

**03**

**툴 설치하기 & 시뮬레이션 방법 익히기**

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## 무료 설계 툴 3가지 소개



# Icarus Verilog

## Icarus Verilog

- ✓ 장점 : 매우 가볍다
- ✓ 단점 : 기능이 적음 (연습용)

## Quartus

- ✓ 장점 : 다양한 기능 / 컴퓨터 사양 적당
- ✓ 단점 : 현업 활용 ↓ (but 실습 적합)



# VIVADO<sup>®</sup>

HLx Editions

## VIVADO

- ✓ 장점 : 가장 많은 기능 / 현업 활용 ↑
- ✓ 단점 : 무겁다 (컴퓨터 사양 ↑)

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Quartus 설치하기

quartus lite download

전체

뉴스

동영상

이미지

지도

더보기

도구

검색결과 약 95,300개 (0.32초)

도움말: 한국어 검색결과만 검색합니다. 환경설정에서 검색 언어를 지정할 수 있습니다.

<https://www.intel.com> > fpga > software > downloads

**FPGA Software Download Center - Intel**

**Download** Intel® Quartus® Prime Software, DSP Builder, Simulation Tools, HLS, ... Intel® Quartus® Prime **Lite** Edition Design Software Version 21.1.1 for Linux.

Intel Quartus Prime Lit...

2 Results

Filter by

Intel Quartus Software

CLEAR

Intel® Quartus® Prime Design Software  
(2)

Additional Software

CLEAR

Intel® FPGA Programming Software (2)

Intel® FPGA Simulation Tools (2)

EXPAND ALL

COLLAPSE ALL

**Intel Quartus Prime Lite Edition** Access to additional

Title **Design Software Version 20.1 for Windows**

Intel® Quartus® Prime Lite Edition Design Software Version 20.1 for Windows

(검증된 버전)

Intel® Quartus® Prime Lite Edition Design Software Version 20.1.1 for Windows

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Quartus 설치하기

링크 바로가기

<https://www.intel.com/content/www/us/en/software-kit/661019/intel-quartus-prime-lite-edition-design-software-version-20-1-for-windows.html>

ID	Date	Version
661019	6/14/2020	20.1

버전 확인할 것!

### Downloads

Multiple Download

Individual Files

Additional Software

Copyright Licensed Source

### Multiple Download

Intel® Quartus® Prime Lite Edition Software (Device support included)

Download  
Quartus-lite-20.1.0.711-windows.tar

Size: 5.9 GB

설치까지 충분한 용량 필요

SHA1: 101faf57b86e4737f40379339b9b7eed4dc8672d

#### What's Included?

- \*\* Nios® II EDS on Windows requires Ubuntu 18.04 LTS on Windows Subsystem for Linux (WSL), which requires a manual installation.
- \*\* Nios® II EDS requires you to install an Eclipse IDE manually.

### 1. 디지털 회로설계

Tool 설치 & 환경  
셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

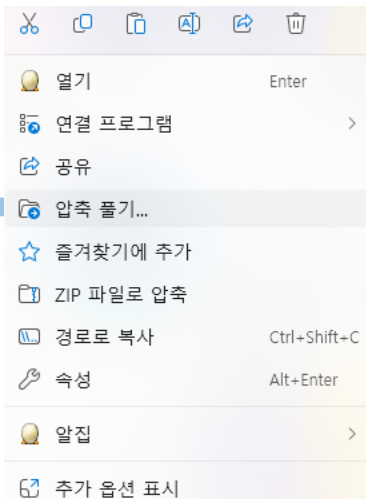
## Quartus 설치하기



.tar 파일 압축 풀기를 위해  
반디집 or 알집 설치



Quartus-lite-20.1  
.0.711-windows.  
tar



압축 풀기



Quartus-lite-20.1  
.0.711-windows

오늘



components

오래 전



readme.txt

Setup.bat 실행



setup.bat

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Quartus 설치하기

Installing Quartus Prime Lite Edition (Free) 20.1.0.711

### Select Components



Select the components you want to install

☒ Quartus Prime Lite Edition (Free)

- ☒ Quartus Prime (includes Nios II EDS) (931.3MB)
- ☒ Quartus Prime Help (508.4MB)
- ☒ Devices
  - ☐ Arria II (536.5MB)
  - ☒ Cyclone IV (516.3MB)
  - ☐ Cyclone 10 LP (293.5MB)
  - ☐ Cyclone V (1434.3MB)
  - ☐ MAX II/V (13.1MB)
  - ☒ MAX 10 FPGA (360.3MB)
- ☒ ModelSim - Intel FPGA Starter Edition (Free) (4318.8MB)
- ☐ ModelSim - Intel FPGA Edition (4318.8MB)

Installs MAX 10 FPGA device support. (360.3MB)

필수 선택!

InstallBuilder

< Back

Next >

Cancel

### 1. 디지털 회로설계

Tool 설치 & 환경  
셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과  
확인하기

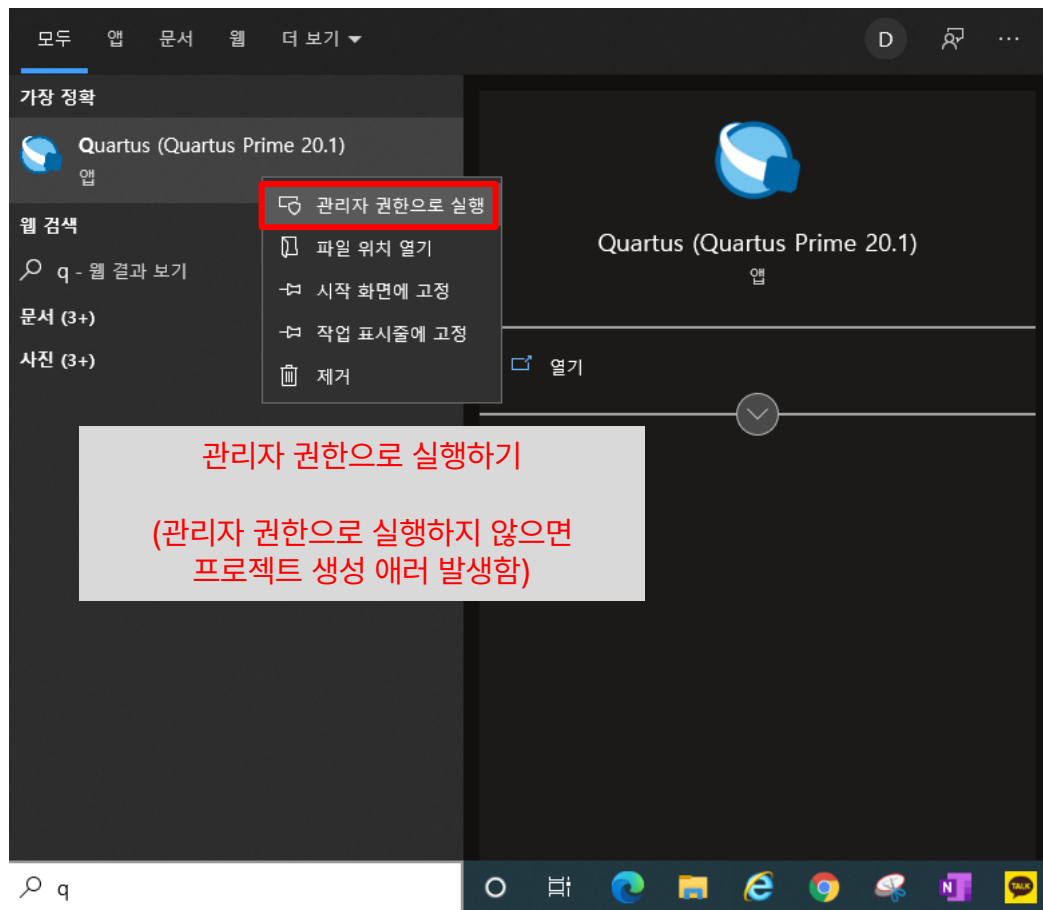
### 4. Timing 분석과

Data-path Delay  
확인하기

### 5. Power 분석과

Static vs. Dynamic  
Power 확인하기

## Quartus 설치하기



### 1. 디지털 회로설계

Tool 설치 & 환경  
셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

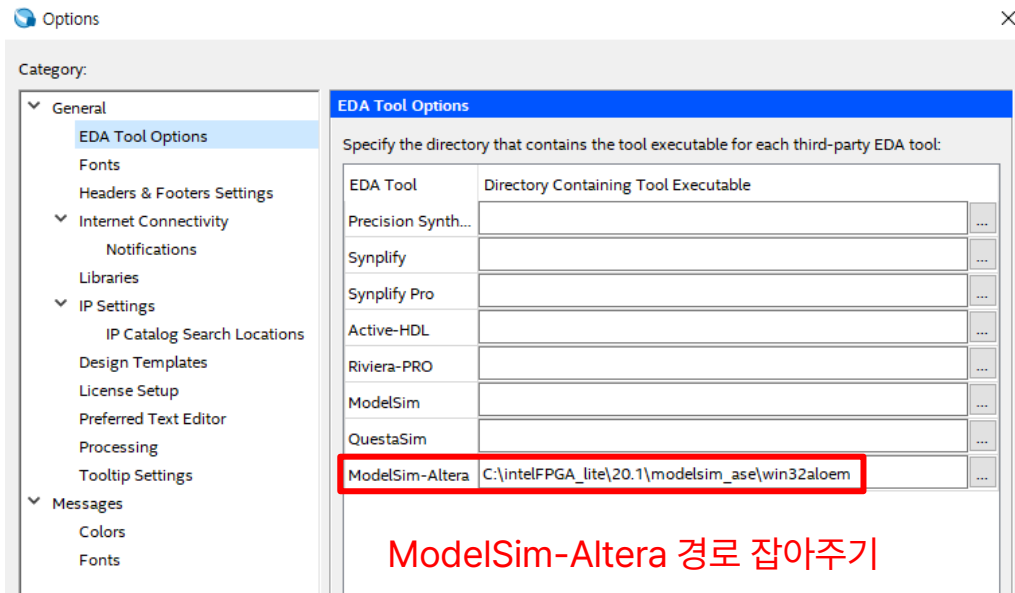
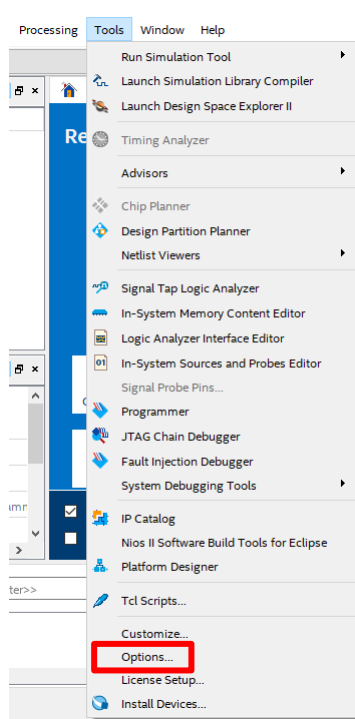
시뮬레이션 결과  
확인하기

### 4. Timing 분석과

Data-path Delay  
확인하기

### 5. Power 분석과

Static vs. Dynamic  
Power 확인하기



\* 경로는 개인마다 다르기 때문에 직접 확인할 것!



### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

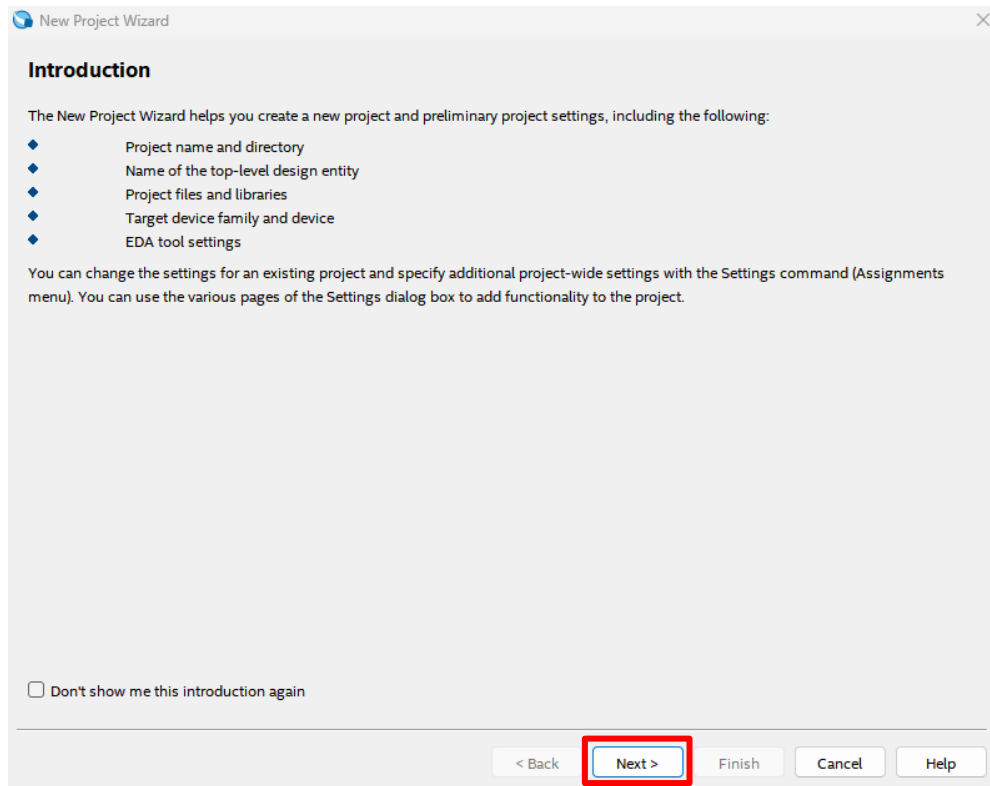
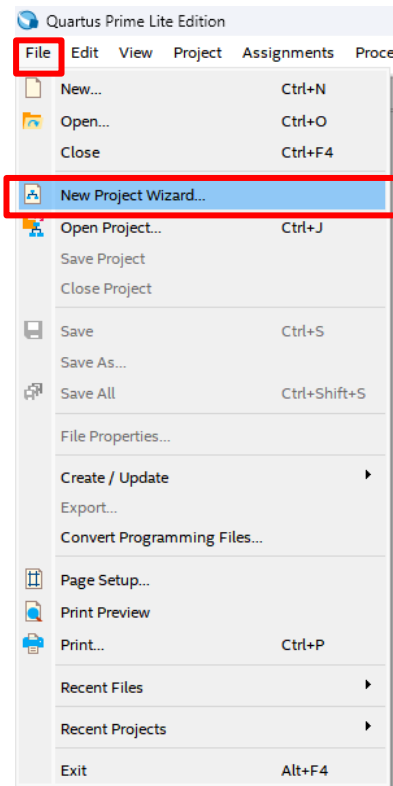
확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Quartus 프로젝트 생성하기



### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

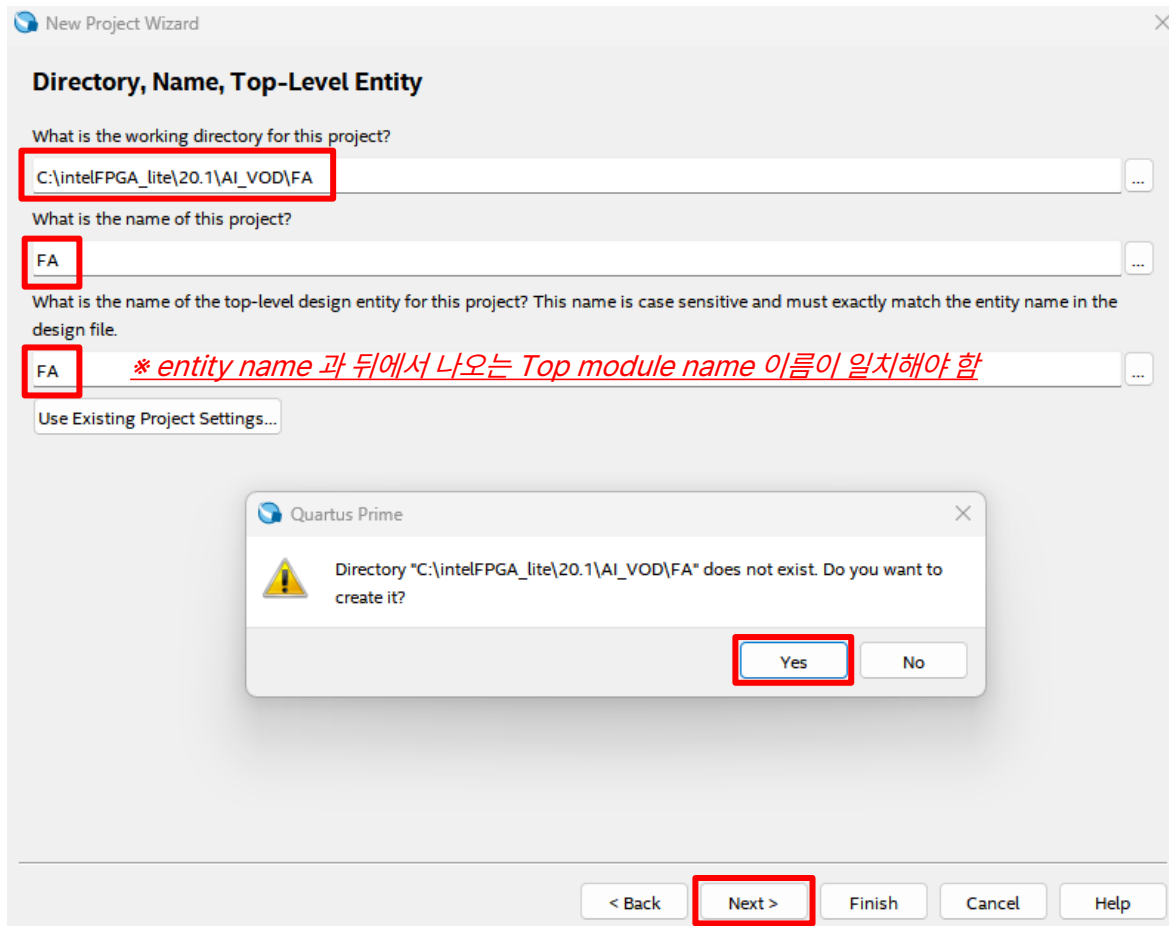
확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Quartus 프로젝트 생성하기



## 1. 디지털 회로설계

Tool 설치 &amp; 환경

셋팅하기

## 2. Verilog code 작성과

Logic 합성해보기

## 3. Test-bench 작성과

시뮬레이션 결과

확인하기

## 4. Timing 분석과

Data-path Delay

확인하기

## 5. Power 분석과

Static vs. Dynamic

Power 확인하기

New Project Wizard

### Family, Device & Board Settings

Device Board

Select the family and device you want to target for compilation.  
You can install additional device support with the Install Devices command on the Tools menu.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the [Device Support List](#) webpage.

Device family

Family: Cyclone IV E

Device: All 싸이클론 4에서 E 또는 GE 무관함

Target device

☒ Auto device selected by the Fitter

☐ Specific device selected in 'Available devices' list

☐ Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Core speed grade: Any

Name filter:

☒ Show advanced devices

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit
EP4CE6E22A7	1.2V	6272	92	92	276480	30

< Back Next > Finish Cancel Help

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Quartus 프로젝트 생성하기

New Project Wizard

### EDA Tool Settings

Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/Synth...	<None>	<None>	<input type="checkbox"/> Run this tool automatically to synthesize the current design
Simulation	ModelSim-Altera	Verilog HDL	<input type="checkbox"/> Run gate-level simulation automatically after compilation
Board-Level	Timing	<None>	ModelSim-Altera 가 설치되어 있어야 함
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

< Back   **Next >**   Finish   Cancel   Help

## Chapter 03.

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

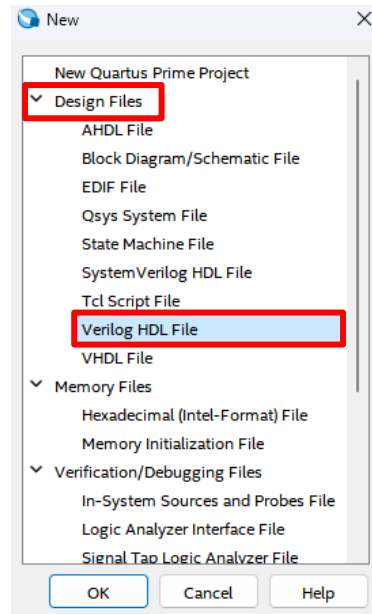
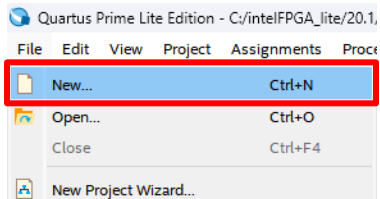
확인하기

### 5. Power 분석과

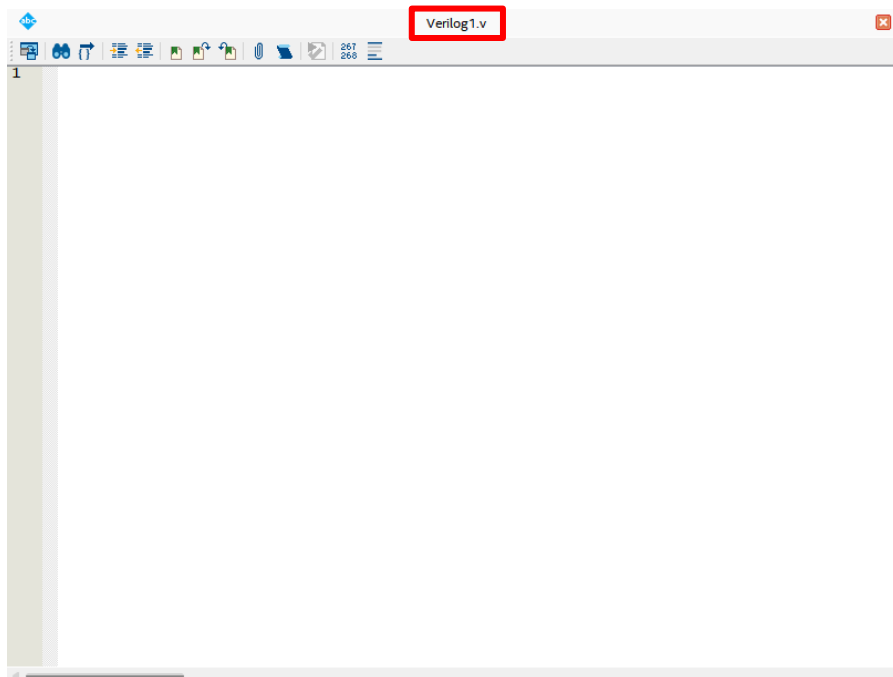
Static vs. Dynamic

Power 확인하기

## Quartus 프로젝트 생성하기



확장자가 .v 임을 확인할 것!



