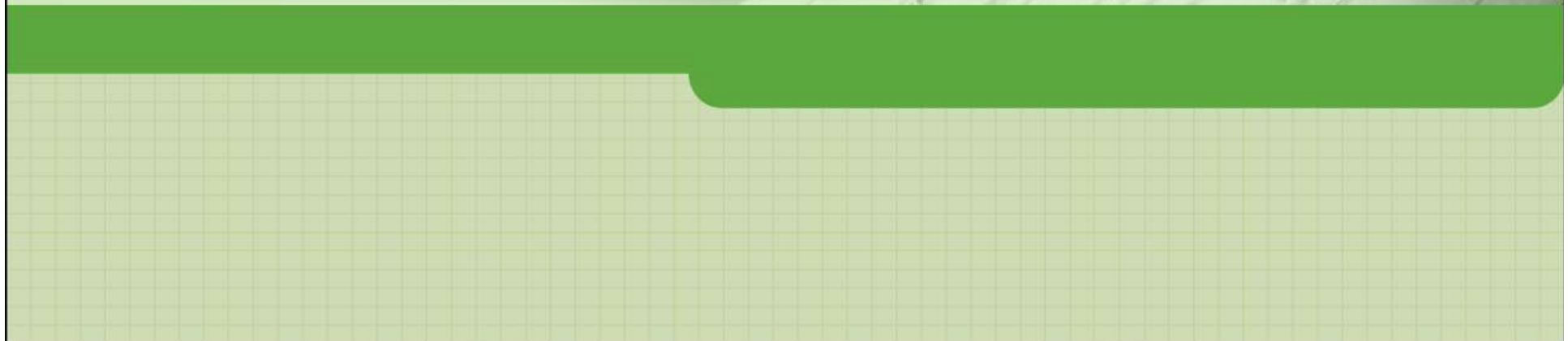
- **GUI** 환경사용

- 3) 명령어 **gedit**

- **GUI** 환경사용

**2016 IDEC**

# **NC-Verilog를 이용한 simulation**

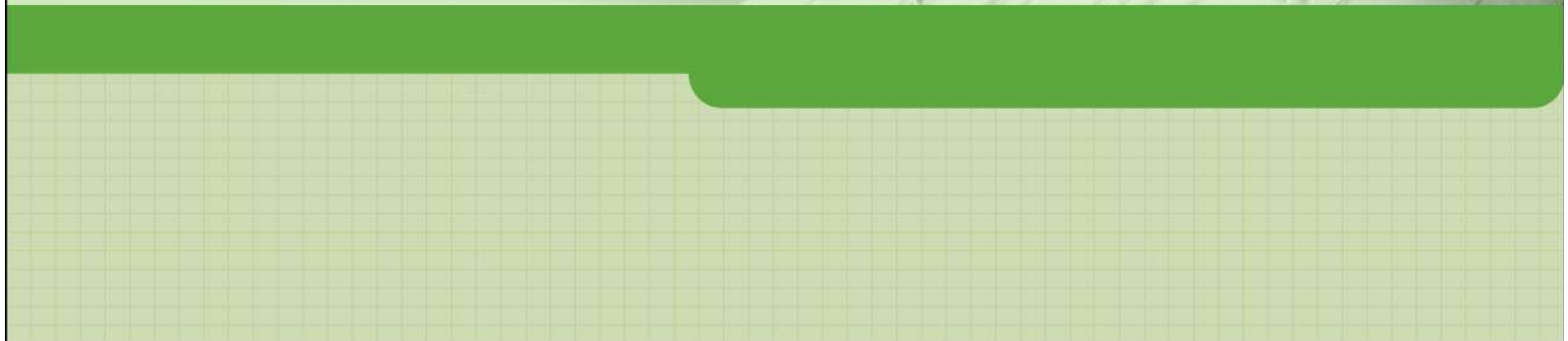


## NC-Verilog를 이용한 simulation

- NC-Verilog로 디자인을 compile 및 simulation 하는 방법
  - 1) 정식 명령어: `ncverilog +access+rwc 파일명(*.v)`
  - 2) 실습환경에서 설정된 alias 명령어: `nc` 파일명(\*.v)



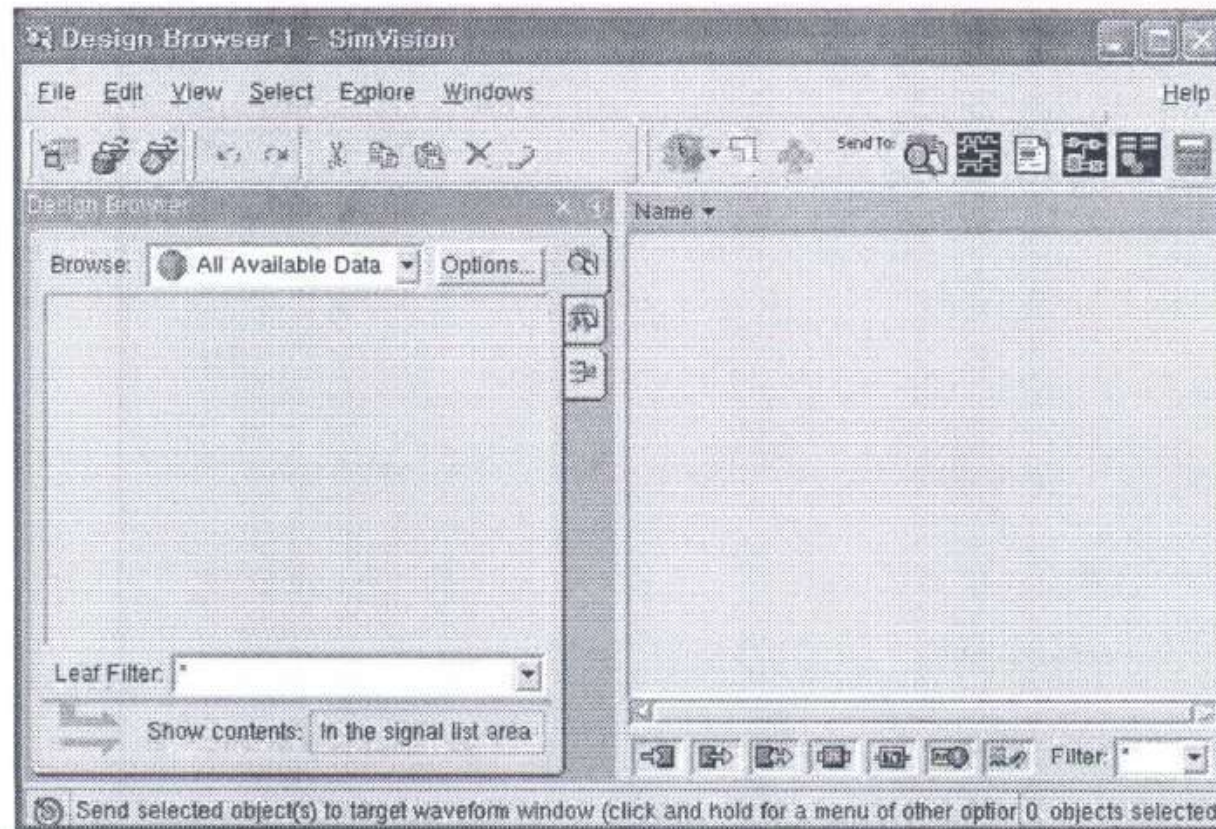
# Simvision을 이용한 파형 분석



## Simvision을 이용한 파형 분석

### • Design Browser Window

- ◆ 프롬프트 상에서 **simvision**을 타이핑하고 **Enter**를 누르면 다음과 같은 window가 뜬다. 실행위치: **user의 lab# 디렉토리**

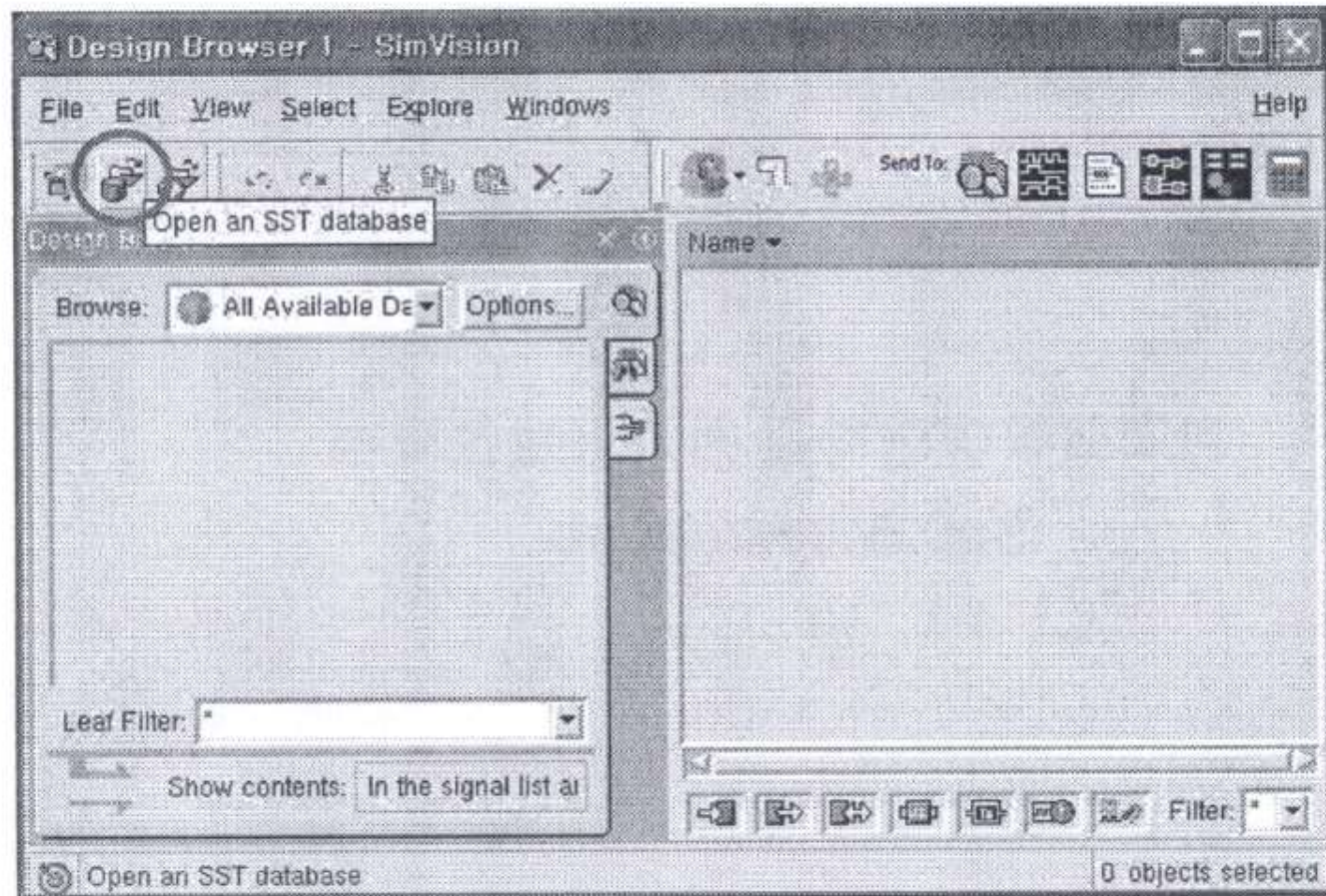




## Simvision을 이용한 파형 분석

- Design Browser Window

◆ database (\*.trn, \*.dsn, \*.vcd) 열기를 선택한다. (첫 번째 방법)

