ISE Foundation 5.2i Manual
for FPGA design

Insight Korea Xilinx FAE Team
2003. 6. 20

ISE Foundation 5.2i(이하 ISE5.2i)는 현재 Xilinx에서 제공하는 통합 설계 Design Software로서 FPGA나 CPLD를 쉽게 디자인할 수 있도록 Design Entry, Synthesis, 그리고 Verification , 기본 Simulation을 할 수 있는 통합 환경을 제공합니다.

주의) ISE 5.2i 는 Windows XP and Windows 2000 이상의 OS에서만 지원합니다.
(Windows98에서는 ISE5.2i 을 사용하실 수 없습니다. Xilinx Tools 4.2i 이하 버전을 이용하셔야 합니다.)

참고) 1. ISE에서5.2i는 XC4000 계열의 디바이스를 지원하지 않습니다.
(XC4000, Spartan / SpartanXL ex: XCS30, XC30XL)
자세한 S/W의 지원사항에 대한 내용은 아래의 파일을 참조하시기 바랍니다.

http://www.xilinx.com/publications/matrix/softmatrix.pdf

이번 Tutorial에서는 Virtex2 Family인 XC2V40-4FG256C Target board상에서 DPRAM을 이용한 LAB을 설명합니다.

이 ISE 5.2i Manual은 VHDL로 작성되었으며, VHDL을 사용하시는 분을 기준으로 만들었습니다.

목　　　차

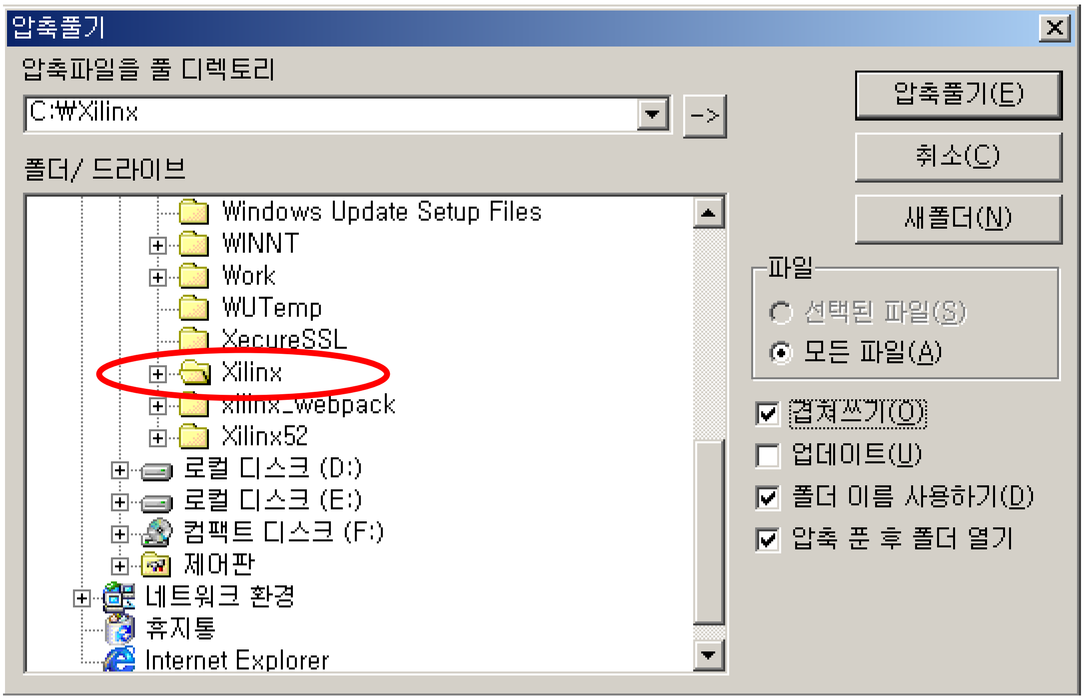
1. ISE5.2i 설치하기

ISE5.2i는 기본적으로 2장의 CD-ROM 과 1개의 ServicePack 파일, ZIP으로 압축된 5x_ip_update.zip으로 구성되어 있습니다.