## Chapter 03.

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

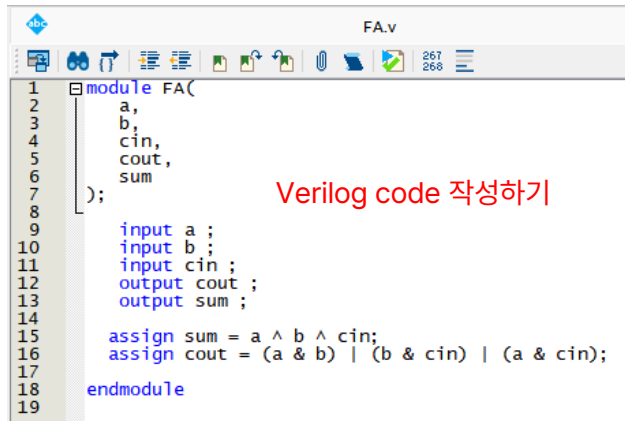
확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Quartus Verilog 코딩 및 Logic 합성하기



```
1 module FA(  
2     a,  
3     b,  
4     cin,  
5     cout,  
6     sum  
7 );  
8  
9     input a ;  
10    input b ;  
11    input cin ;  
12    output cout ;  
13    output sum ;  
14  
15    assign sum = a ^ b ^ cin;  
16    assign cout = (a & b) | (b & cin) | (a & cin);  
17  
18 endmodule  
19
```

Verilog code 작성하기

```
module FA(  
    a,  
    b,  
    cin,  
    cout,  
    sum  
);
```

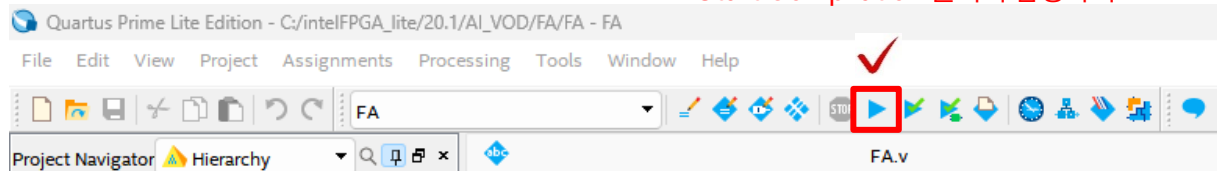
(Verilog 문법은 다음 챕터에서 배웁니다)

```
input a ;  
input b ;  
input cin ;  
output cout ;  
output sum ;
```

```
assign sum = a ^ b ^ cin;  
assign cout = (a & b) | (b & cin) | (a & cin);
```

```
endmodule
```

Start Compilation 눌러서 합성하기



### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

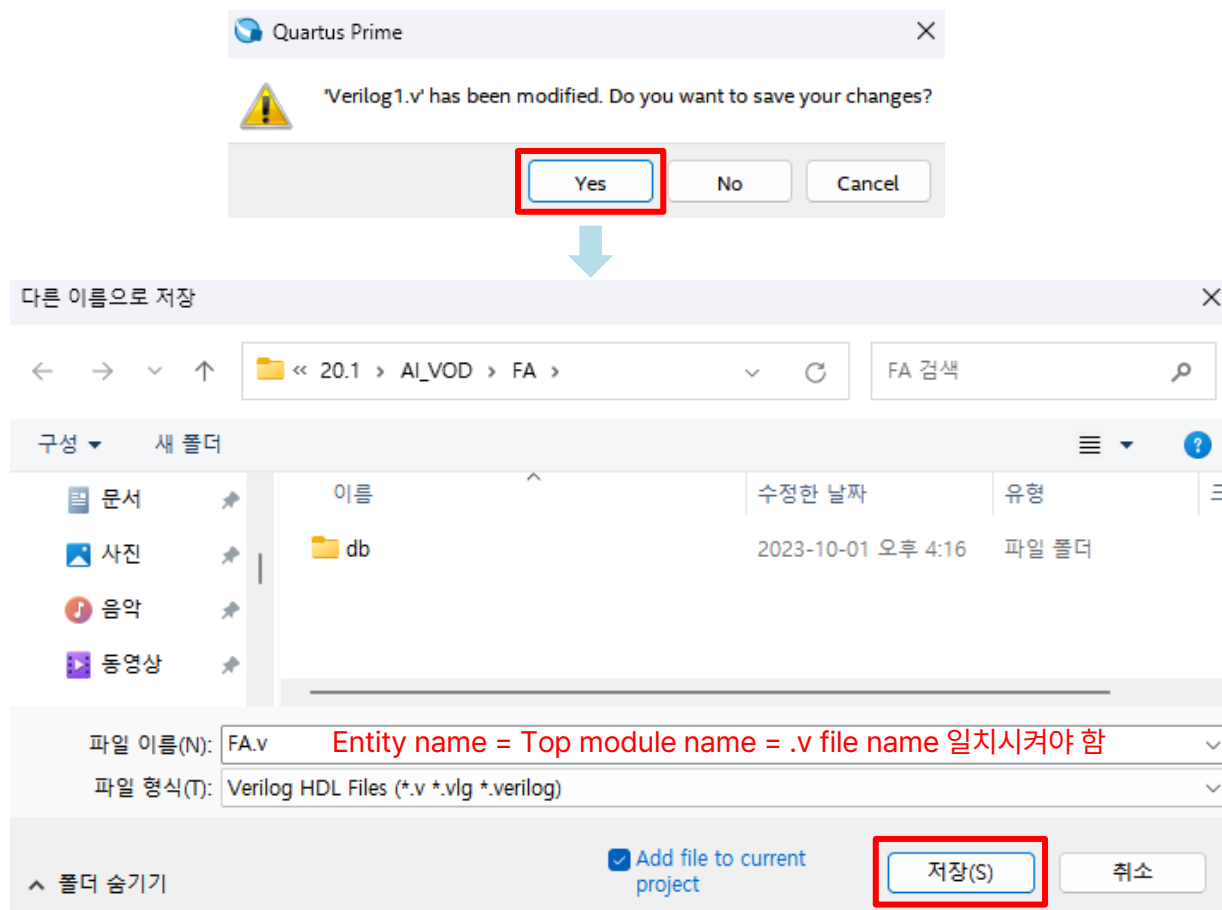
확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Quartus Verilog 코딩 및 Logic 합성하기



## Chapter 03.

### 1. 디지털 회로설계

Tool 설치 & 환경  
셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과  
확인하기

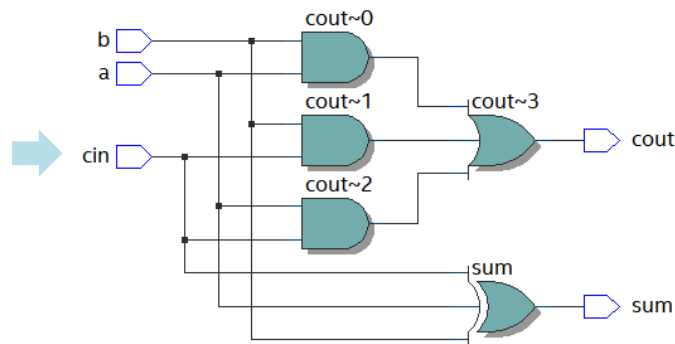
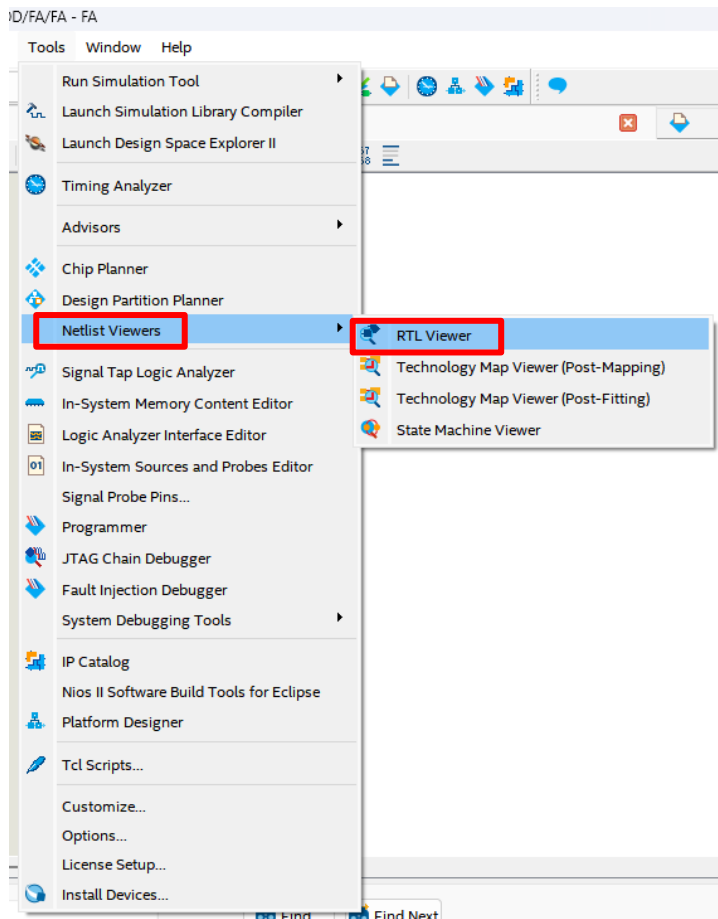
### 4. Timing 분석과