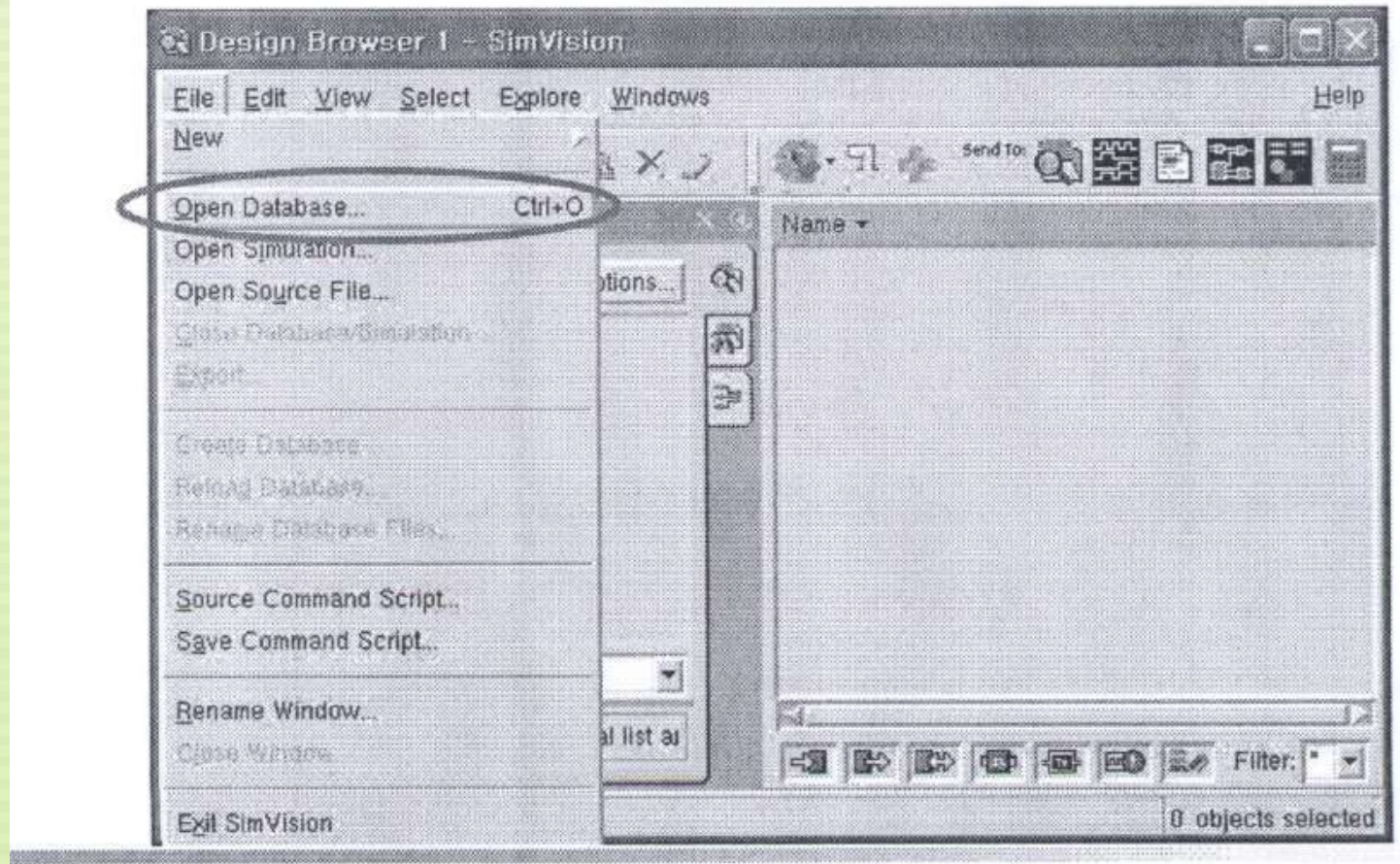




## Simvision을 이용한 파형 분석

- Design Browser Window

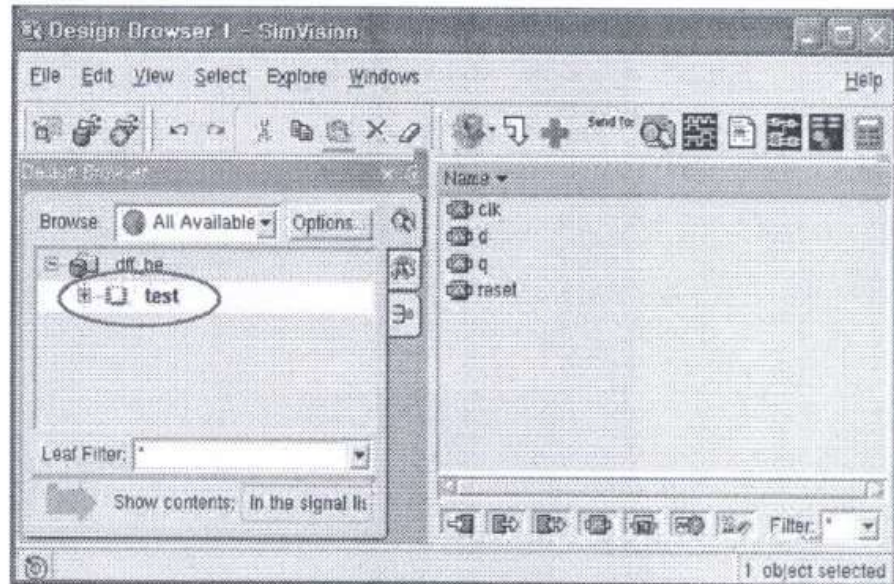
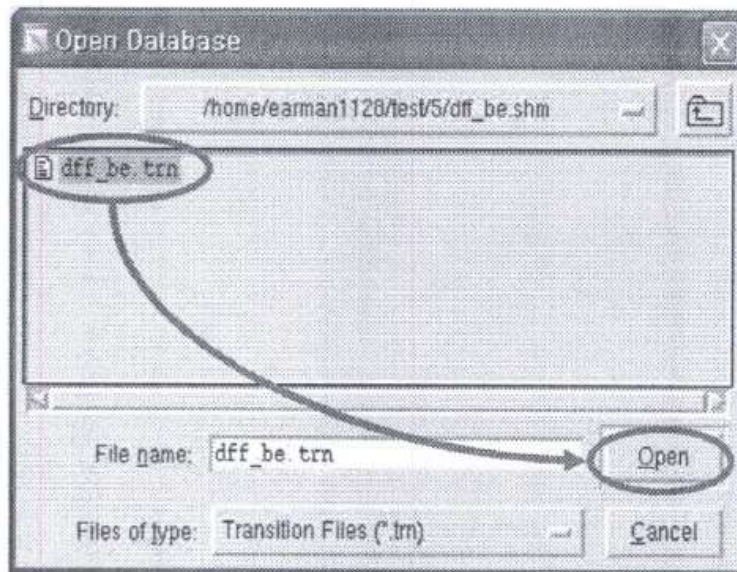
◆ database (\*.trn, \*.dsn, \*.vcd) 열기를 선택한다. (두 번째 방법)



## Simvision을 이용한 파형 분석

### • Design Browser Window

◆ \*.trn(transition files)이나 \*.dsn(design files) 또는 \*.vcd (vcd files)를 선택

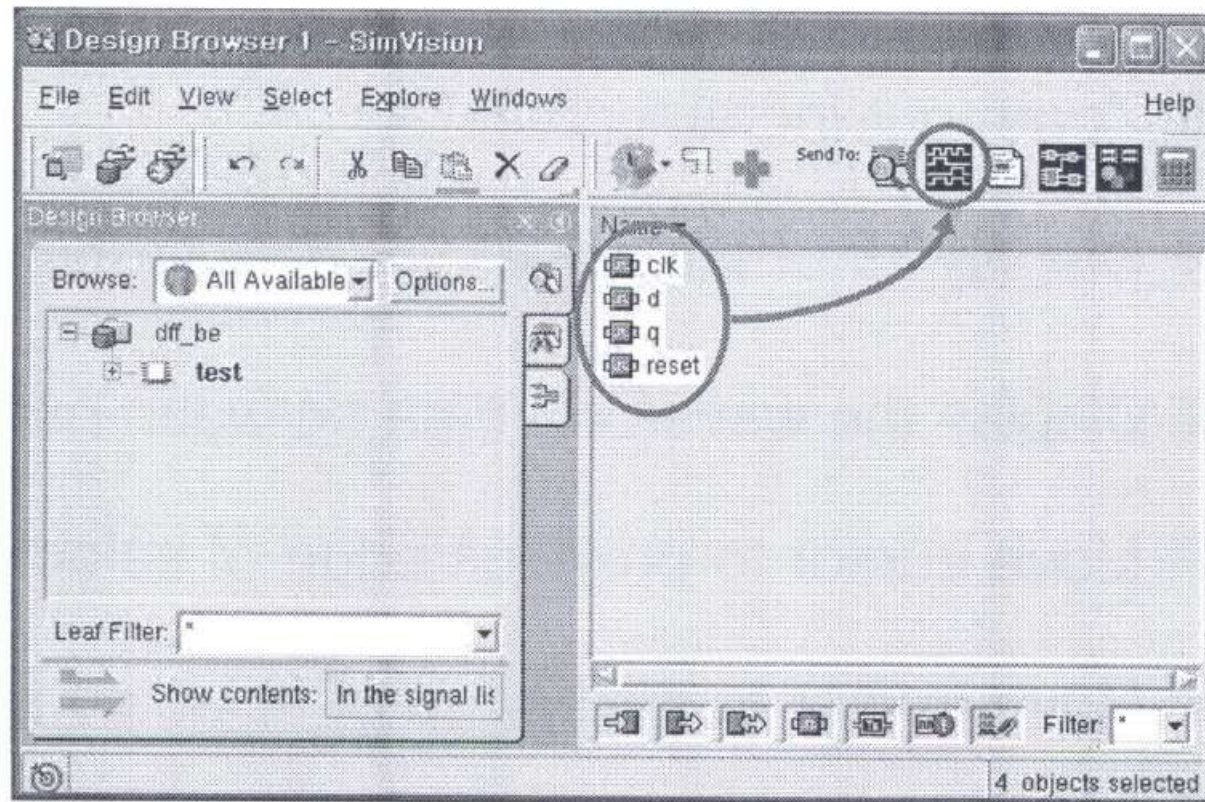




## Simvision을 이용한 파형 분석

### • Design Browser Window

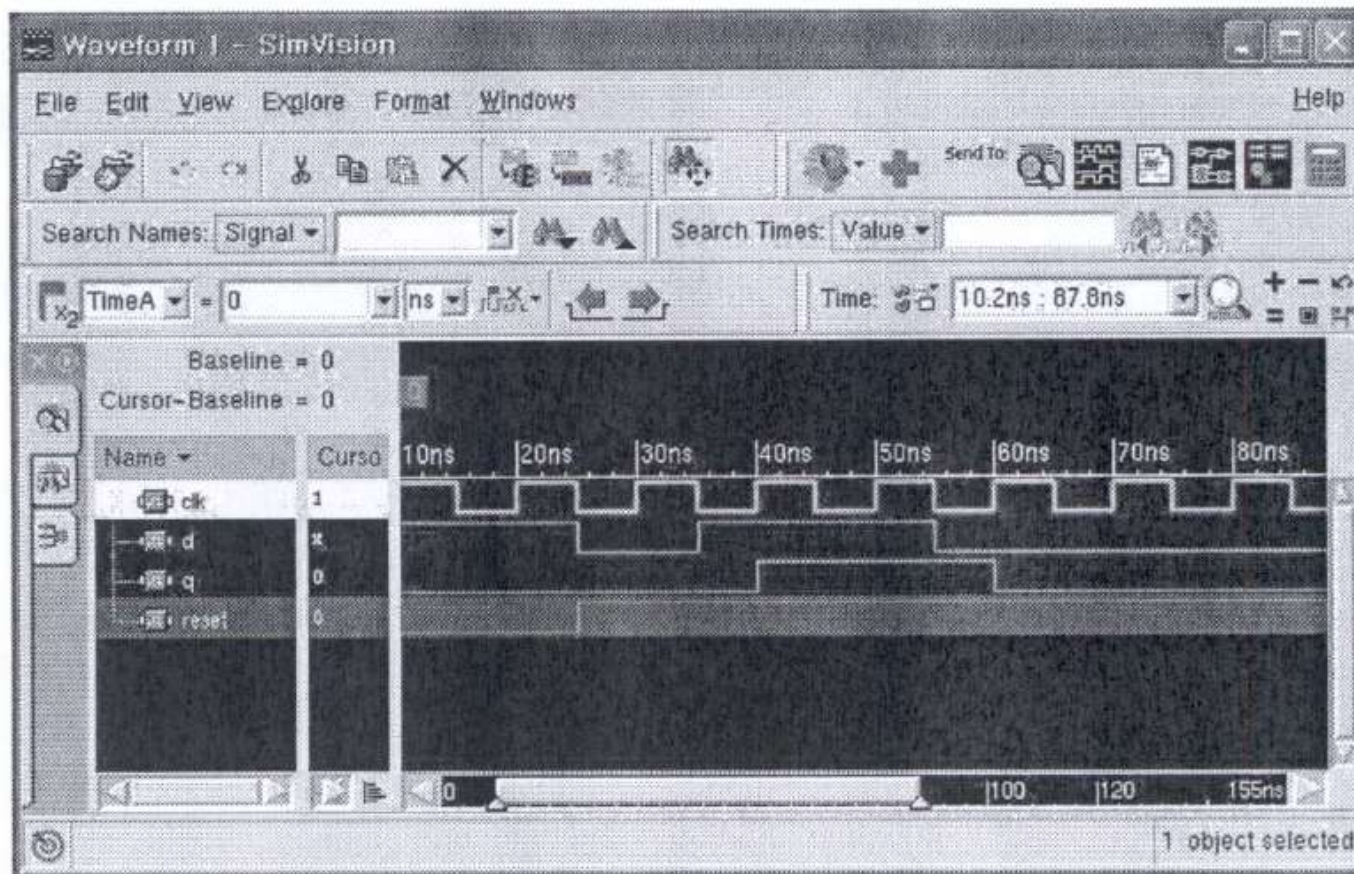
◆ 보기 원하는 신호들을 드래그하여 선택하고 **waveform** 아이콘을 선택



## Simvision을 이용한 파형 분석

### • Waveform Window













◆ 파형을 축소 확대 하며 출력 값을 확인한다.





## Simvision을 이용한 파형 분석

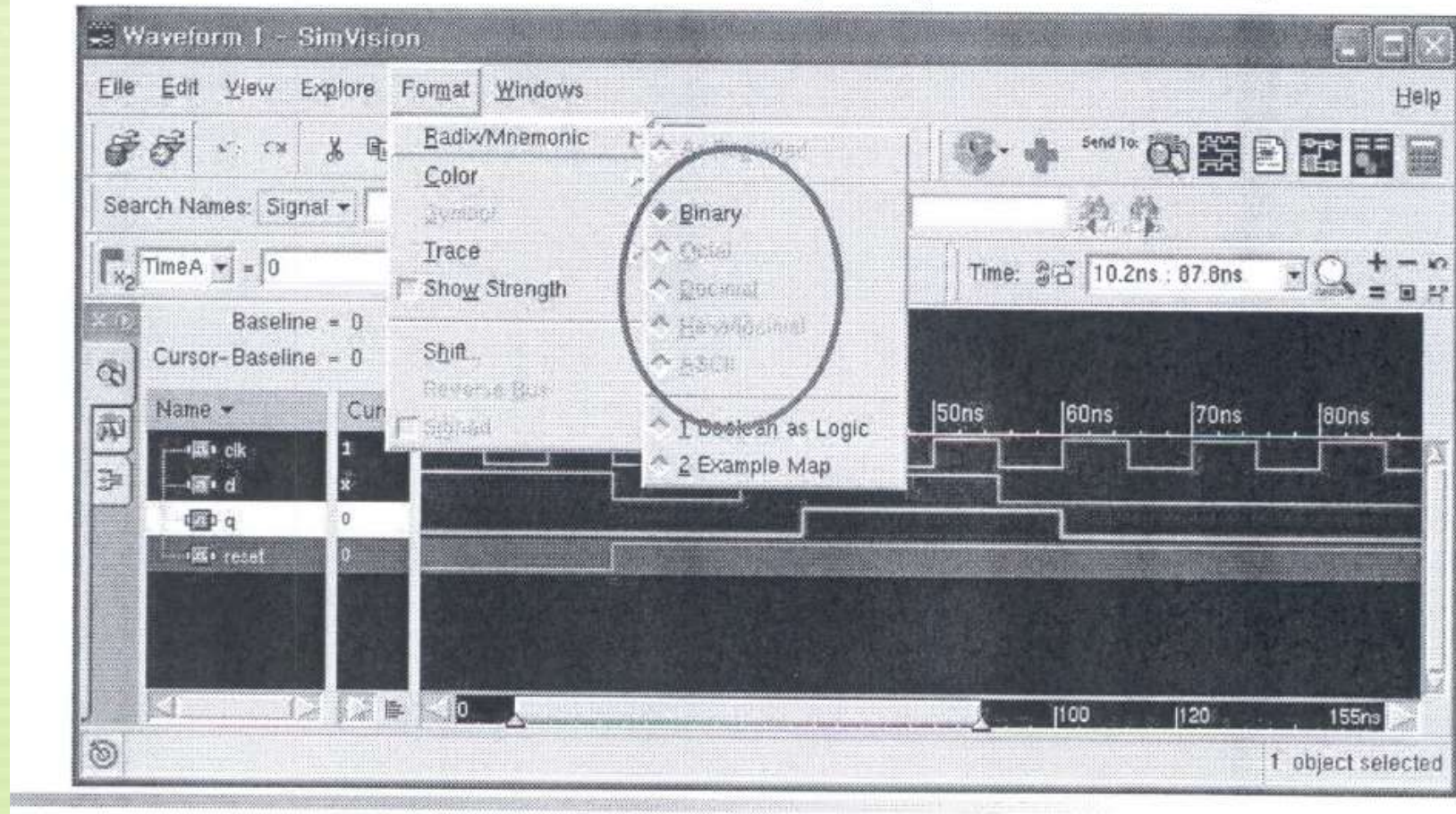
### • Waveform Window

-   Zoom in incrementally
-   Zoom out incrementally
-   Display all of the waveform data in the window
-   Display the waveform data between cursor
-   Move cursor to previous edge of selected signals
-   Move cursor to next edge of selected signals