순서는1. CD-ROM 2장을 순서대로 설치 후

2. ServicePack3 파일을 실행시키고(2003년 6월 20dlf현재)

3. 5x_ip_updata.zip 파일을 아래와 같이 C: \ Xilinx \ 디렉토리에 모든 파일 겹쳐 쓰기를 합니다.



< 그림 1 >

주의) 지금까지 설치한 파일 들에는 Simulation Tools인 Modelsim이 기본적으로 설치되지는 않습니다. Xilinx에서 무료로 제공하는 Modelsim MXE version "MXE_5.6e_Full_installer.exe"을 내려 받아 추가 설치하거나, 개인적으로 마련하셔야 합니다. 무료로 제공되는 Modelsim MXE version에는 Synthesis 기능에 제약이 있습니다.

자세한 내용은 담당 FAE에게 문의하시기 바랍니다.

2. ISE5.2i 실행하기

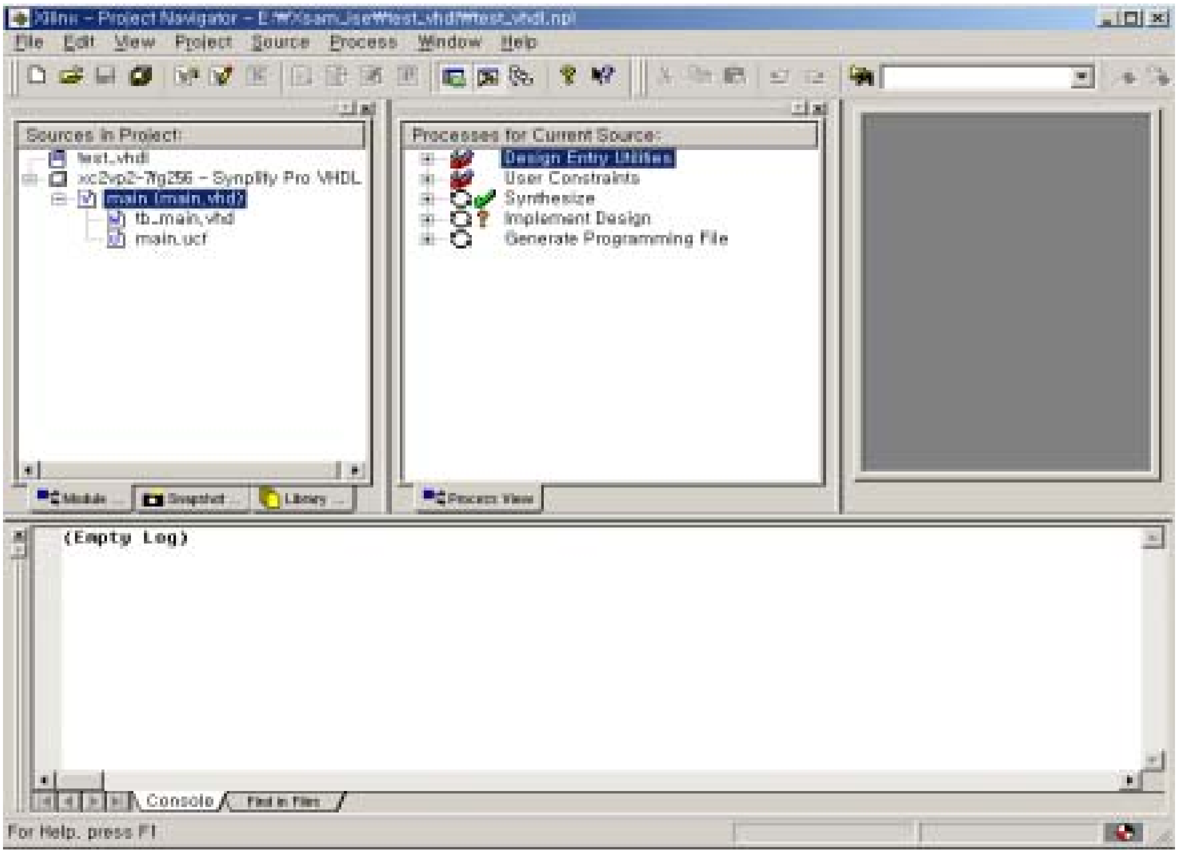
ISE 5.2i 를 실행하기 위하여 아래와 같이 바탕화면에 있는 Project Navigator를 클릭합니다. .