Data-path Delay  
확인하기

### 5. Power 분석과

Static vs. Dynamic  
Power 확인하기

## Quartus Verilog 코딩 및 Logic 합성하기



합성 결과 확인



## Chapter 03.

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Quartus Test-bench 작성 및 실행하기

```
1  module tb_FA;
2      reg a, b, cin;
3      wire sum, cout;
4
5      FA module_FA(a, b, cin, cout, sum);
6
7      initial begin
8          $monitor("At time %0t: a=%b b=%b, cin=%b, sum=%b, carry=%b", $time, a,b,cin,cout,sum);
9          a = 0; b = 0; cin = 0; #1;
10         a = 0; b = 0; cin = 1; #1;
11         a = 0; b = 1; cin = 0; #1;
12         a = 0; b = 1; cin = 1; #1;
13         a = 1; b = 0; cin = 0; #1;
14         a = 1; b = 0; cin = 1; #1;
15         a = 1; b = 1; cin = 0; #1;
16         a = 1; b = 1; cin = 1;
17     end
18 endmodule
```

일단 작성해서 '저장' 누르기 (tb\_FA.v로 저장)

→ 자세한 설명은 실습에서!

```
module tb_FA;
    reg a, b, cin;
    wire sum, cout;

    FA module_FA(a, b, cin, cout, sum);

    initial begin
        $monitor("At time %0t: a=%b b=%b, cin=%b, sum=%b, carry=%b", $time, a,b,cin,cout,sum);
        a = 0; b = 0; cin = 0; #1;
        a = 0; b = 0; cin = 1; #1;
        a = 0; b = 1; cin = 0; #1;
        a = 0; b = 1; cin = 1; #1;
        a = 1; b = 0; cin = 0; #1;
        a = 1; b = 0; cin = 1; #1;
        a = 1; b = 1; cin = 0; #1;
        a = 1; b = 1; cin = 1;
    end
endmodule
```

**(Test-bench 작성 방법은 실습 시간에 배웁니다)**

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

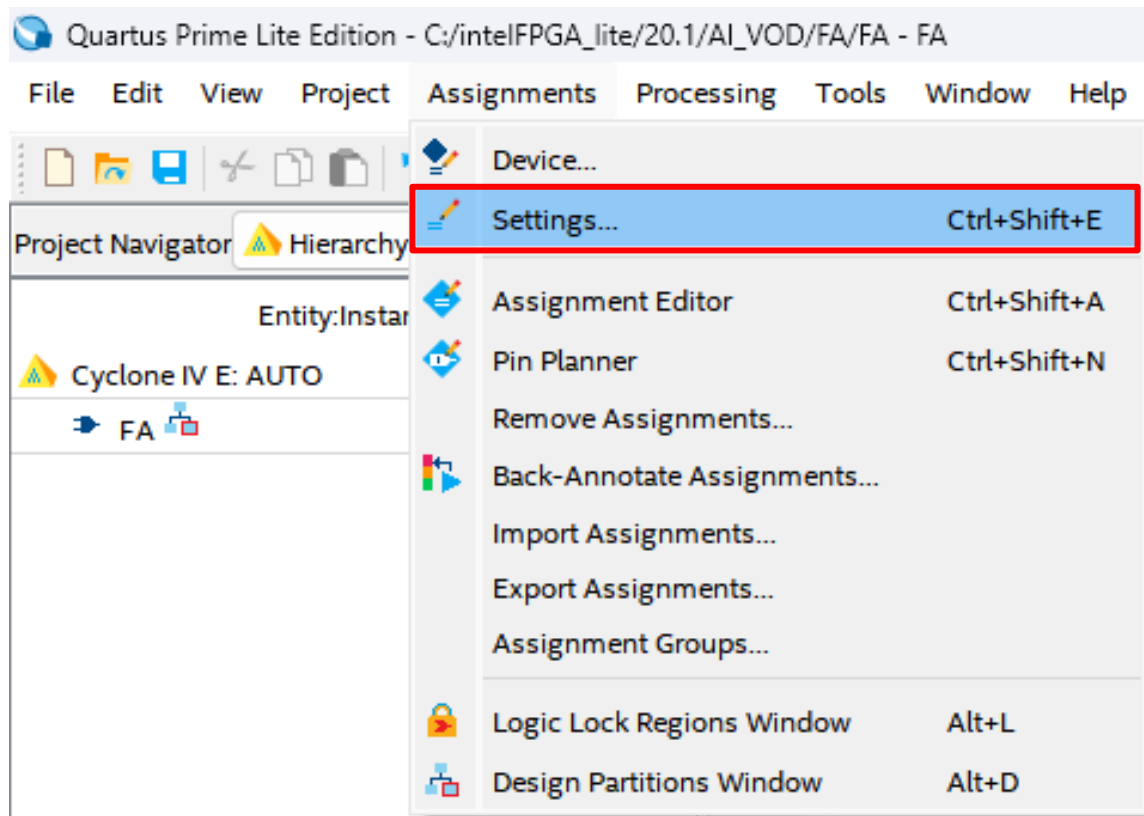
확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Quartus Test-bench 작성 및 실행하기



### 1. 디지털 회로설계

Tool 설치 & 환경  
셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과  
확인하기

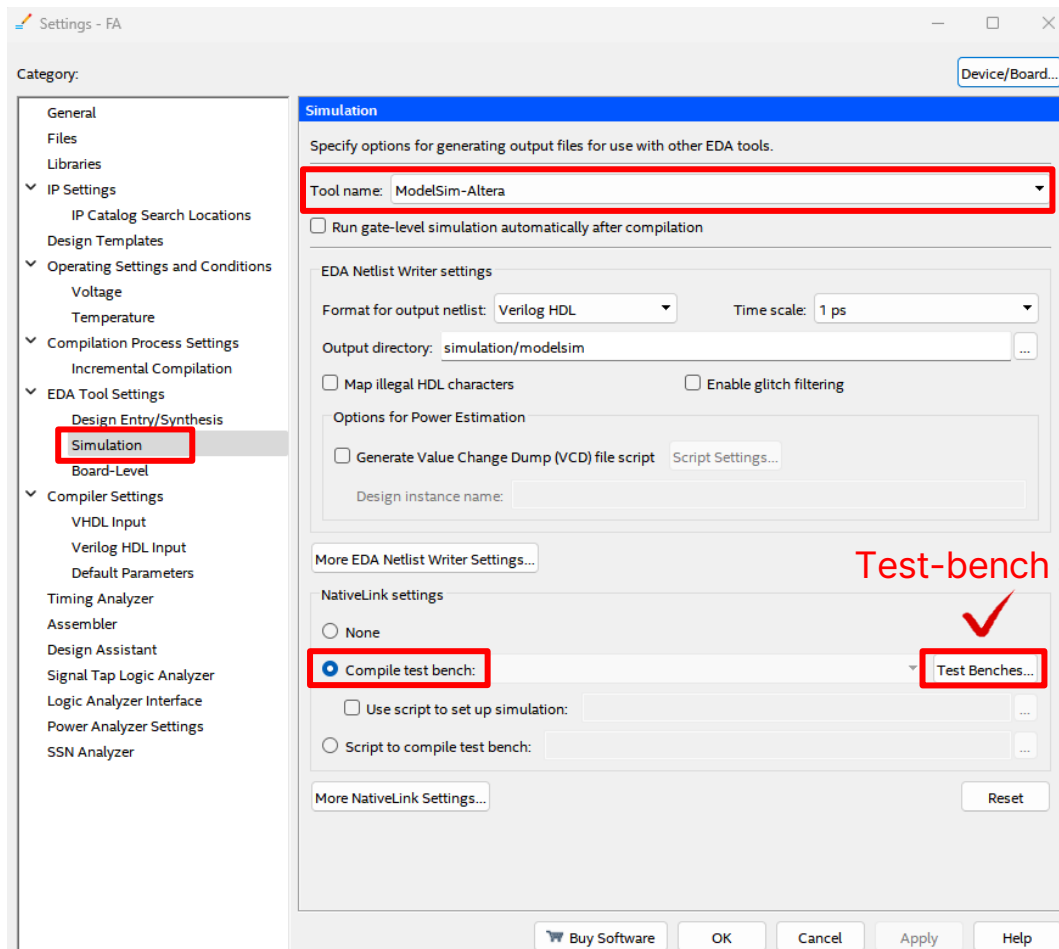
### 4. Timing 분석과

Data-path Delay  
확인하기

### 5. Power 분석과

Static vs. Dynamic  
Power 확인하기

## Quartus Test-bench 작성 및 실행하기



### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

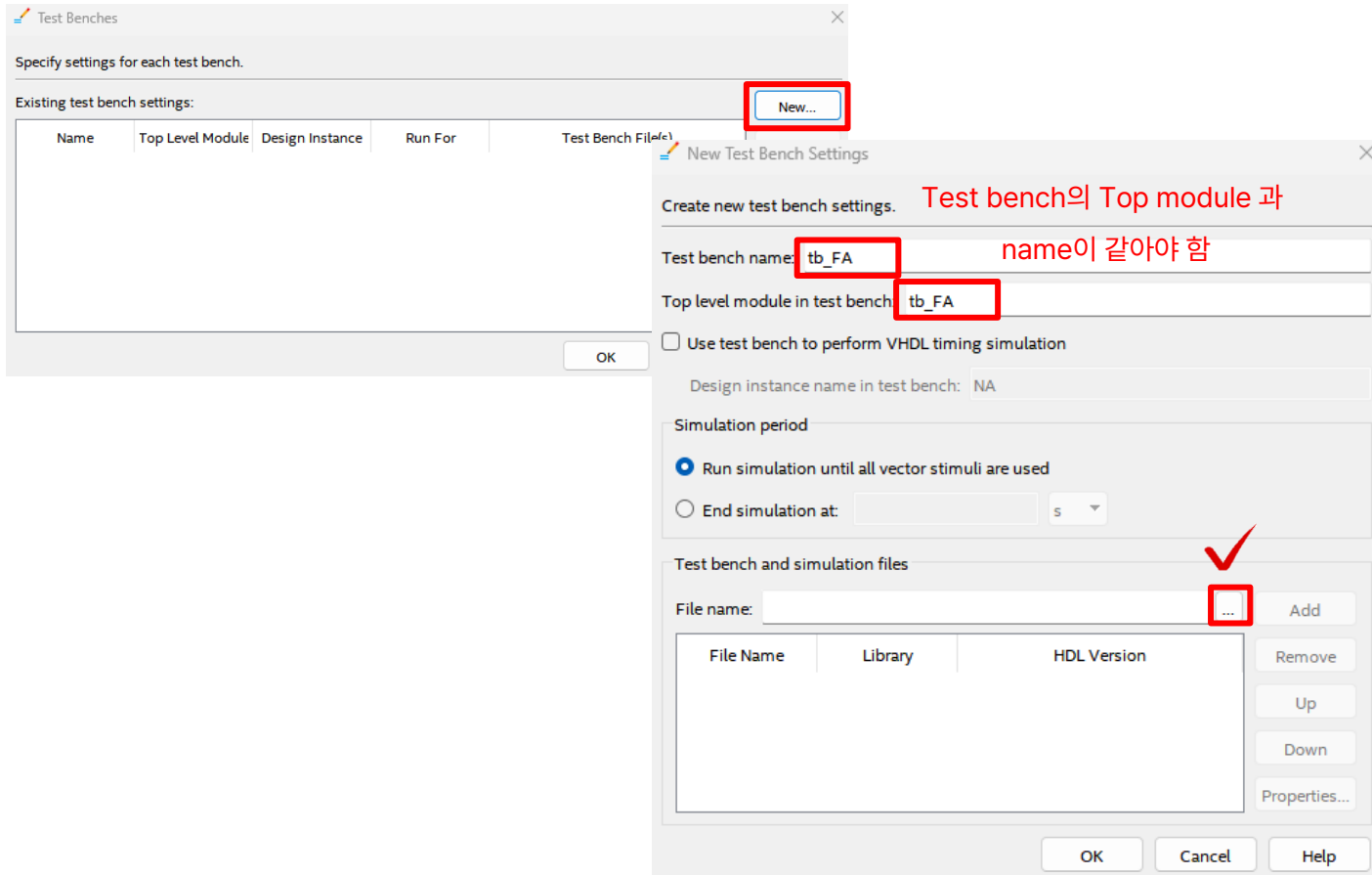
확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Quartus Test-bench 작성 및 실행하기



### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

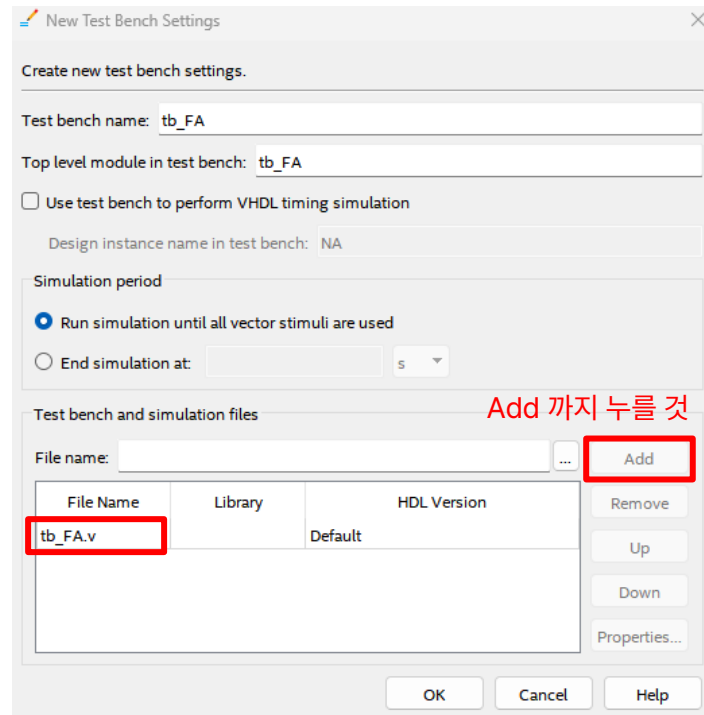
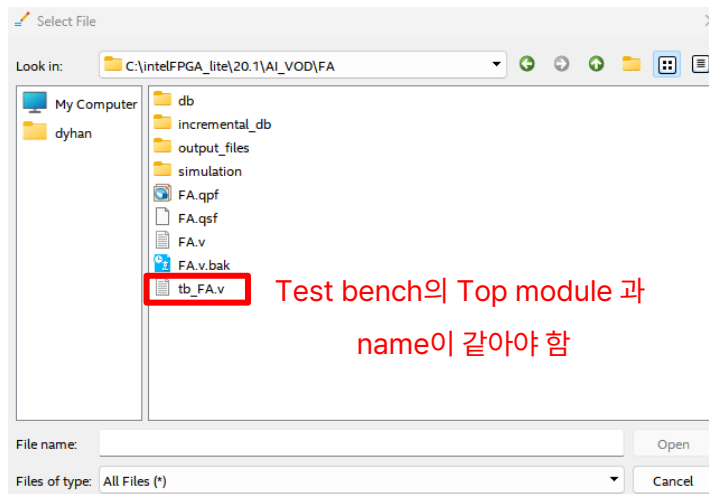
확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Quartus Test-bench 작성 및 실행하기



### 1. 디지털 회로설계

Tool 설치 & 환경  
셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과  
확인하기

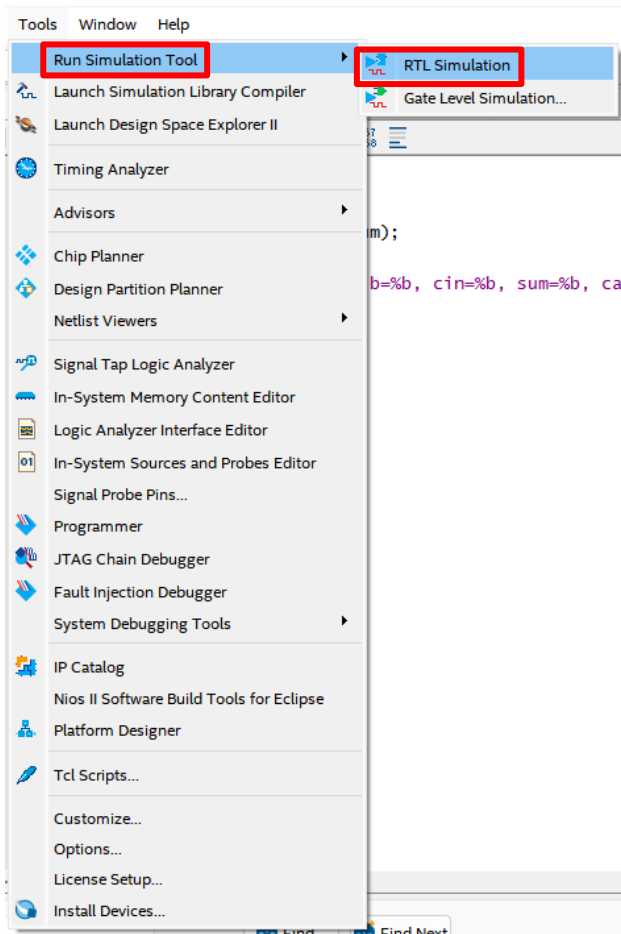
### 4. Timing 분석과

Data-path Delay  
확인하기

### 5. Power 분석과

Static vs. Dynamic  
Power 확인하기

## Quartus Test-bench 작성 및 실행하기



시뮬레이션 실행

## 1. 디지털 회로설계

Tool 설치 &amp; 환경

셋팅하기

## 2. Verilog code 작성과

Logic 합성해보기

## 3. Test-bench 작성과

시뮬레이션 결과

확인하기

## 4. Timing 분석과

Data-path Delay

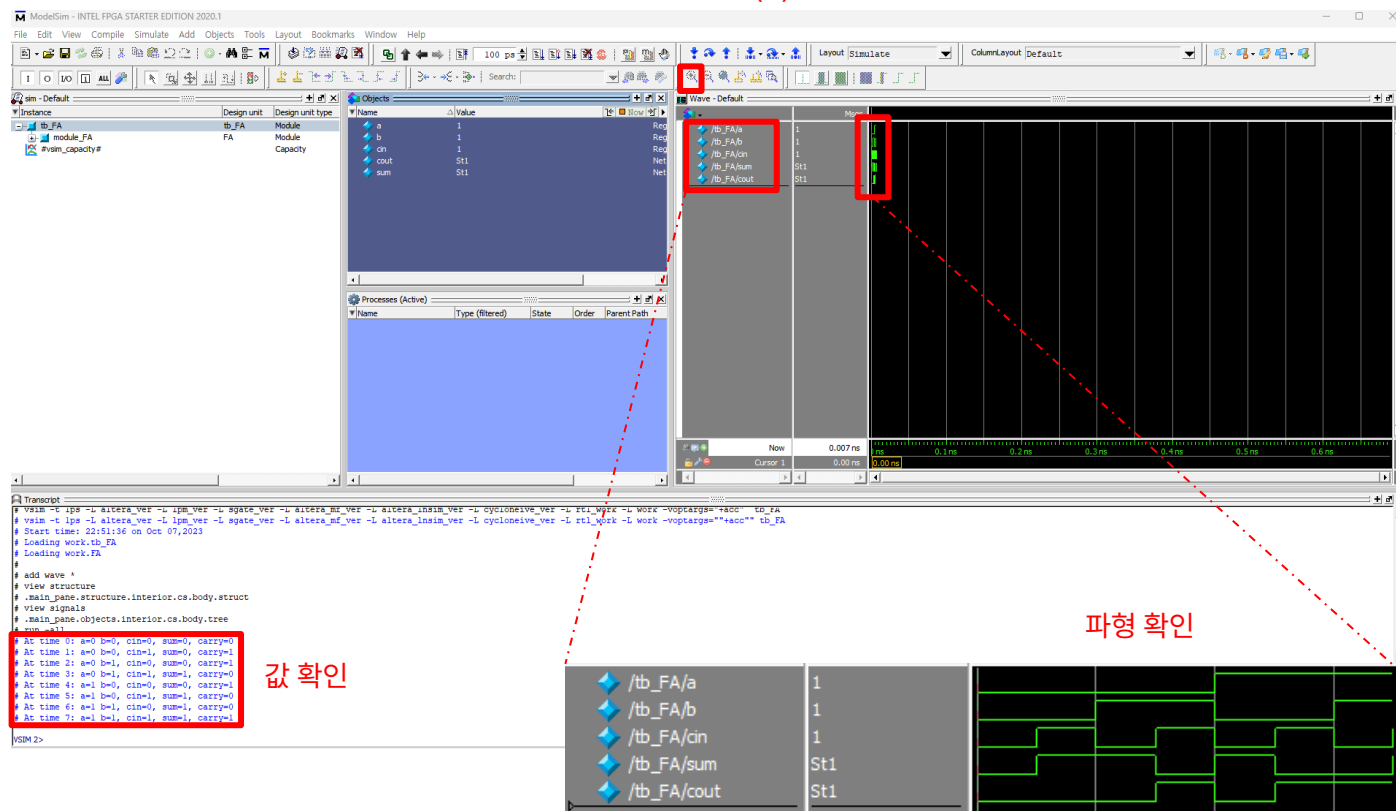
확인하기

## 5. Power 분석과

Static vs. Dynamic

Power 확인하기

돋보기 (+) 버튼 또는 Ctrl + 마우스 휠 → 확대



## Chapter 03.

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Timing 분석과 Data-path Delay 확인하기

```
module FA(
```

```
    a,  
    b,  
    cin,  
    cout,  
    sum
```

```
);
```

```
    input a ;  
    input b ;  
    input cin ;  
    output cout ;  
    output sum ;
```