## Simvision을 이용한 파형 분석

### • Waveform Window

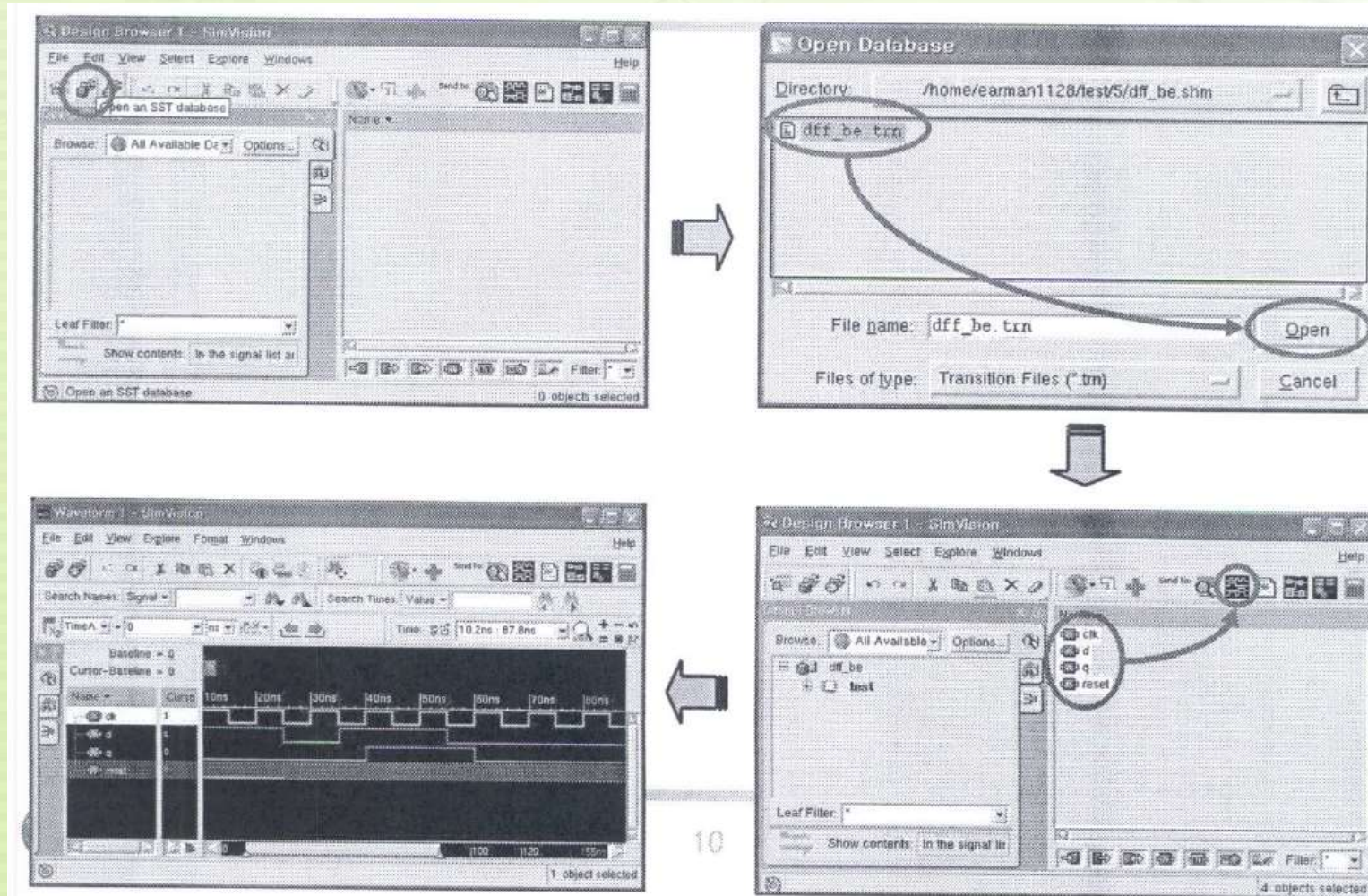
◆ 진수를 바꾸려면 **Format -> Radix -> (Binary or Decimal or ... )**





# Simvision을 이용한 파형 분석

## • 전체과정







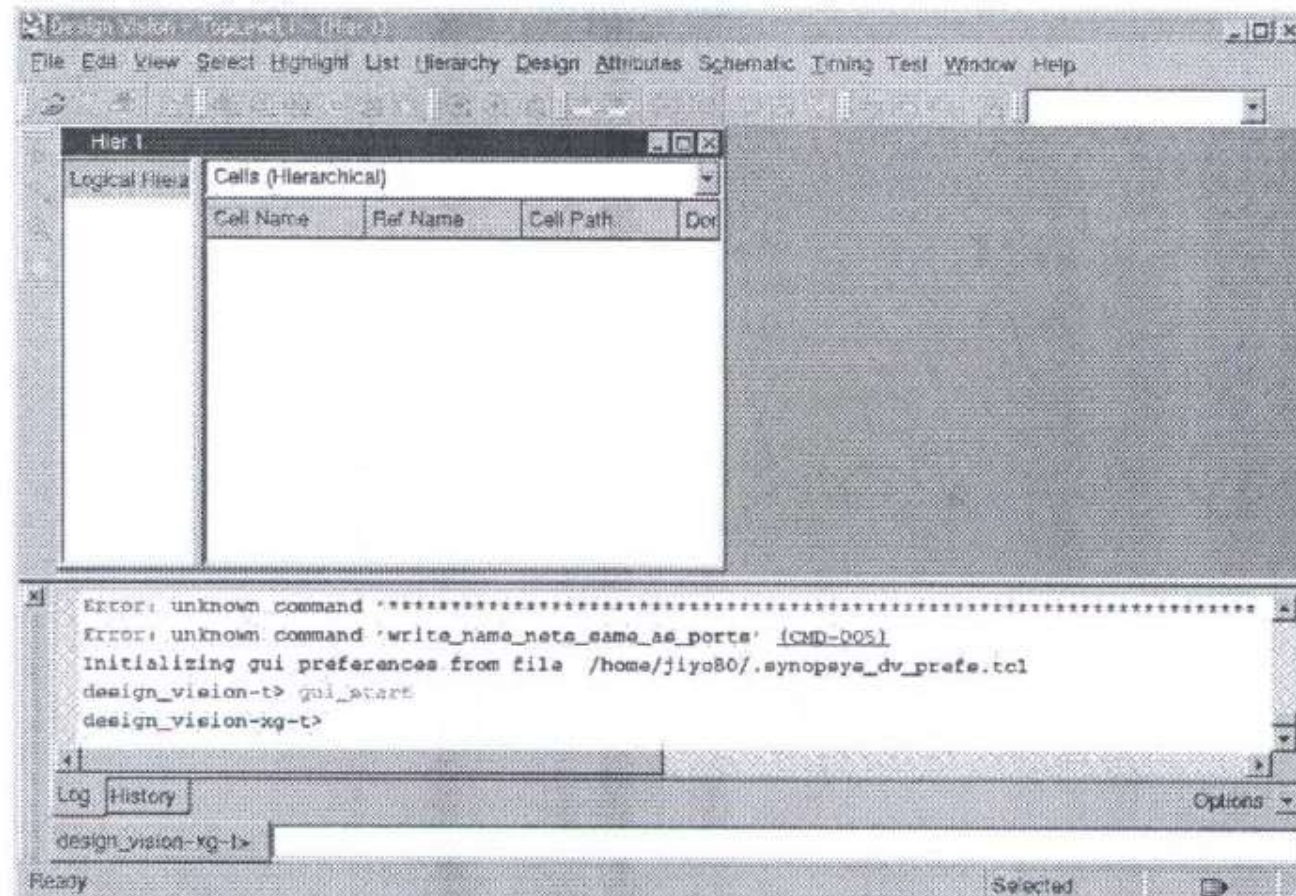
# Design Vision을 이용한 합성

## Design Vision을 이용한 합성

### • Graphical Interface

◆ 'design\_vision &'를 실행한다. 실행위치: **user의 lab# 디렉토리**

- Design-Vision 은 기본적으로 TCL 기반의 명령어를 사용한다.

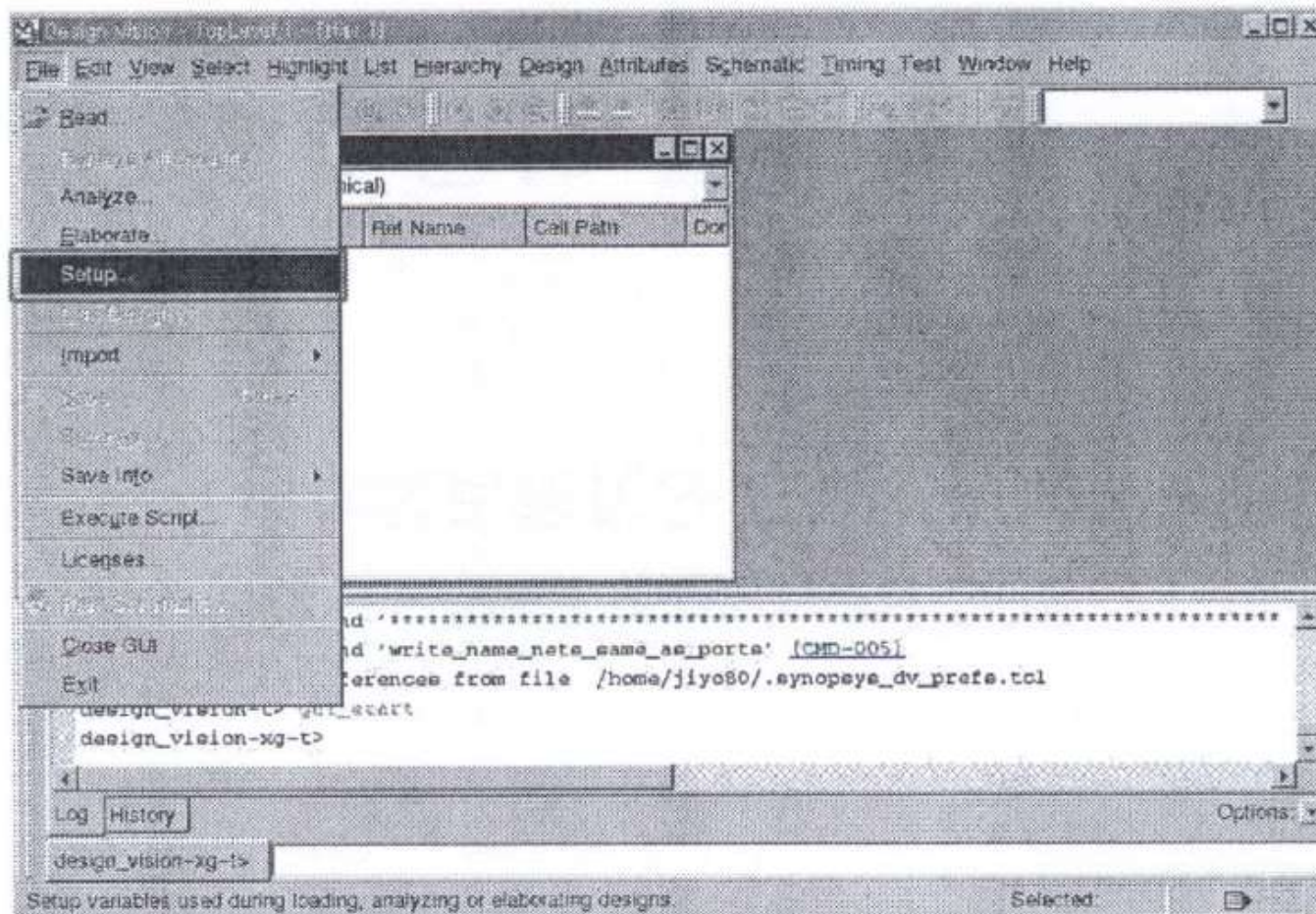




## Design Vision을 이용한 합성

### • Check Default Setup (1)

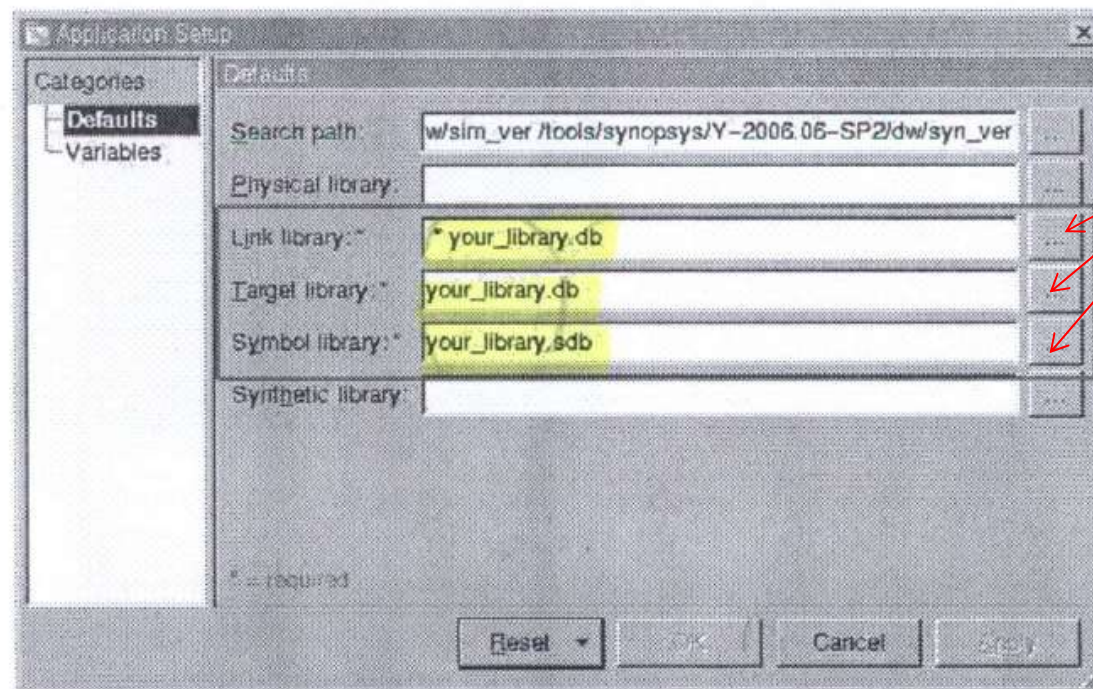
◆ Pop-up Menu 중 File → Setup... 을 선택한다.



## Design Vision을 이용한 합성

### • Check Default Setup (2)

- ◆ Window의 Defaults 항목에서 Link Library(\* \*.db)와 Target Library(\*.db), Symbol Library(\*.sdb)에 자신이 사용하고자 하는 library를 setting한다.