< 그림 2 >   바탕 화면의 Project Navigator

3. 프로젝트 작성

ISE 5.2i을 실행하기 위하여 Project Navigator를 Double 클릭하게 되면 아래와 같은 Project Navigator라는 window가 나타납니다. .

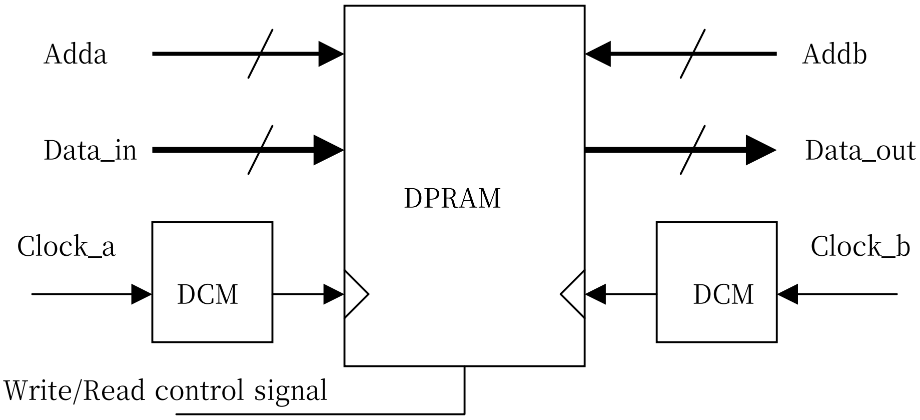
이 통합환경은 주로 네 가지 창으로 구성되어있다. 프로젝트 관리 및 각종소스데이타를 관리하는 **Source in Project** 와 그때 사용된 소스의 Process의 상태를 나타내는 **Process for Current Source** 와 각종 명령 시 진행사항을 나타내는 **Console**창과 각종 소스 코드를 display 해주는 **Editor** 창으로 이루어 있습니다.