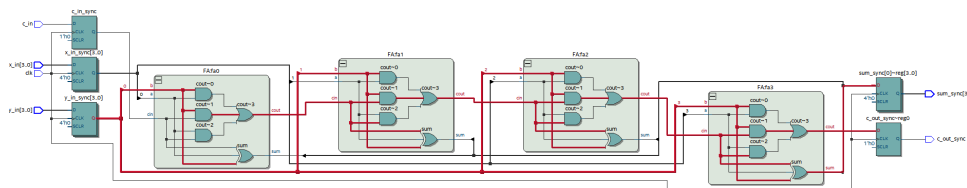
```
    assign sum = a ^ b ^ cin;  
    assign cout = (a & b) | (b & cin) | (a & cin);
```

```
endmodule
```

```
module FA_4bit(x_in,y_in,c_in,c_out_sync,sum_sync,clk);  
    input [3:0] x_in;  
    input [3:0] y_in;  
    input c_in, clk;  
    output [3:0] sum_sync;  
    output c_out_sync;  
    wire c1,c2,c3,c_out;  
    wire [3:0] sum;  
    reg [3:0] x_in_sync, y_in_sync, sum_sync;  
    reg c_in_sync, c_out_sync;  
    FA fa0(x_in_sync[0],y_in_sync[0],c_in_sync,c1,sum_sync[0]);  
    FA fa1(x_in_sync[1],y_in_sync[1],c1,c2,sum_sync[1]);  
    FA fa2(x_in_sync[2],y_in_sync[2],c2,c3,sum_sync[2]);  
    FA fa3(x_in_sync[3],y_in_sync[3],c3,c_out,sum_sync[3]);
```

```
    always @(posedge clk) begin  
        x_in_sync <= x_in;  
        y_in_sync <= y_in;  
        c_in_sync <= c_in;  
        sum_sync <= sum;  
        c_out_sync <= c_out;  
    end
```

```
endmodule
```





## Chapter 03.

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

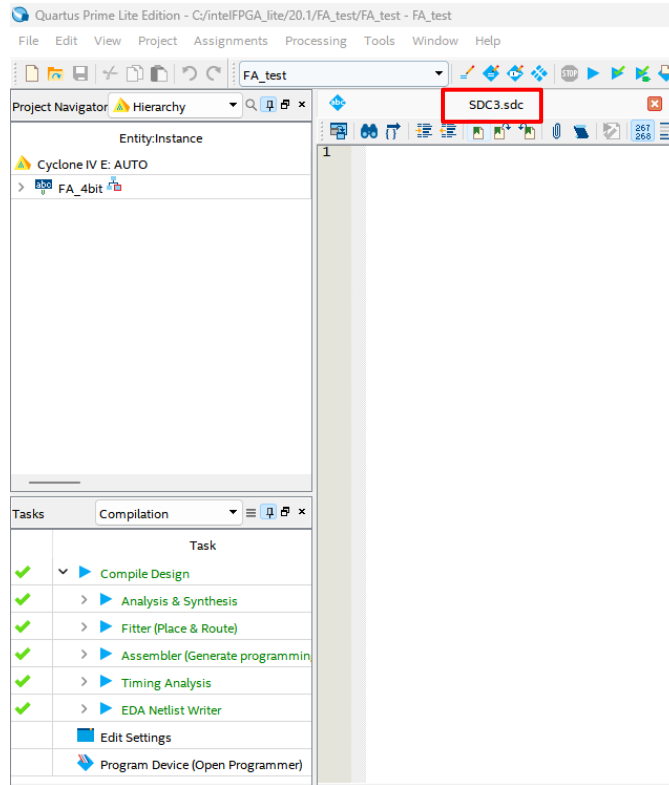
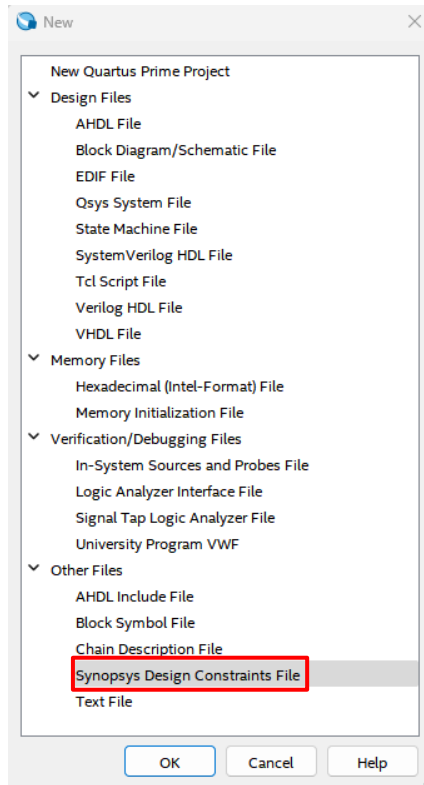
확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Timing 분석과 Data-path Delay 확인하기



## Chapter 03.

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

확인하기

### 5. Power 분석과

Static vs. Dynamic

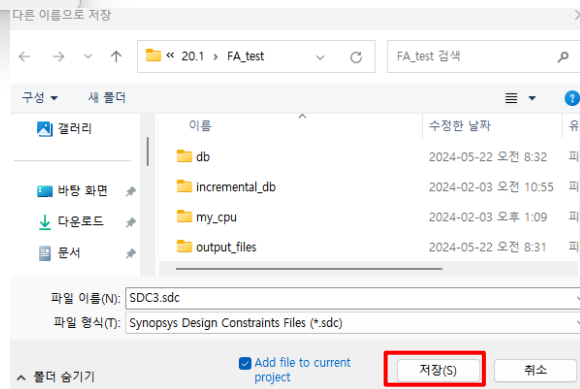
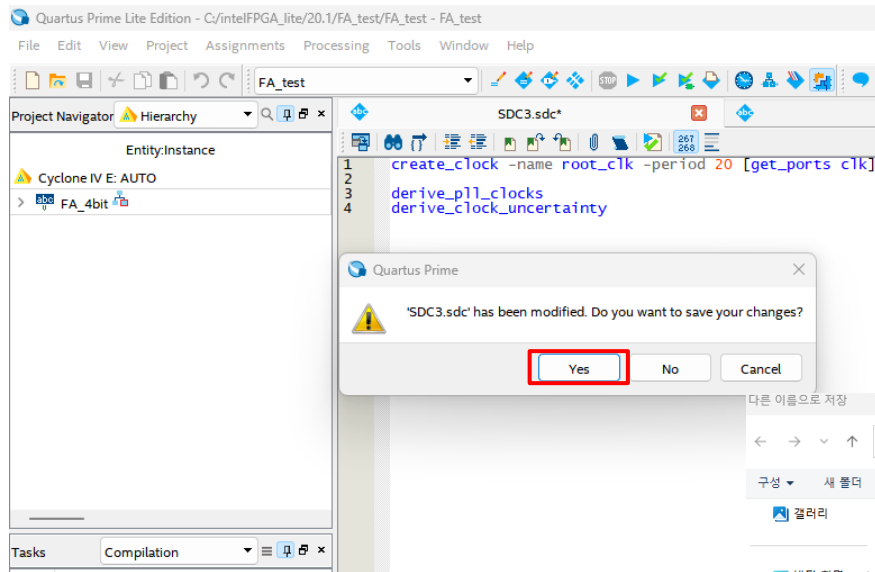
Power 확인하기

## Timing 분석과 Data-path Delay 확인하기

```
create_clock -name root_clk -period 20 [get_ports clk]
```

```
derive_pll_clocks
```

```
derive_clock_uncertainty
```



### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

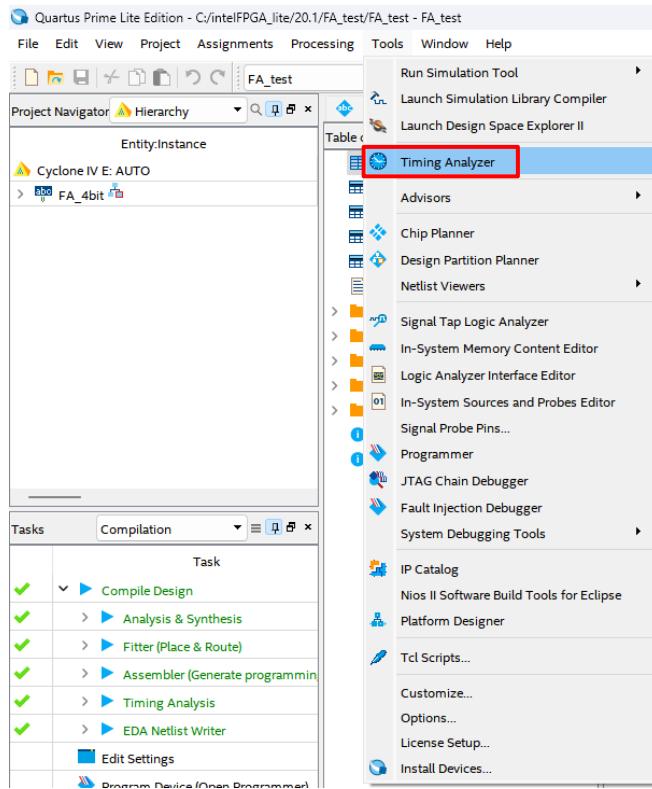
확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Timing 분석과 Data-path Delay 확인하기



## 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

## 2. Verilog code 작성과

Logic 합성해보기

## 3. Test-bench 작성과

시뮬레이션 결과

확인하기

## 4. Timing 분석과

Data-path Delay

확인하기

## 5. Power 분석과

Static vs. Dynamic

Power 확인하기

Timing Analyzer - C:/intelFPGA\_lite/20.1/FA\_test/FA\_test - FA\_test

File View Netlist Constraints Reports Script Tools Window Help

Search altera.com

Set Operating Conditions

No timing corners available

Report

Report not available

Tasks

- Report Timing Closure Recomm...
- Macros
  - Report All Summaries
  - Report Top Failing Paths
  - Report All I/O Timings
  - Report All Core Timings**
  - Create All Clock Histograms
- Write SDC File...

Getting Started

### Welcome to the Quartus Prime Timing Analyzer

The Quartus Prime Timing Analyzer is a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in your design using industry-standard constraint, analysis, and reporting methodology. You can use the Timing Analyzer to constrain, run, and view results for all timing paths in your design. The Timing Analyzer offers the following features:

- Report Pane** Lists generated report panels.
- Tasks Pane** Lists common tasks you can perform. Double-click a command to start the step in the flow.
- View Pane** Displays selected report panels. Split the view pane into subpanes by dragging the splitter control in the upper right corner of the pane.
- Console** Displays SDC and Tcl commands executed by the GUI or command line.

Console

```

- Type "exit" to exit.
- Type "help" to view a list of quartus Prime Tcl packages.
- Type "help <package name>" to view a list of Tcl commands available for the specified quartus Prime Tcl package.
- Type "help -tcl" to get an overview on quartus Prime Tcl usages.
*****
tcl> project_open -force "c:/intelFPGA_lite/20.1/FA_test/FA_test.qpf" -revision FA_test
tcl>
  
```

0% 00:00:00 Ready

## Chapter 03.

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Timing 분석과 Data-path Delay 확인하기

The screenshot displays the Timing Analyzer interface for a project named 'FA\_test'. The 'Slow 1200mV 85C Model' is selected. The main table lists timing paths with columns for Slack, From Node, To Node, Launch Clock, Latch Clock, Relationship, Clock Skew, and Data Delay. The first row is highlighted in red.