추가/삭제 할 수 있는  
browser button

## Design Vision을 이용한 합성

- Check Default Setup (3)

- Link library: \* **class.db** (your\_library.db 는 삭제)
- Target library: **class.db** (your\_library.db 는 삭제)
- Symbol library: **class.sdb** (your\_library.sdb 는 삭제)

- **class.db** 및 **class.sdb** 위치 :

- 현재 위치가 **lab#** 일 때: **../techlib/**

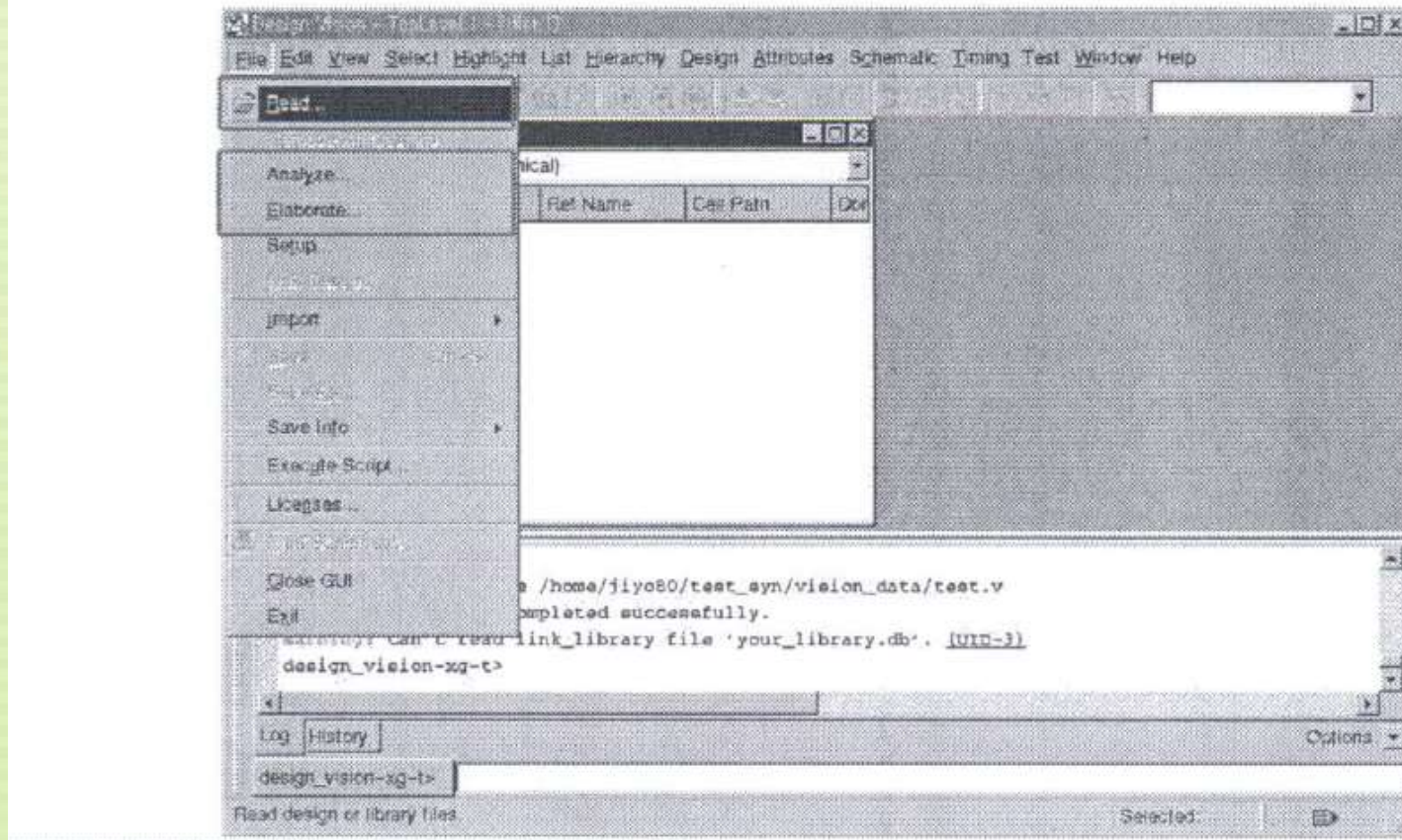


## Design Vision을 이용한 합성

### • Design Read

◆ 합성하고자 하는 **source code**를 읽는다.

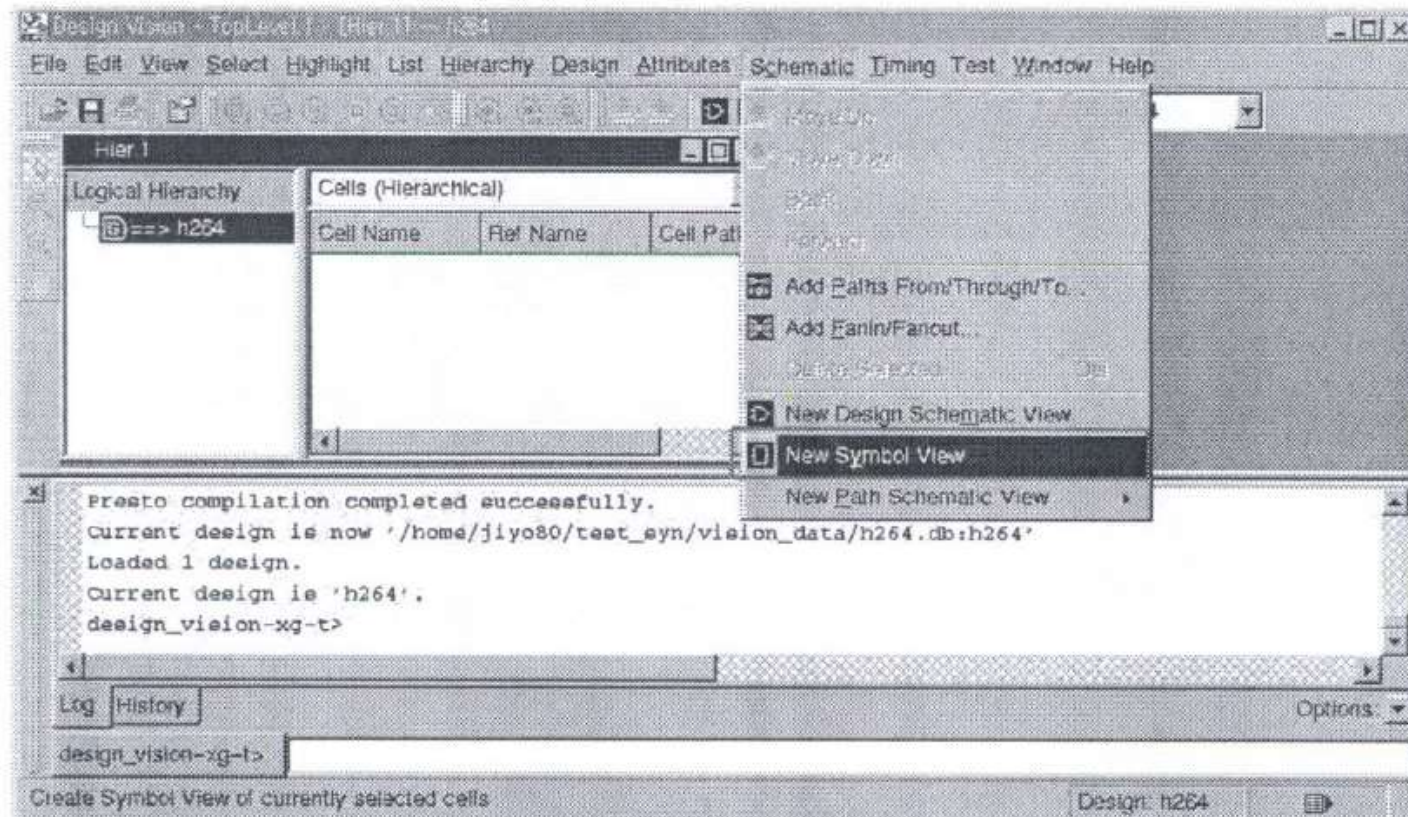
- File → Read...나 File → Analyze...,Elaborate...



## Design Vision을 이용한 합성

### • Symbol View (1)

- ◆ 'Hier.1' 창에서 Top module의 이름을 선택한 다음 Schematic → New Symbol View 를 선택한다.

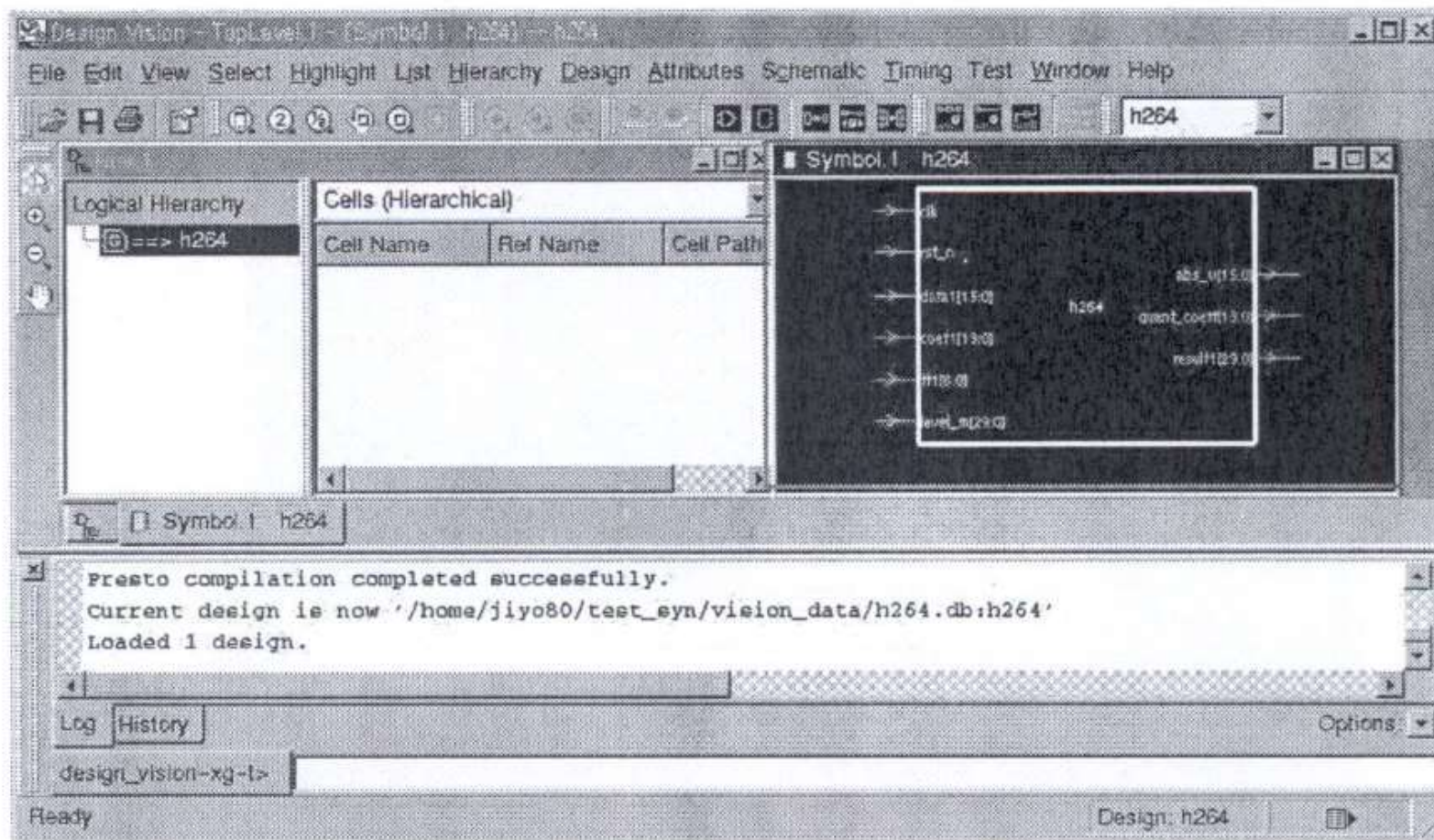




## Design Vision을 이용한 합성

### • Symbol View (2)

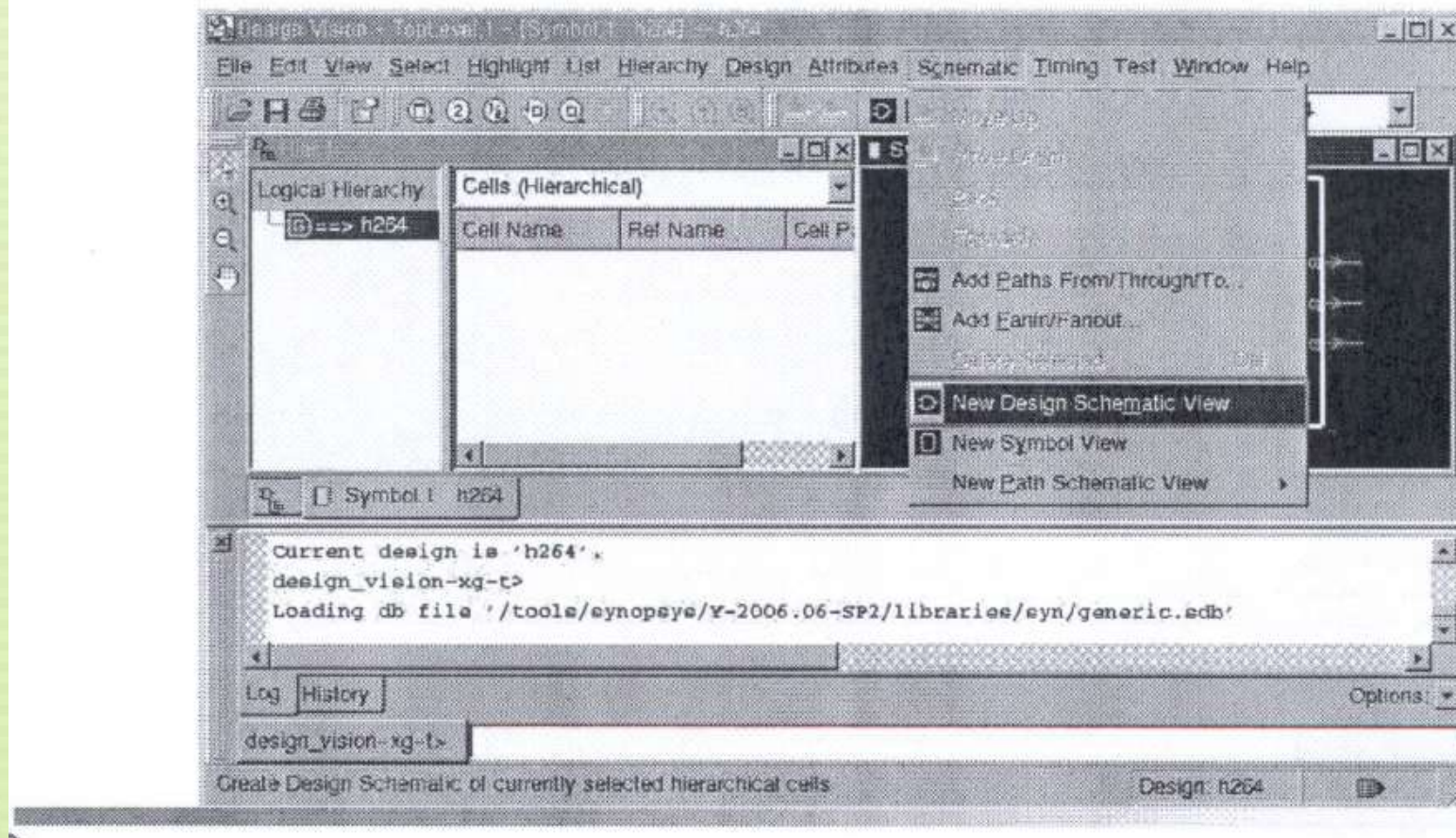
- 설계자가 만든 module의 입출력 포트를 확인할 수 있다.