< 그림 3 > Project Navigator window

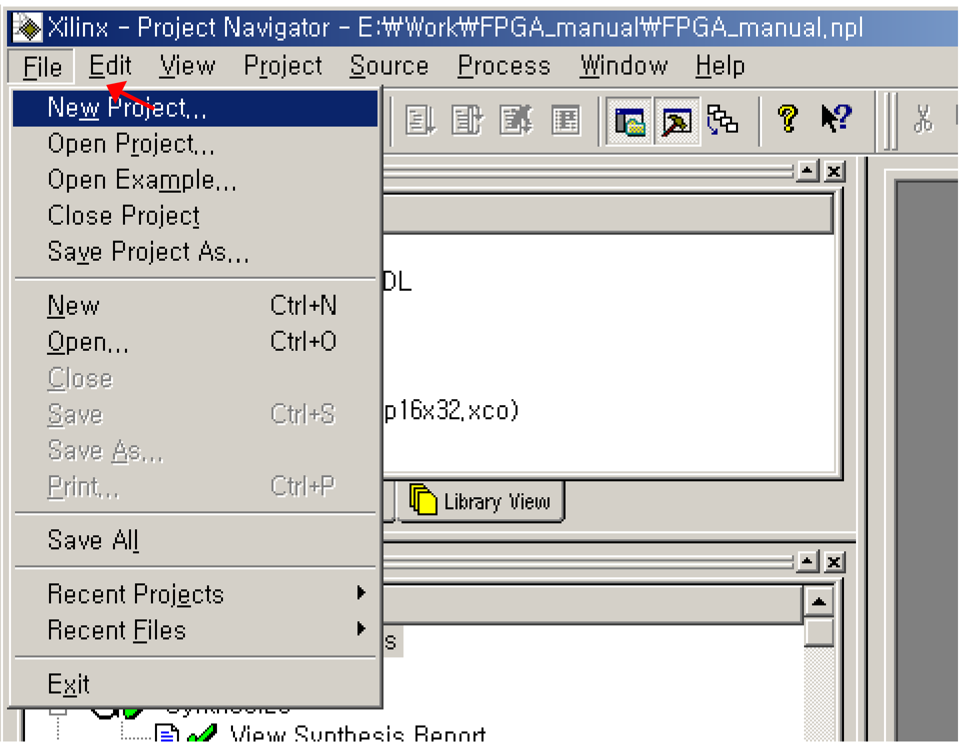
3.1 Making New project

Virtex2 디바이스 내에 있는 메모리를 이용하여 8bit * 16 Dual Port Block Memory 만들어 보는 LAB입니다. 이 LAB의 Block diagram을 살펴봅니다. 2개의 다른 Clock 신호가 들어오며 Dual Block Memory는 Write only port와 Read only port를 가지게 됩니다.

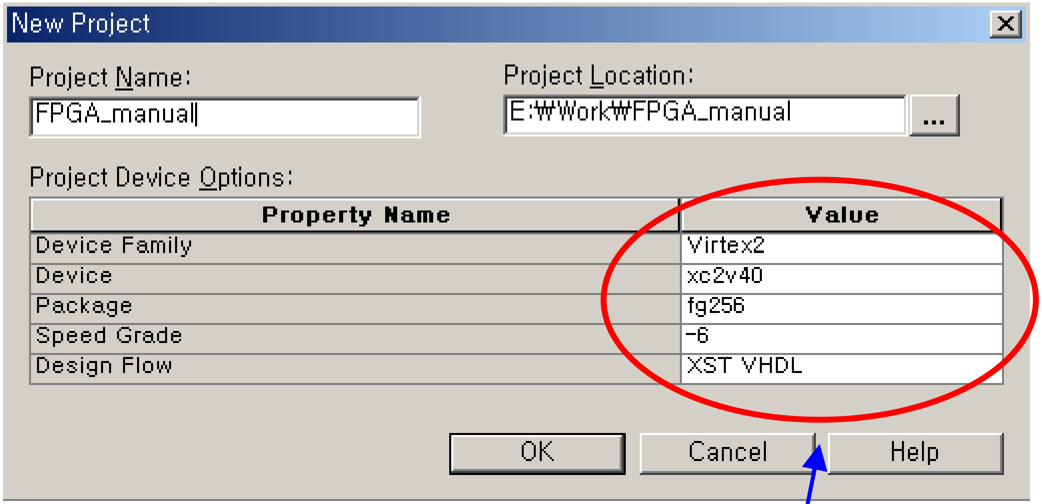


지금부터 위와 같은 내용의 새로운 Project를 ISE 5.2i를 이용하여 만들어 봅니다.

새로운 프로젝트를 만들려면 Project Navigator 에서 File > New Project를 클릭하면 아래와 같은 <그림 4> New Project 창이 나타납니다.