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	17.228	y_in_sync[2]	sum_sync[3]~reg0	root_clk	root_clk	20.000	-0.064	2.703
2	17.230	y_in_sync[2]	c_out_sync~reg0	root_clk	root_clk	20.000	-0.064	2.701
3	17.536	x_in_sync[2]	sum_sync[3]~reg0	root_clk	root_clk	20.000	-0.064	2.395
4	17.538	x_in_sync[2]	c_out_sync~reg0	root_clk	root_clk	20.000	-0.064	2.393
5	17.567	y_in_sync[0]	sum_sync[3]~reg0	root_clk	root_clk	20.000	-0.064	2.364
6	17.569	y_in_sync[0]	c_out_sync~reg0	root_clk	root_clk	20.000	-0.064	2.362
7	17.843	x_in_sync[0]	sum_sync[3]~reg0	root_clk	root_clk	20.000	-0.064	2.088
8	17.845	x_in_sync[0]	c_out_sync~reg0	root_clk	root_clk	20.000	-0.064	2.086
9	17.902	c_in_sync	sum_sync[3]~reg0	root_clk	root_clk	20.000	-0.064	2.029
10	17.904	c_in_sync	c_out_sync~reg0	root_clk	root_clk	20.000	-0.064	2.027
11	18.327	y_in_sync[0]	sum_sync[1]~reg0	root_clk	root_clk	20.000	-0.064	1.604
12	18.351	y_in_sync[0]	sum_sync[2]~reg0	root_clk	root_clk	20.000	-0.064	1.580
13	18.436	y_in_sync[1]	sum_sync[3]~reg0	root_clk	root_clk	20.000	-0.064	1.495
14	18.438	y_in_sync[1]	c_out_sync~reg0	root_clk	root_clk	20.000	-0.064	1.493
15	18.455	y_in_sync[2]	c_out_sync~reg0	root_clk	root_clk	20.000	-0.064	1.476
16	18.461	y_in_sync[2]	sum_sync[3]~reg0	root_clk	root_clk	20.000	-0.064	1.470
17	18.467	y_in_sync[2]	sum_sync[2]~reg0	root_clk	root_clk	20.000	-0.064	1.464
18	18.507	x_in_sync[1]	sum_sync[3]~reg0	root_clk	root_clk	20.000	-0.064	1.424
19	18.509	x_in_sync[1]	c_out_sync~reg0	root_clk	root_clk	20.000	-0.064	1.422
20	18.603	x_in_sync[0]	sum_sync[1]~reg0	root_clk	root_clk	20.000	-0.064	1.328
21	18.603	x_in_sync[2]	c_out_sync~reg0	root_clk	root_clk	20.000	-0.064	1.328
22	18.609	x_in_sync[2]	sum_sync[3]~reg0	root_clk	root_clk	20.000	-0.064	1.322
23	18.610	x_in_sync[2]	sum_sync[2]~reg0	root_clk	root_clk	20.000	-0.064	1.321

The console window shows the following commands and output:

```
tcl> update_timing_netlist
tcl> Deriving clock uncertainty. Please refer to report_sdc in the Timing Analyzer to see clock uncertainties.
tcl> qsta_utility::generate_all_core_timing_reports "Report Timing (Core)" 1000 -multi_corner
> 1 Report Timing: Found 37 setup paths (0 violated). Worst case slack is 17.228
> 1 Report Timing: Found 37 hold paths (0 violated). Worst case slack is 0.376
> 1 No fmax paths to report
> 1 No fmax paths to report
tcl>
```

## Chapter 03.

### 1. 디지털 회로설계

Tool 설치 & 환경  
셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과  
확인하기

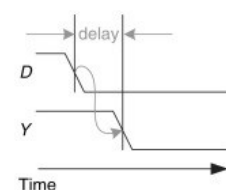
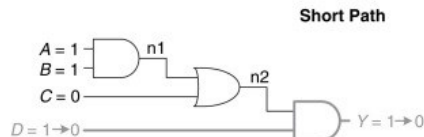
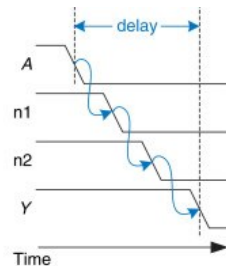
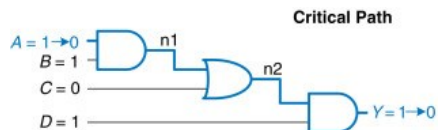
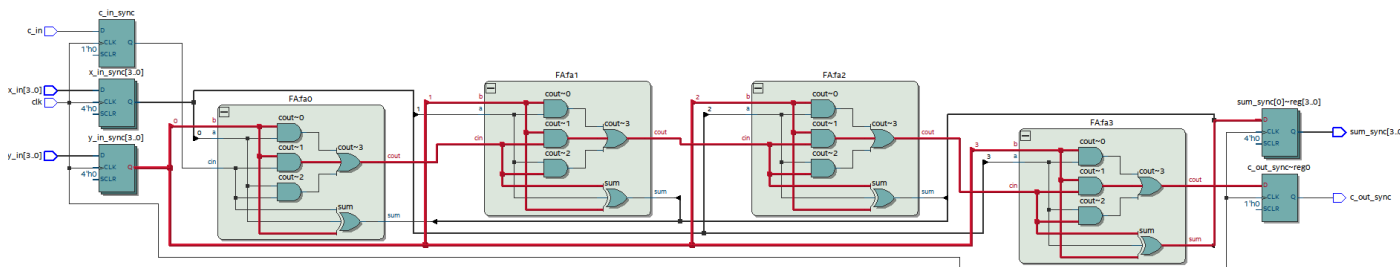
### 4. Timing 분석과

Data-path Delay  
확인하기

### 5. Power 분석과

Static vs. Dynamic  
Power 확인하기

## Timing 분석과 Data-path Delay 확인하기



### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

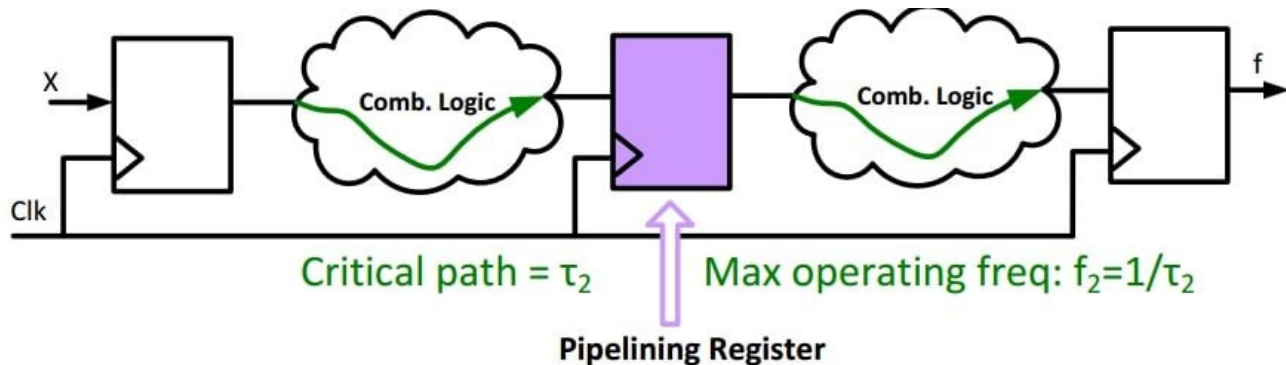
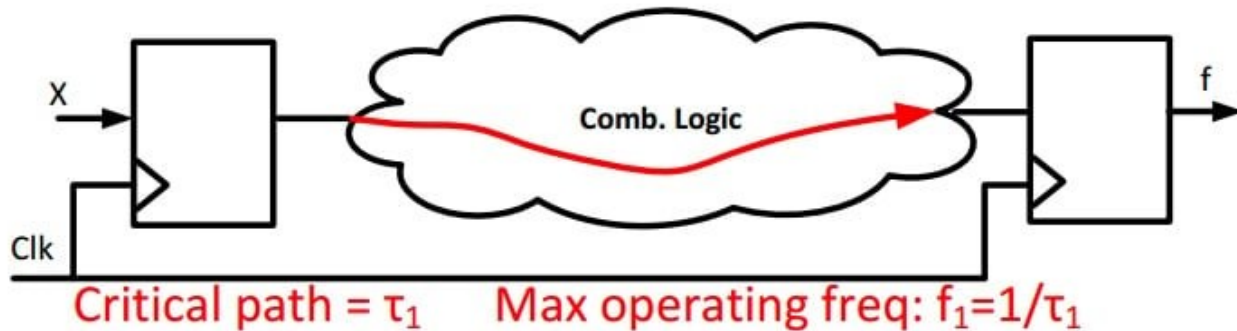
확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Timing 분석과 Data-path Delay 확인하기



### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

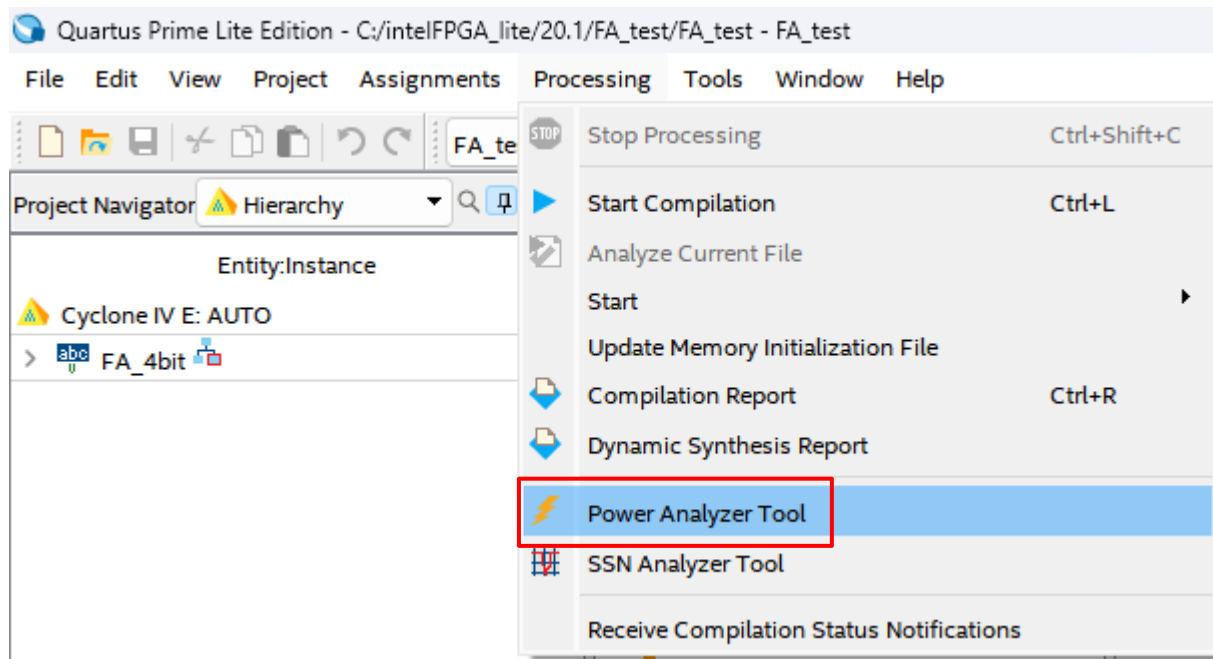
확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Power 분석과 Static vs. Dynamic Power 확인하기