## Design Vision을 이용한 합성

### • Schematic View (1)

◆ Schematic → New Design Schematic View 를 선택한다.

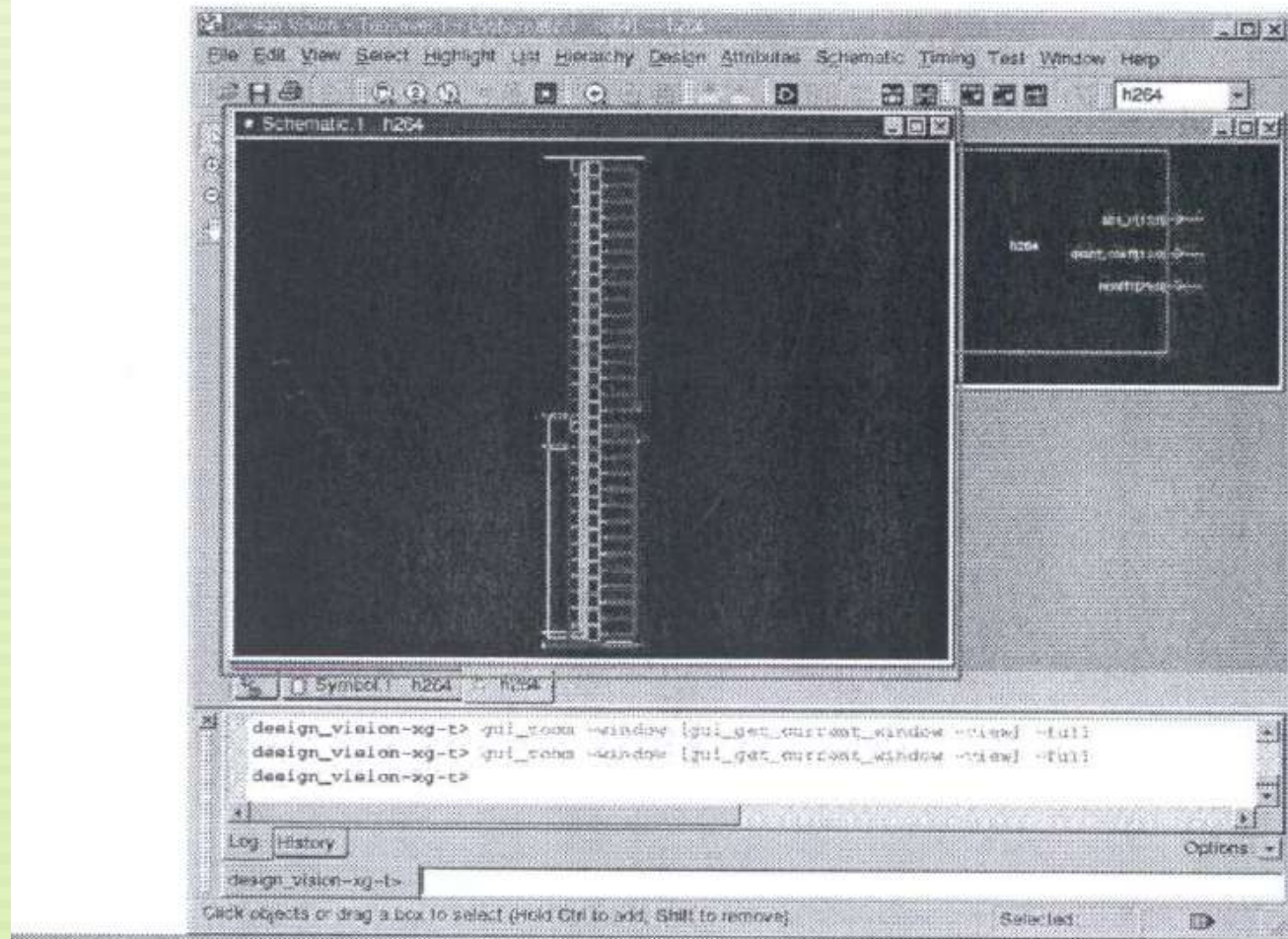




## Design Vision을 이용한 합성

### • Schematic View (2)

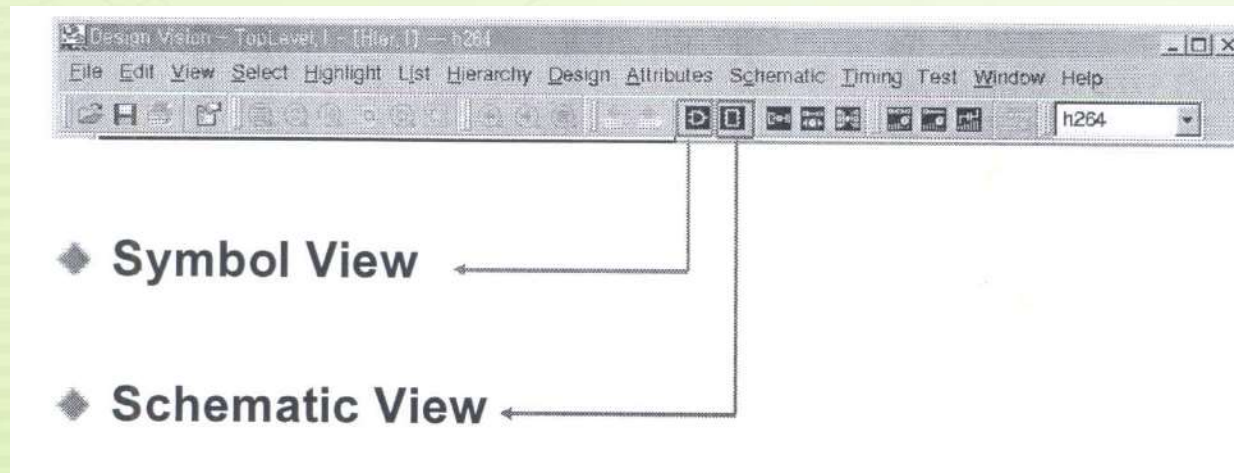
- 합성 전의 게이트 수준의 회로를 확인해 볼 수 있다.





## Design Vision을 이용한 합성

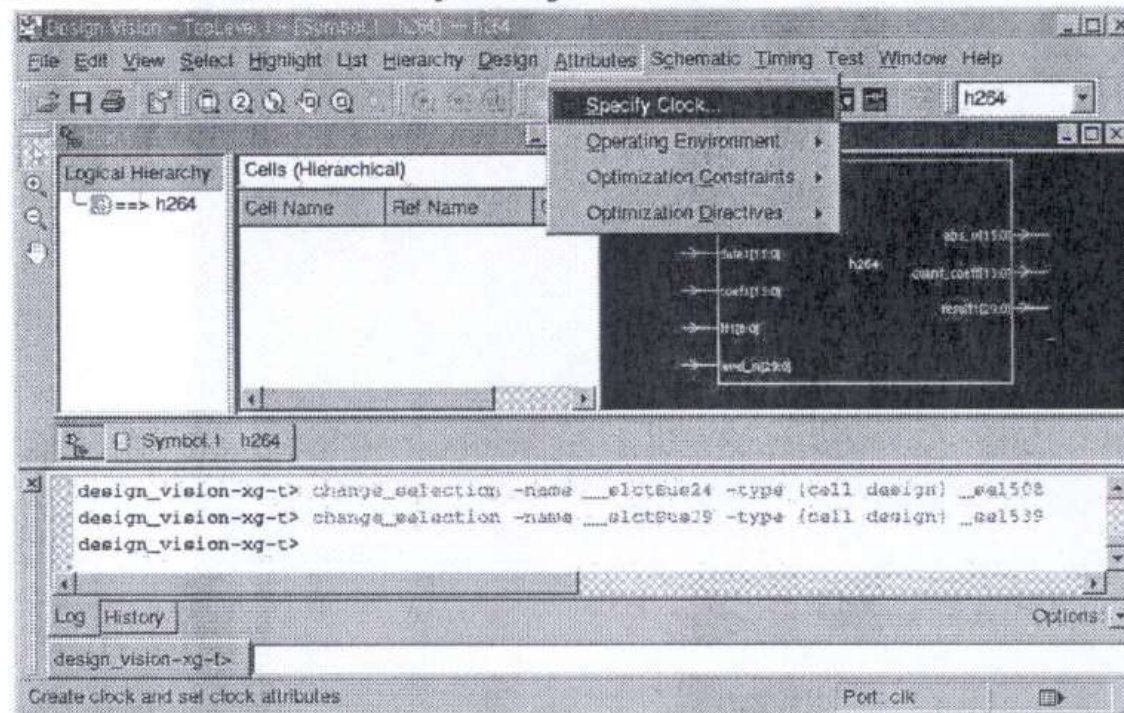
### • Views



## Design Vision을 이용한 합성

### • Clock Constraints (1)

- ◆ Symbol View 상태에서 Clock Port를 선택한다.
- ◆ 그 다음 Attributes → Specify Clock... 을 선택한다.



# Design Vision을 이용한 합성

## • Clock Constraints (2)

- ◆ 원하는 clock의 period와 edge를 설정한다.
- ◆ Don't Touch Network를 선택한다.

Specify Clock

Clock name:

Port name:

☐ Remove clock

Clock creation

Period:

Edge	Value
Rising	0
Falling	10

☒ Don't touch network

☐ Fix hold

0.00 10.00 20.00

Clock 이름

Clock의 주기

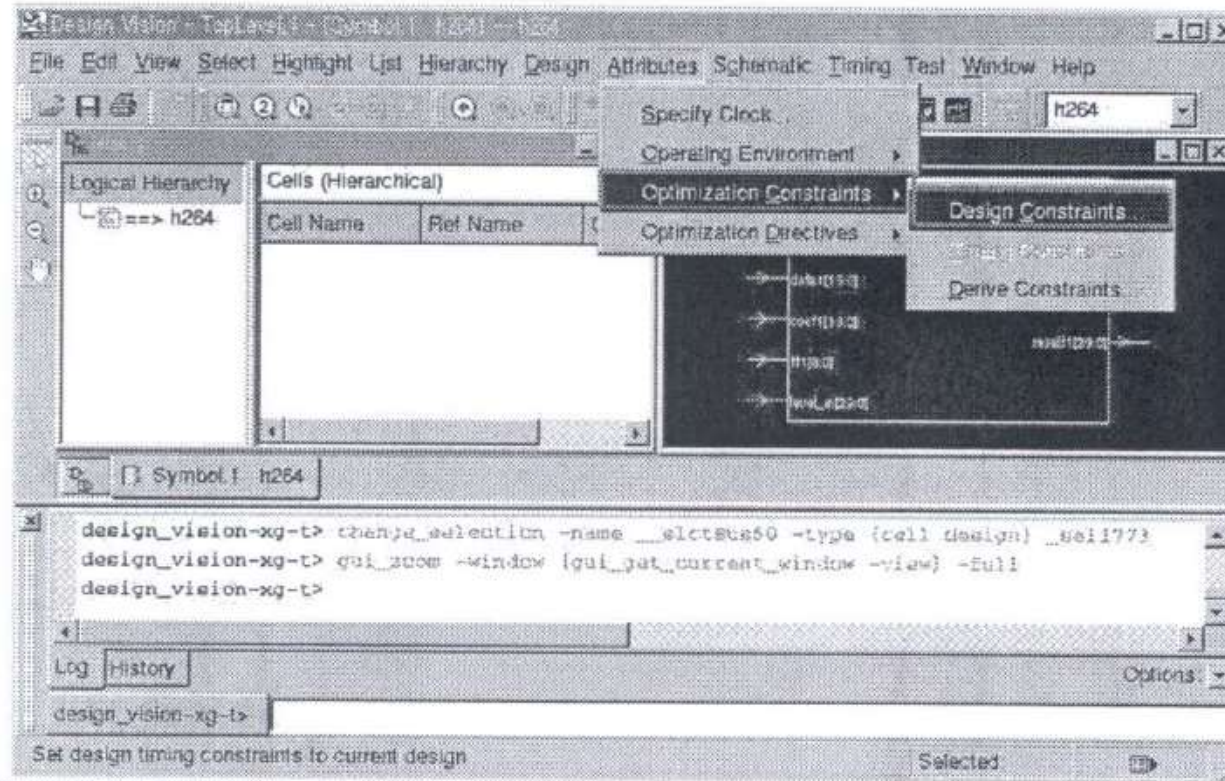
Clock의 rising edge의 시간과 falling edge의 시간



## Design Vision을 이용한 합성

### • Design Constraints (1)

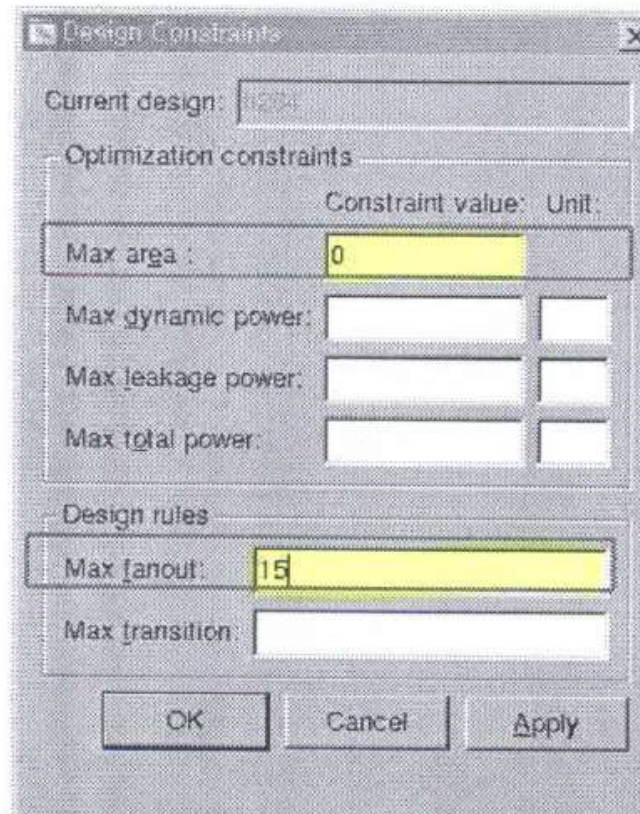
- ◆ Optimization 과정을 위한 Constraints를 설정한다.
  - Attributes → Optimization Constraints → Design Constraints...를 선택한다.