< 그림 4 > Project Navigator 에서 New Project 생성



< 그림 5 > New Project 창

* 사용하려는 Device와 내용이 *
정확히 일치하게 작성

구성을 살펴보면

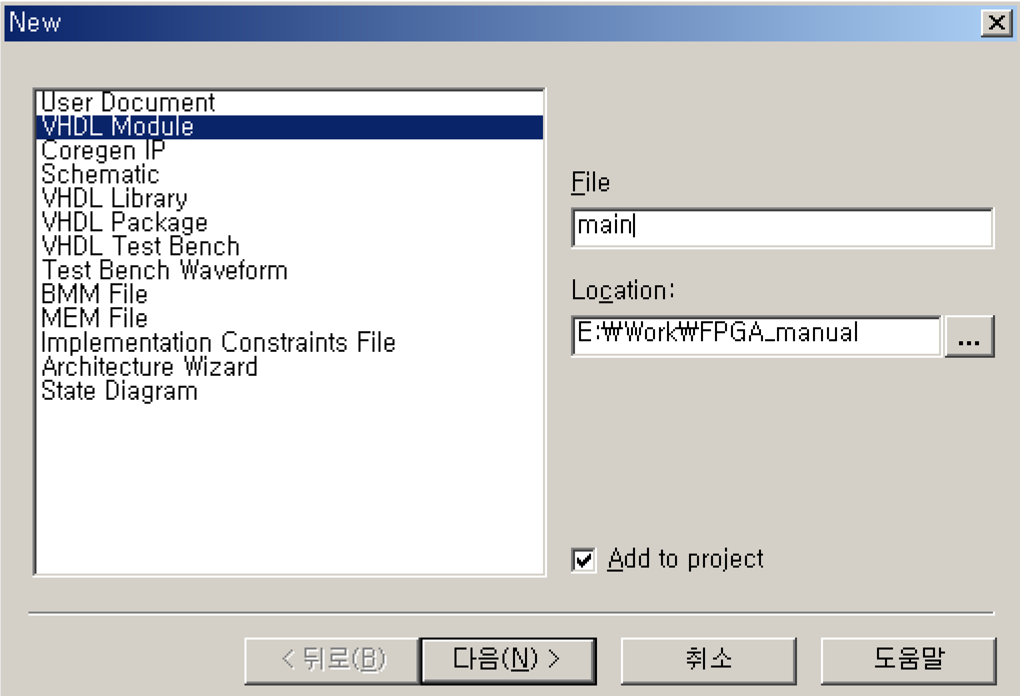
- Project Name : 작성할 프로젝트 이름
- Project Location : 작성할 프로젝트의 Working Directory 를 지정
- Device Family : 사용할 Device Family 선택
- Device : 사용될 Device Package 선택
- Design Flow : 사용될 Synthesis tool 선택

위와 같이 각 항목에 맞는 내용을 선택합니다.
<참조> XST 는 Xilinx Synthesis Tool 입니다. 만약 PC에 Simplify가 설치되어 있다면, 이것을 이용하여 Synthesis를 하여도 됩니다.

New Project를 작성하고 OK버튼을 누르면 새로운 Project 의 구성이 완성됩니다.
앞으로 작성되는 모든 파일들은 Working Directory인 \Work\FPGA_manual Folder내에서 작업이 이루어집니다.

3.2 Making Program file

Project Navigator의 Menu에서 Project > New Source를 Click 하거나 Source in Project 창에서 마우스 오른쪽을 클릭하여 New source를 클릭하면 아래 같은 창이 뜹니다.