## Chapter 03.

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

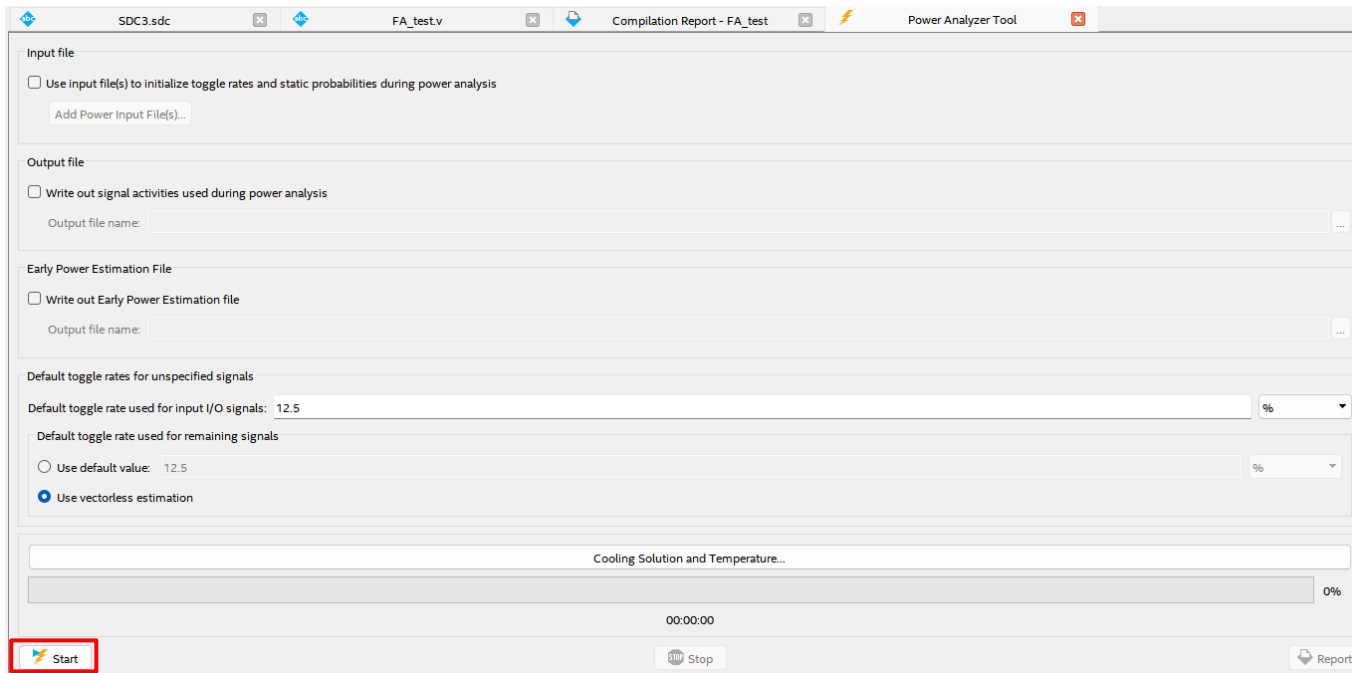
확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Power 분석과 Static vs. Dynamic Power 확인하기



### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

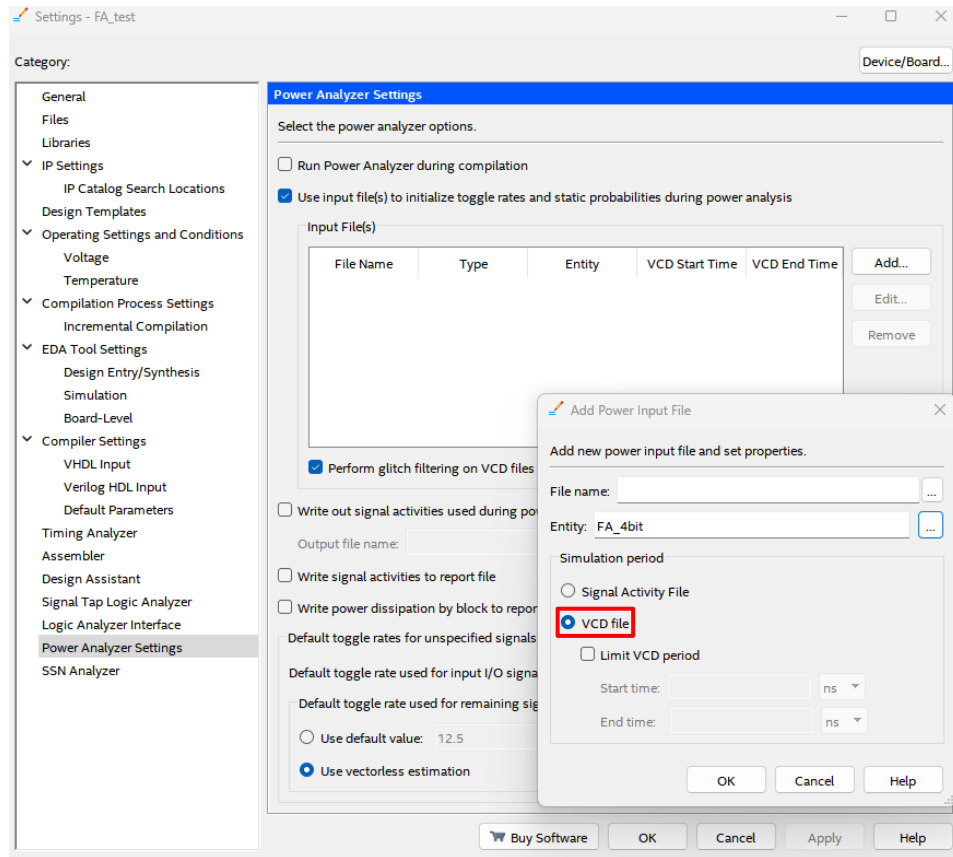
확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Power 분석과 Static vs. Dynamic Power 확인하기



## Chapter 03.

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

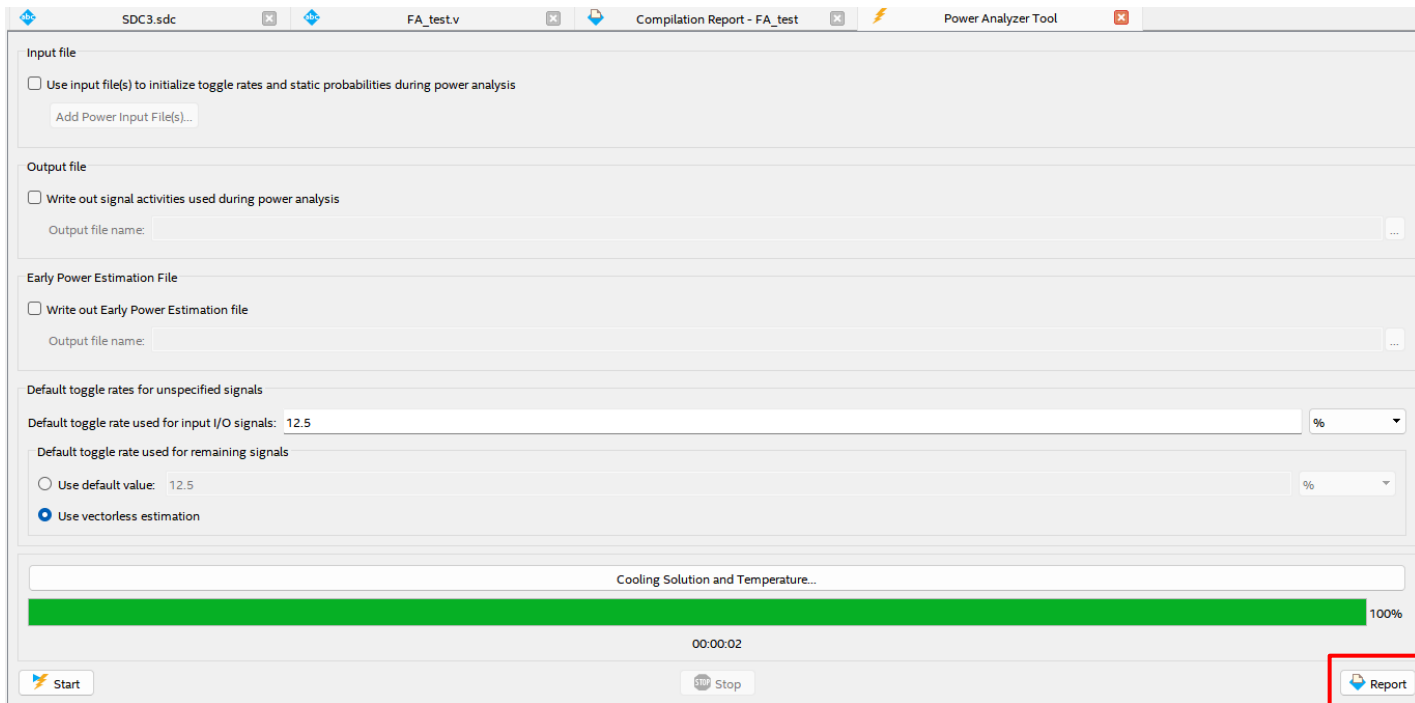
확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Power 분석과 Static vs. Dynamic Power 확인하기



### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay


확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Power 분석과 Static vs. Dynamic Power 확인하기

Power Analyzer Summary	
 <<Filter>>	
Power Analyzer Status	Successful - Wed May 22 08:58:13 2024
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	FA_test
Top-level Entity Name	FA_4bit
Family	Cyclone IV E
Device	EP4CE6E22C6
Power Models	Final
Total Thermal Power Dissipation	63.31 mW
Core Dynamic Thermal Power Dissipation	0.71 mW
Core Static Thermal Power Dissipation	42.82 mW
I/O Thermal Power Dissipation	19.78 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

### 1. 디지털 회로설계

Tool 설치 & 환경

셋팅하기

### 2. Verilog code 작성과

Logic 합성해보기

### 3. Test-bench 작성과

시뮬레이션 결과

확인하기

### 4. Timing 분석과

Data-path Delay

확인하기

### 5. Power 분석과

Static vs. Dynamic

Power 확인하기

## Power 분석과 Static vs. Dynamic Power 확인하기