## Design Vision을 이용한 합성

### • Design Constraints (2)

- ◆ Optimization 과정을 위한 Constraints를 설정한다.
  - Popup된 창에서 Max Area, Max fanout 등을 설정한다.



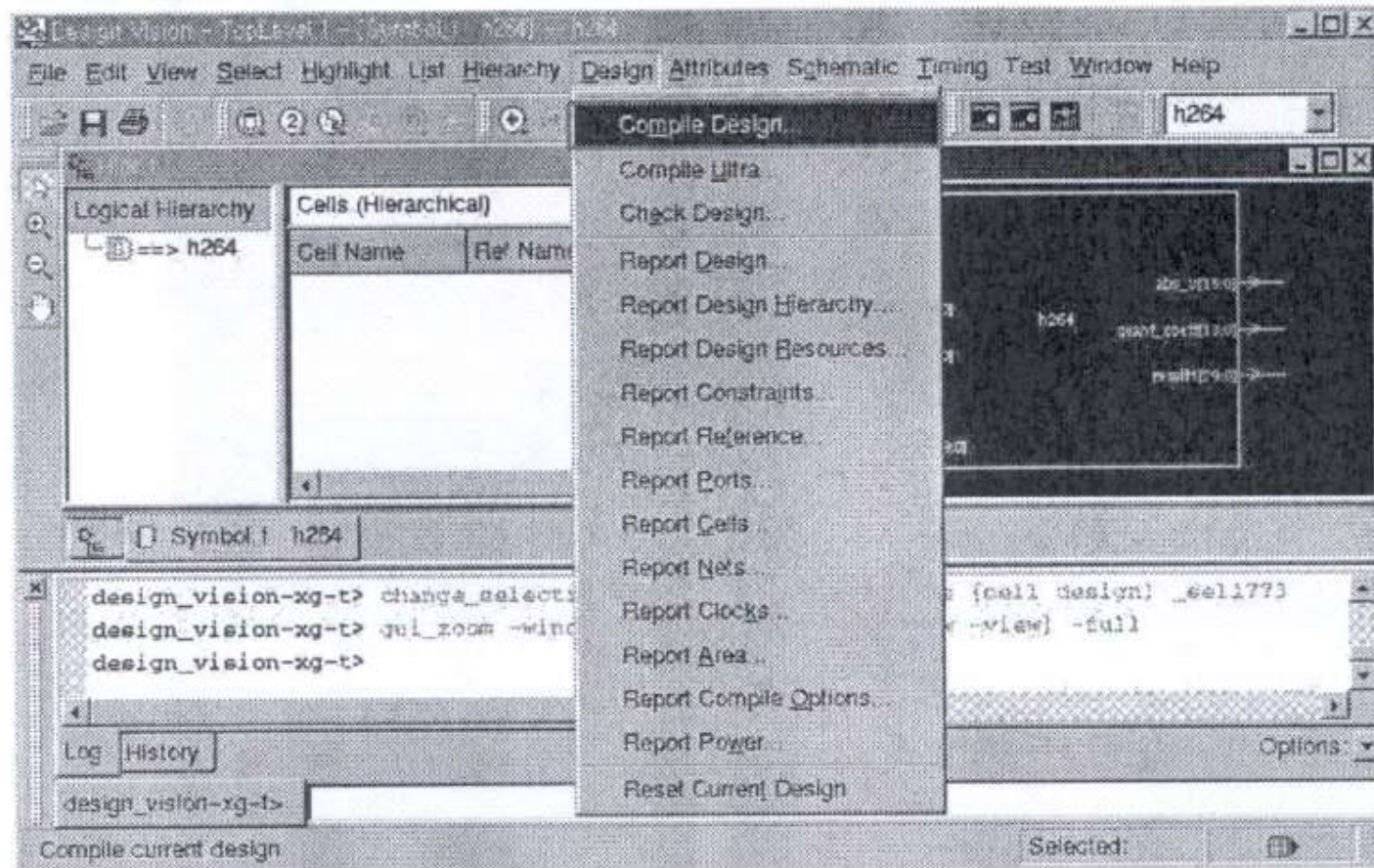


## Design Vision을 이용한 합성

### • Design Optimization (1)

#### ◆ Optimization 과정을 수행한다.

- Design → Compile Design...을 수행한다.

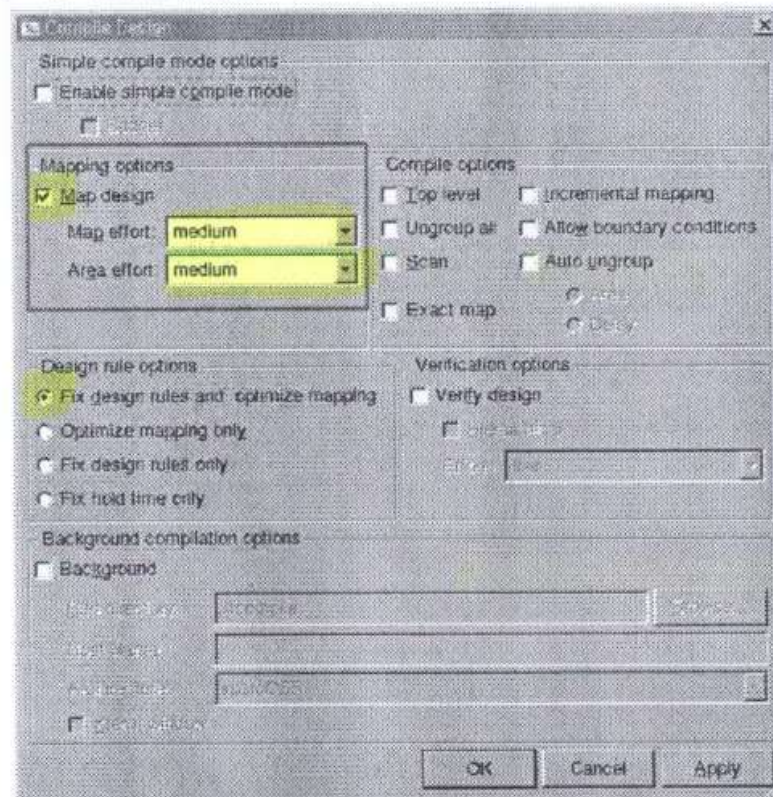




## Design Vision을 이용한 합성

### • Design Optimization (2)

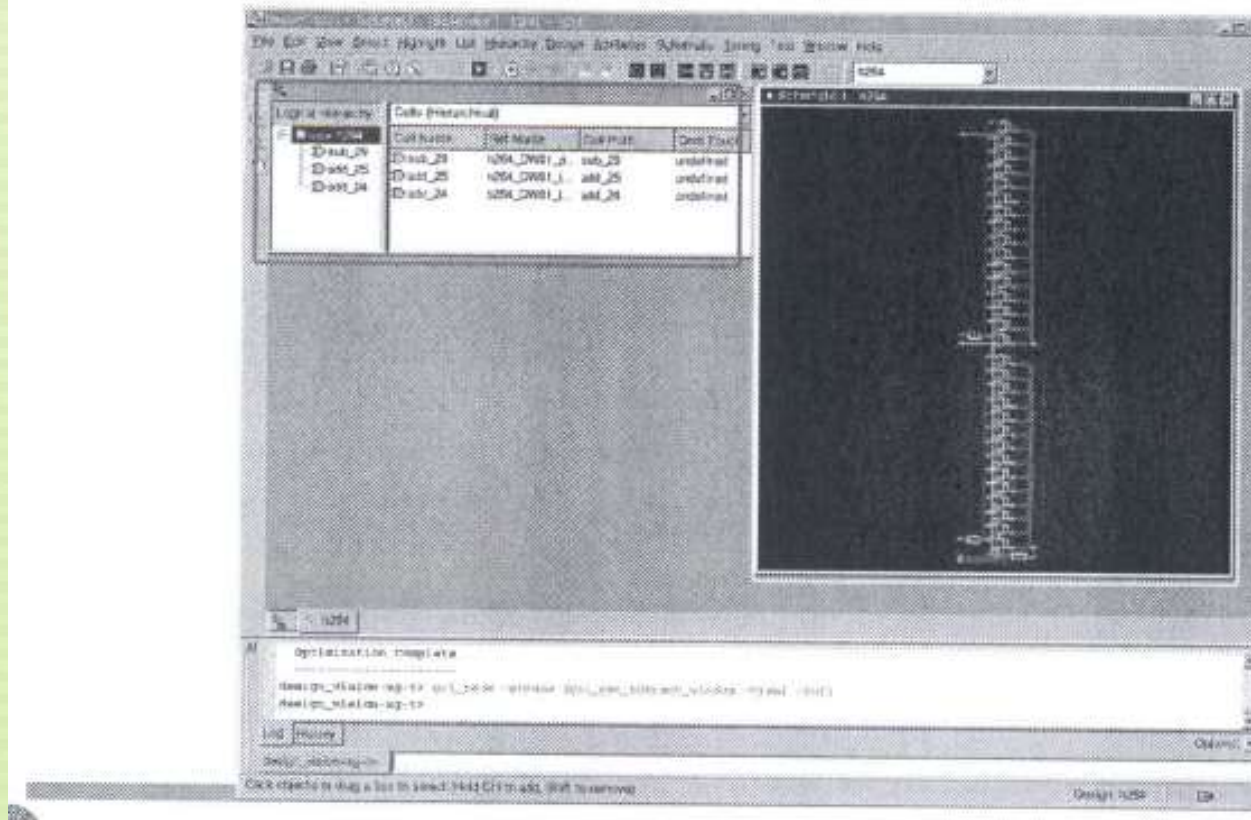
- ◆ **Optimization** 의 단계를 결정하는 부분으로서 보통 **Medium**을 선택하여 수행한다.



## Design Vision을 이용한 합성

### • Design Optimization (3)

- ◆ Optimization 수행한 후 다음과 같은 계층 구조를 볼 수 있으며, Schematic View를 통하여 게이트 수준의 회로 구성을 볼 수 있다.

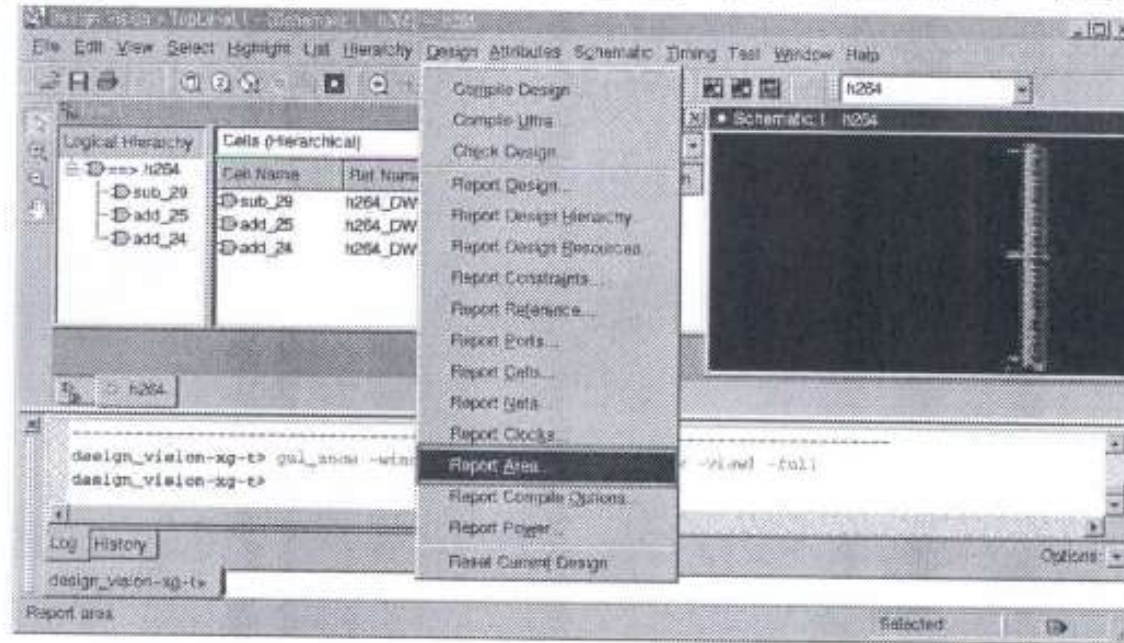


## Design Vision을 이용한 합성

### • Report Area (1)

◆ 합성된 회로의 area, timing, power 등의 정보를 얻는다.

- Design → Report Area... 를 선택하여 크기를 확인한다.

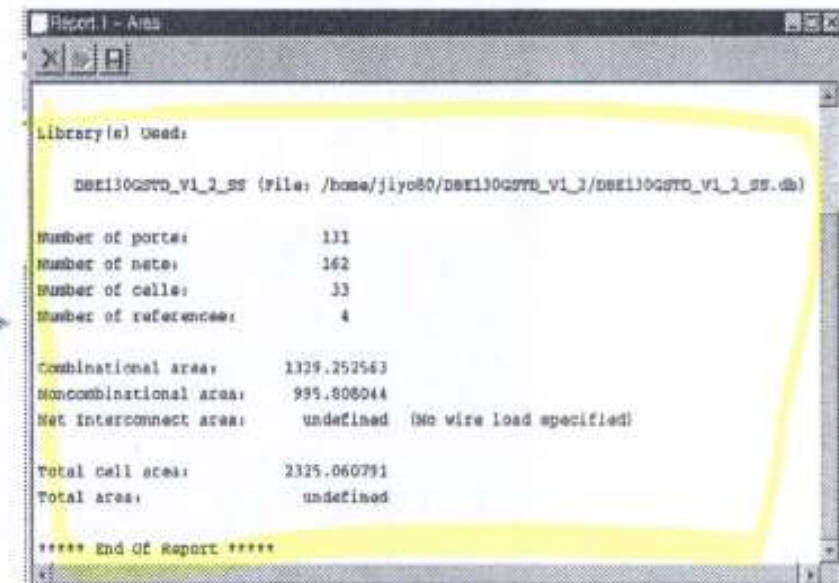
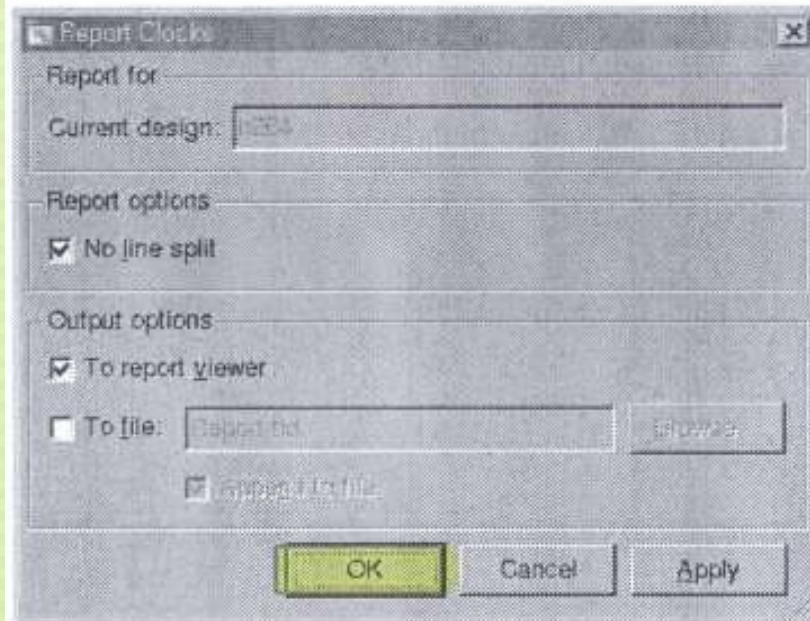




## Design Vision을 이용한 합성

### • Report Area (2)

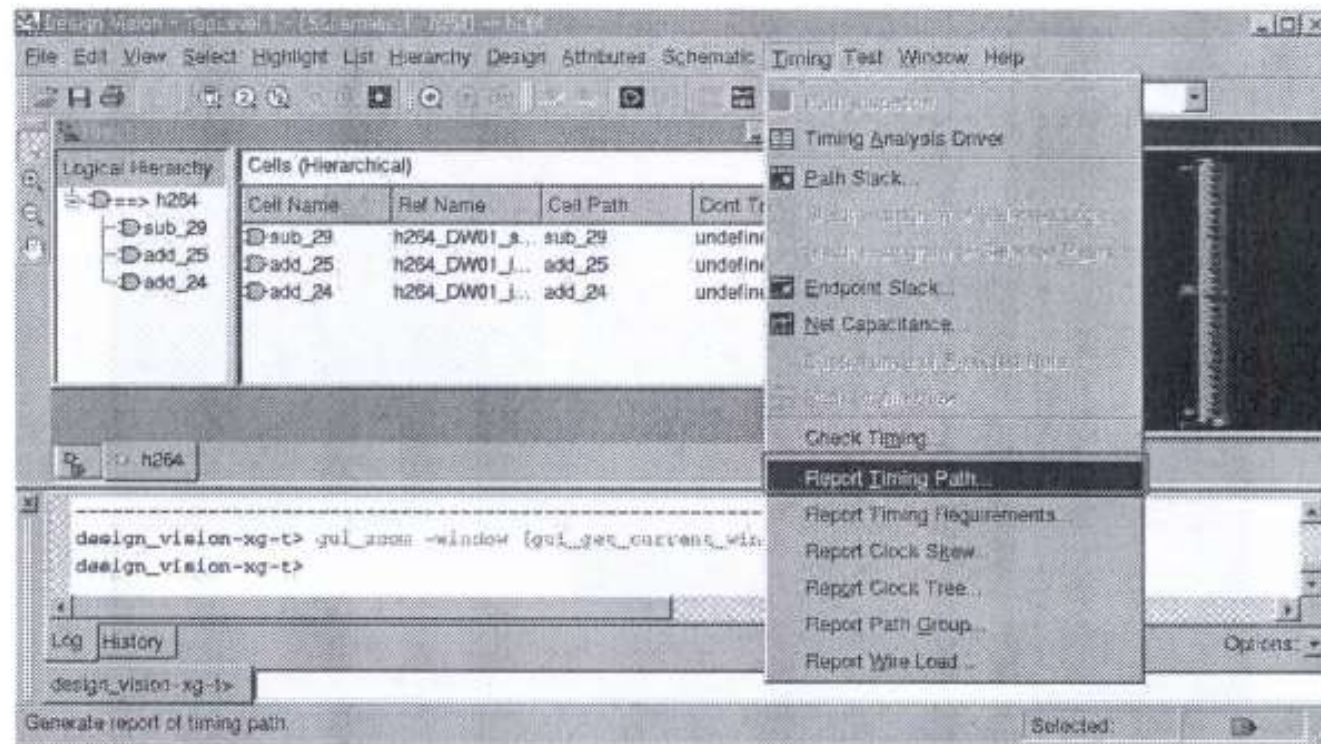
- 다음과 같은 Popup 창이 나오면 OK 버튼을 누른다.
- Report 창이 뜨면 정보를 확인한다.



## Design Vision을 이용한 합성

### • Report Timing (1)

- **Timing → Report Timing Path...** 를 선택하여 지연시간을 확인한다.



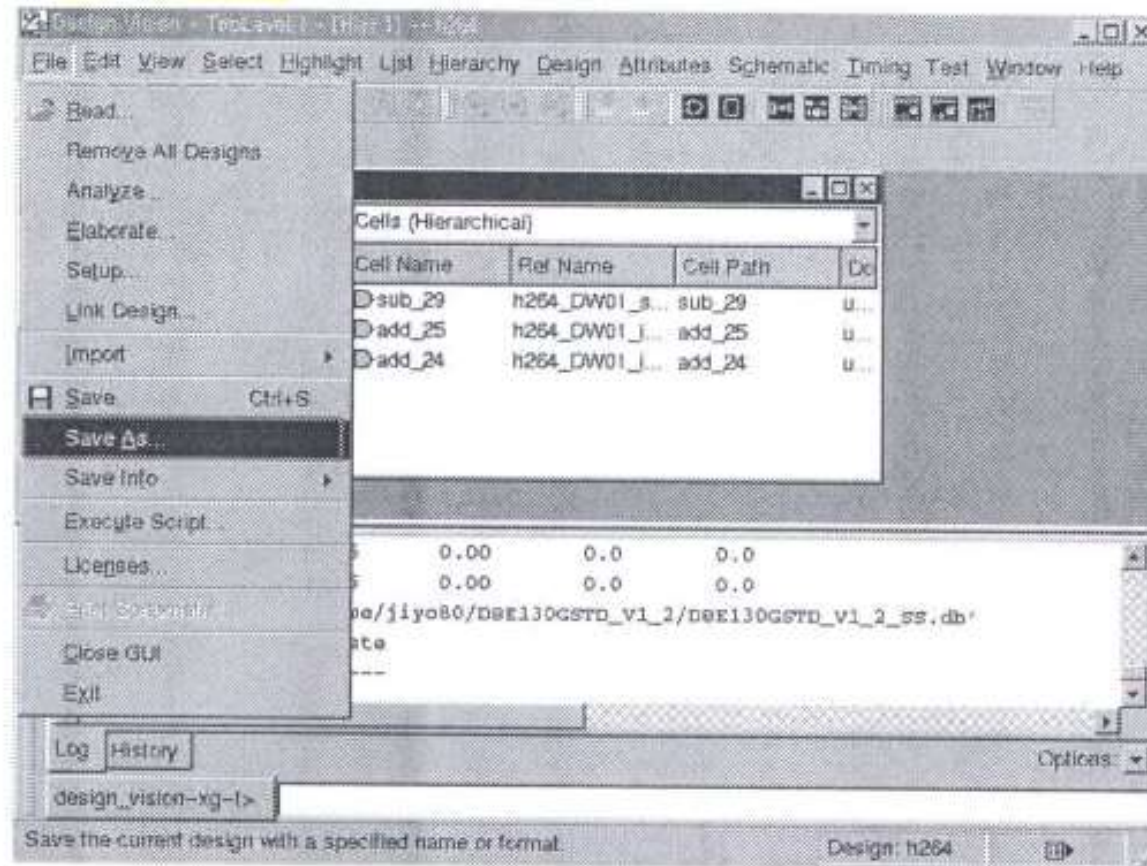




## Design Vision을 이용한 합성

- Netlist Save (1)

◆ File → **Save As...** 를 선택한다.

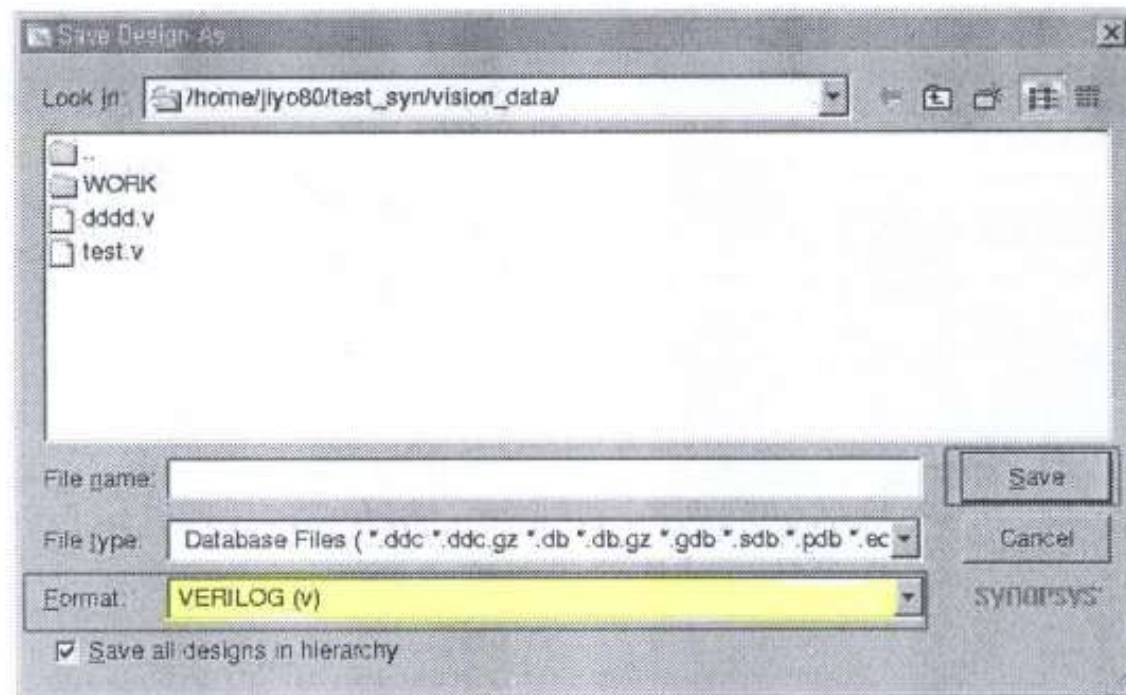




## Design Vision을 이용한 합성

- Netlist Save (2)

◆ Format은 VERILOG (v)를 선택하고 원하는 이름으로 저장한다.



## Tool 사용 실습 - (1/2)

### [Tool 사용 예제 위치]

- directoy: `./example/`
- design: `./example/spram_g.v` ← RTL 디자인
- testbench: `./example/spram_g_tb.v` ← 검증을 위해 사용되는 테스트 벤치 파일

### [RTL level 실습]

#### 1) RTL simulation

- `./example/nc spram_g_tb.v` ← NC-verilog를 사용한 simulation

#### 2) 상기 '1)'의 RTL simulation 결과를 Simvision을 통해 signal wave form 확인

- `./example/spram_g_64x16_shm/spram_g_64x16_shm.trn`

파일에 simulation 결과가 저장되며 ,

Simvision 을 사용하여 simulation 결과 확인



## Tool 사용 실습 - (2/2)

### [Gate level 실습]

#### 1) Gate-level logic synthesis

- DesignVison (DC) 으로 **spram\_g.v** 디자인 파일을 logic synthesis 후 **spram\_g\_syn.v** 이름으로 저장

“주의: **Synthesis** 후 그 결과를 파일로 저장할 경우

파일이름이 **RTL** 원본 디자인 파일 이름과 다르게 할 것

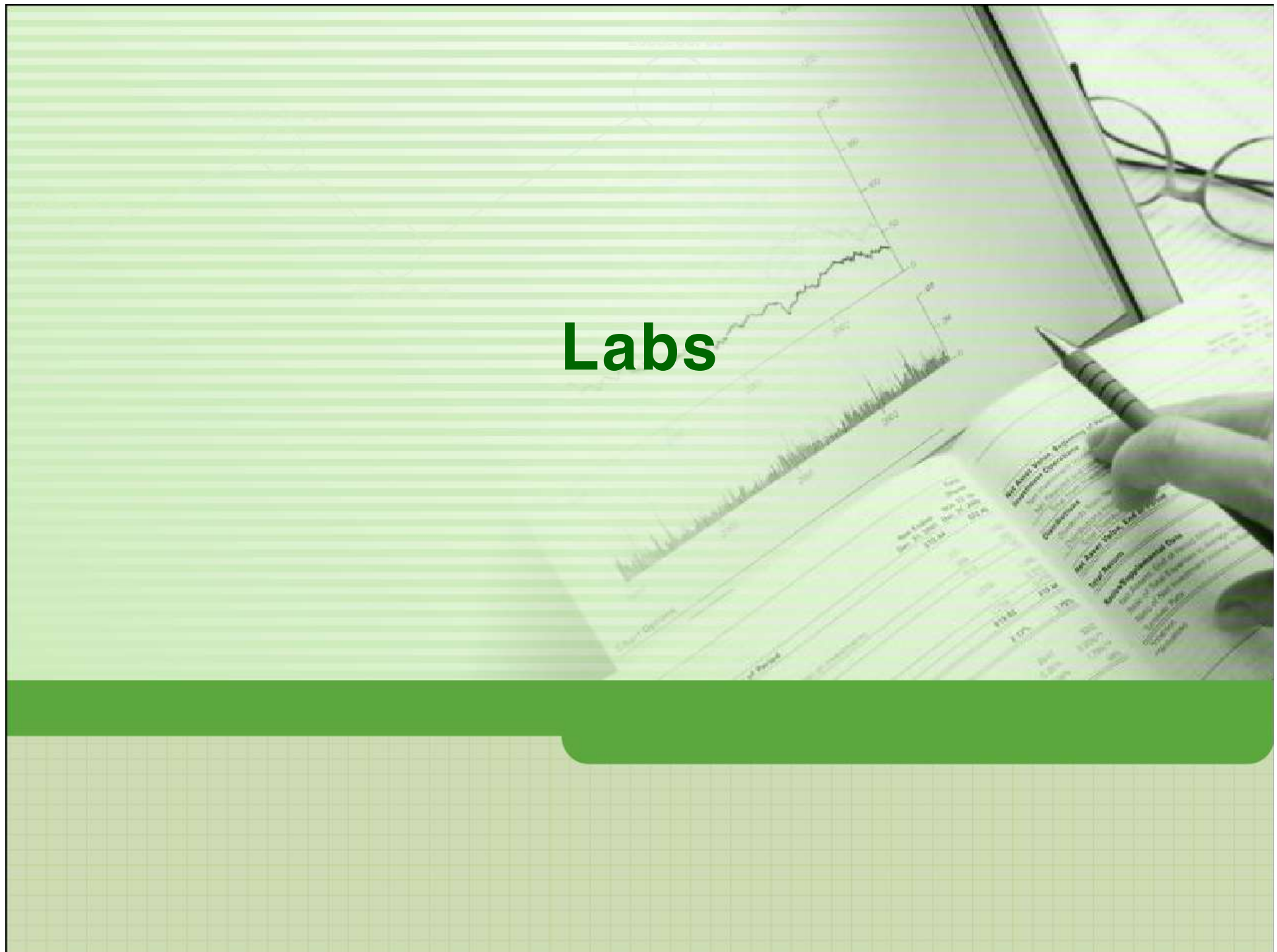
(같은게 되면 원본 디자인 파일이 **override** 됨)”

#### 2) Gate-level simulation

- **spram\_g\_tb.v** 파일 내용을 수정
  - ➔ **`define GATE\_SIM** 활성화(uncomment 처리) 시킨 후 저장
- NC-verilog를 사용한 simulation
  - ➔ **./example/nc spram\_g\_tb.v**
- simulation 결과 확인
  - ➔ **./example/spram\_g\_64x16\_syn\_shm/spram\_g\_64x16\_syn\_shm.trn**

에 simulation 결과가 저장되며 **Simvision** 을 사용하여 결과 확인

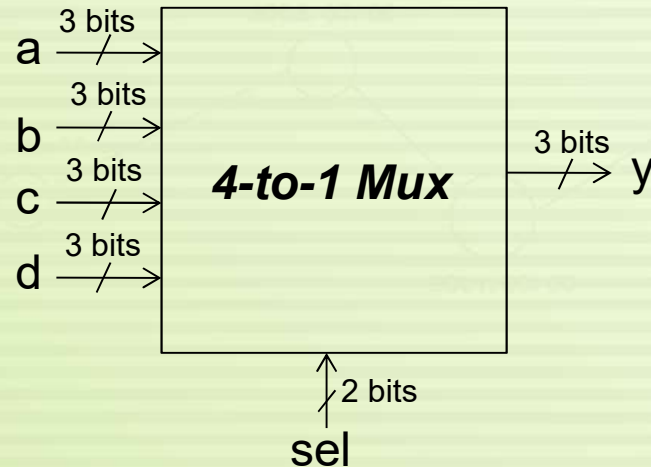
# Labs





## Lab. 1: 4-to-1 Mux

### • 4-to-1 Mux block



### • 4-to-1 Mux block spec.

- 3-bit 4 data inputs (a, b, c, d)
- 3-bit data output (y)
- 2-bit select input (sel)
- Use “case” statement
- Default output value is ‘0’ for ‘x’ or ‘z’ on the “sel” input
- 모듈이름: mux4\_1
- 모듈 설계 파일이름: mux4\_1.v,
- 모듈의 게이트레벨 합성파일이름: mux4\_1\_syn.v (netlist file)
- Test bench 파일이름: mux4\_1\_tb.v (주어짐)
- RTL sim. 과 Gate sim. 결과 비교하기

### • 4-to-1 Mux block waveform (RTL Sim.)



## Lab. 1: 4-to-1 Mux

### • 4-to-1 Mux block test bench

```

`timescale 1ns / 1ps
//`define GATE_SIM

`ifndef GATE_SIM
    `include "mux4_1.v"          // design for RTL sim.
`else
    `include "mux4_1_syn.v"      // design for gate sim.
    `include "../techlib/class.v" // tech. lib. for gate sim.
`endif

module tb_mux4_1;

wire[1:0] out;
reg[2:0] a, b, c, d;
reg[1:0] sel;

// Connect DUT to test bench
mux4_1 u_mux(out, a, b, c, d, sel);
    
```

```

initial
begin
    `ifndef GATE_SIM
        $shm_open("mux4_1_shm"); // for RTL design
    `else
        $shm_open("mux4_1_syn_shm"); // for netlist design
    `endif

    $shm_probe("AMCTF");

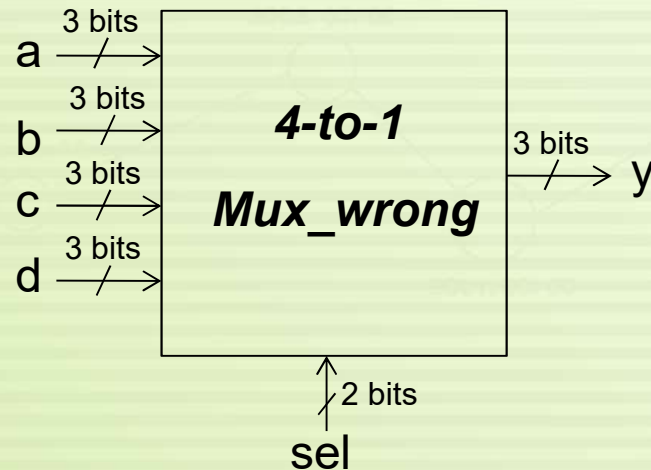
    a = 3'b111;
    b = 3'b100;
    c = 3'b010;
    d = 3'b110;
    sel = 2'b00;
    #10 sel = 2'b01;
    #10 sel = 2'b10;
    #10 sel = 2'b11;
    .....
    #40 $finish;

end
endmodule
    
```



## Lab. 2: 4-to-1 Mux\_wrong

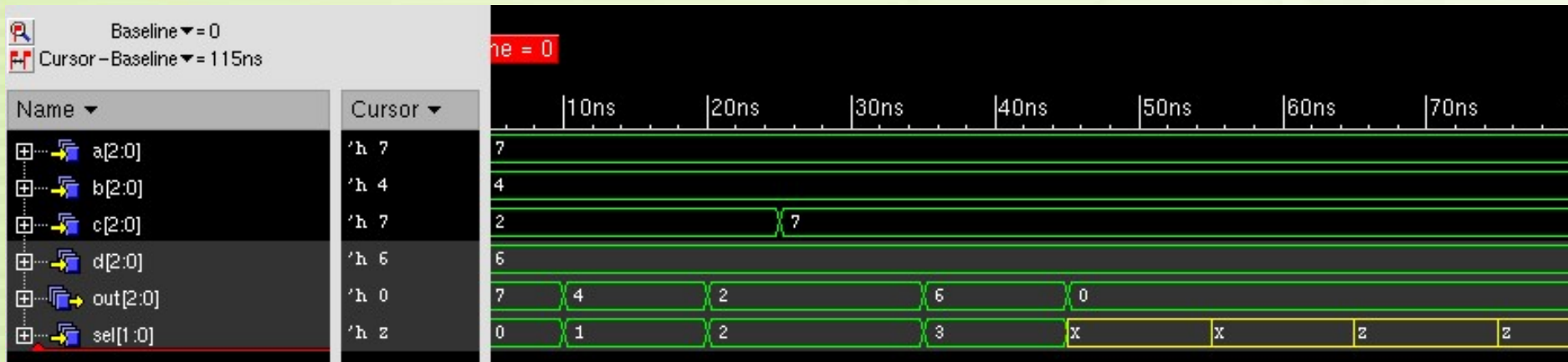
### • 4-to-1 Mux\_wrong block



### • 4-to-1 Mux\_wrong block spec.

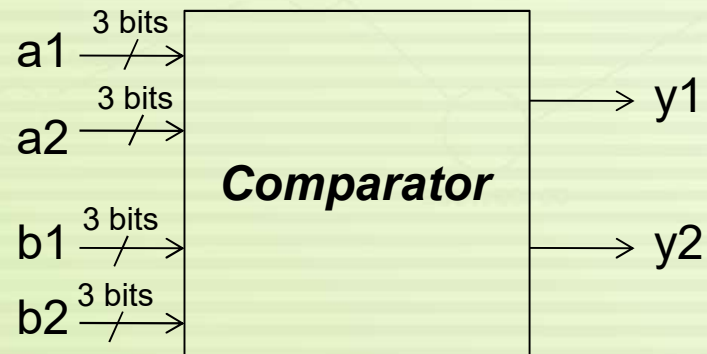
- Lab.1 4-to-1 Mux 의 verilog code 에서 sensitivity list 상의 입력포트 'c' 를 삭제 한 후 RTL Sim. 및 Gate Sim. 결과를 비교하기.
- 모듈이름: mux4\_1\_wrong
- 모듈 설계 파일이름: mux4\_1\_wrong.v,
- 모듈의 게이트레벨 합성파일이름:  
mux4\_1\_wrong\_syn.v (netlist file)
- Test bench 파일이름: mux4\_1\_wrong\_tb.v (주어짐)

### • 4-to-1 Mux\_wrong block waveform (RTL Sim.)



## Lab. 3: Comparator

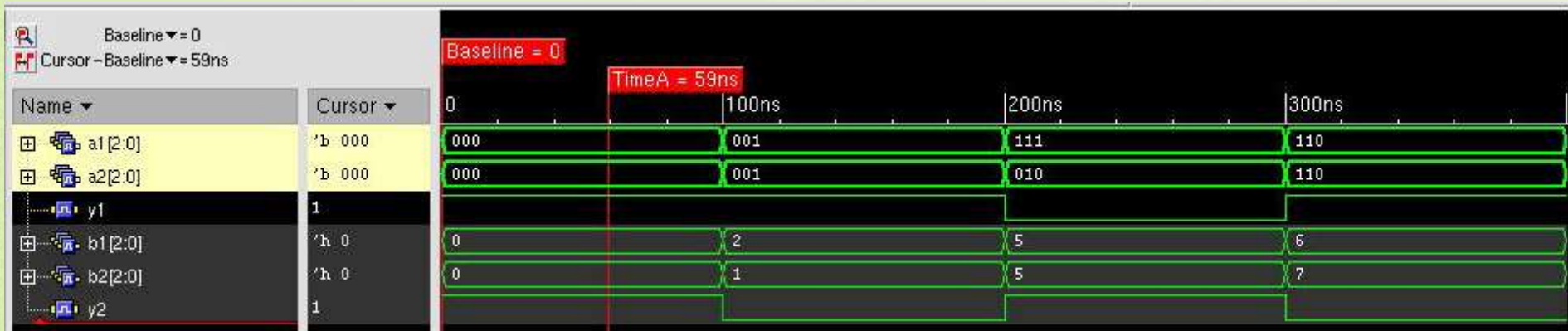
### • Comparator block



### • Comparator block spec.

- 두 개의 입력을 비교하여 등가 여부를 판별
- y1은 a1, a2의 비교 결과 출력
- y2은 b1, b2의 비교 결과 출력
- 두 수가 같을 때: logic 값 '1' 출력
- 두 수가 다를 때: logic 값 '0' 출력
- 모듈 설계이름: comparator
- Input ports: a1, a2, b1, b2 [각 3-bit]
- output ports: y1, y2 [각 1-bit]
- 모듈 설계 파일이름: comparator.v,
- 모듈의 게이트레벨 합성파일이름: comparator\_syn.v (netlist file)
- Test bench 파일이름: comparator\_tb.v (만들기)

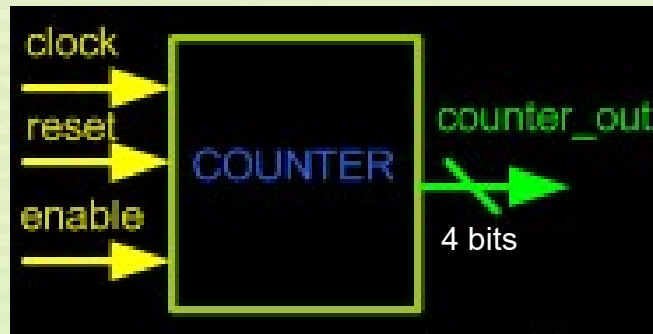
### • Comparator block waveform



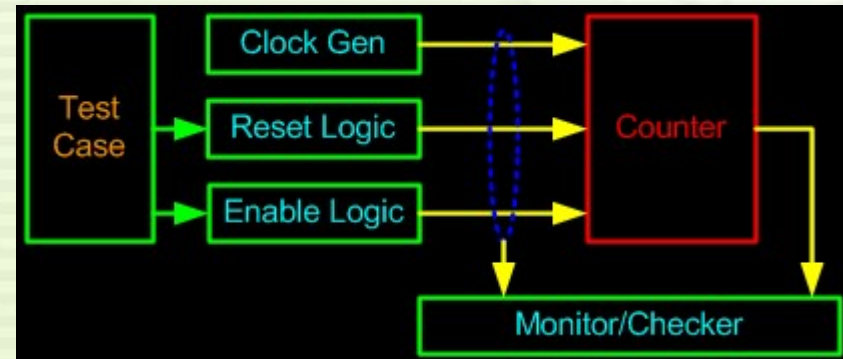


## Lab. 4: 4-bit up counter

### • 4-bit up counter block



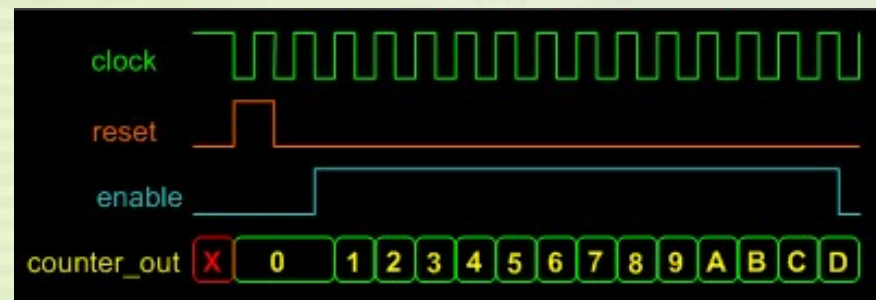
### • 4-bit up counter test bench diagram



### • 4-bit up counter spec.

- 4-bit synchronous up counter.
- Active high, synchronous reset.
- Active high enable
- 모듈이름: up\_counter
- 모듈 설계 파일이름: up\_counter.v,
- 모듈의 게이트레벨 합성파일이름: up\_counter\_syn.v (netlist file)
- Test bench 파일이름: up\_counter\_tb.v (제공됨)

### • 4-bit up counter waveform



## Lab. 4: 4-bit up counter

```

`timescale 1ns / 1ps
//`define GATE_SIM
`ifndef GATE_SIM
    `include "up_counter.v"          // design for RTL sim.
`else
    `include "up_counter_syn.v"      // design for gate sim.
    `include "../techlib/class.v"    // tech. lib. for gate sim.
`endif

module up_counter_tb();
// Declare inputs as regs and outputs as wires
reg clock, reset, enable;
wire [3:0] counter_out;

// Connect DUT to test bench
up_counter U_counter (clock, reset, enable, counter_out);

// Clock generator
always begin
    #5 clock = ~clock; // Toggle clock every 5 ticks
end

```

```

// Initialize all variables
initial begin
`ifndef GATE_SIM
    $shm_open("up_counter_shm"); // for RTL design
`else
    $shm_open("up_counter_syn_shm"); // for netlist design
`endif

    $shm_probe("AMCTF");
    $display ("time\t clk reset enable counter");
    $monitor ("%g\t %b  %b  %b  %b",
               $time, clock, reset, enable, counter_out);

    clock = 1;    // initial value of clock
    reset = 0;    // initial value of reset
    enable = 0;   // initial value of enable
    #5 reset = 1; // Assert the reset
    #10 reset = 0; // De-assert the reset
    #10 enable = 1; // Assert enable
    #200 enable = 0; // De-assert enable
    #5 $finish;    // Terminate simulation
end
endmodule

```



Thank you!