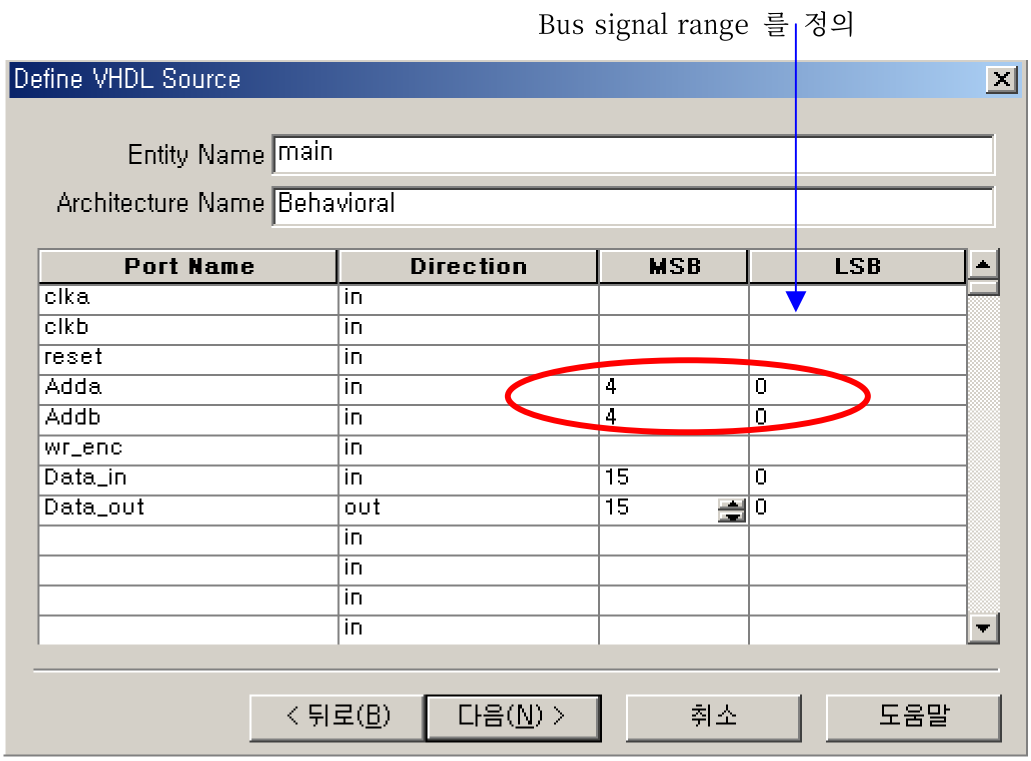
< 그림 6 > New Source 삽입 창

위와 같이 VHDL코드로 Source를 작성을 하기위해 VHDL Module을 선택하고 File란에 알맞은 .vhd file의 이름을 넣습니다. 여기서는 Main 이라고 하였습니다.

< 참조 > Source 의 종류를 간단히 살펴보면

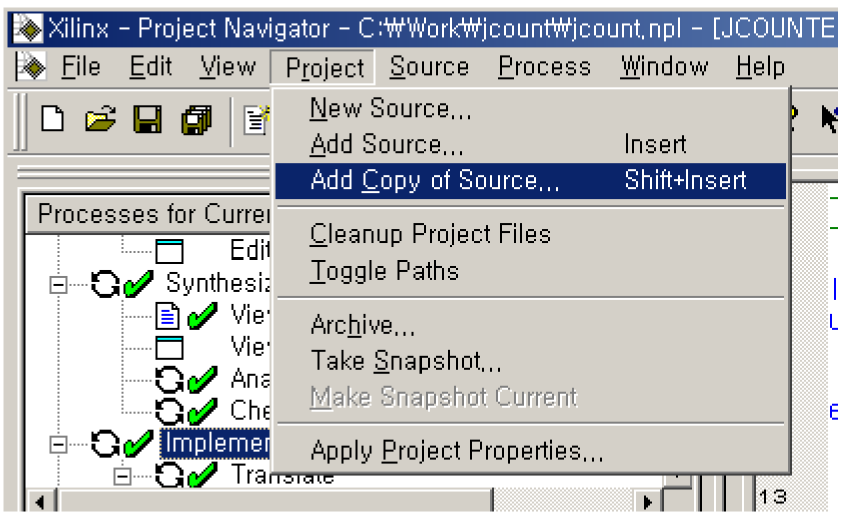
- User Document : 사용자가 프로젝트에 대한 전반적인 설명이 가능하게 함.
- VHDL Module : 작성할 Source 로직이 VHDL 코드일 경우 사용.
- Schematic : 작성할 Source 로직을 schematic으로 작성할 경우 사용.
- VHDL Test Bench : ModelSim을 이용하여 시뮬레이션을 하기위해 Test Bench를 작성할 경우 사용.
- Implementation Constraints File : UCF 파일을 작성할 경우 사용.

[다음]을 클릭하여 다음 단계로 넘어가면 아래와 같은 Define VHDL source 화면이 뜹니다. Port Name에 들어갈 이름과 Direction을 정하고 마침을 하면 main.vhd 라는 VHDL파일이 생성됩니다.



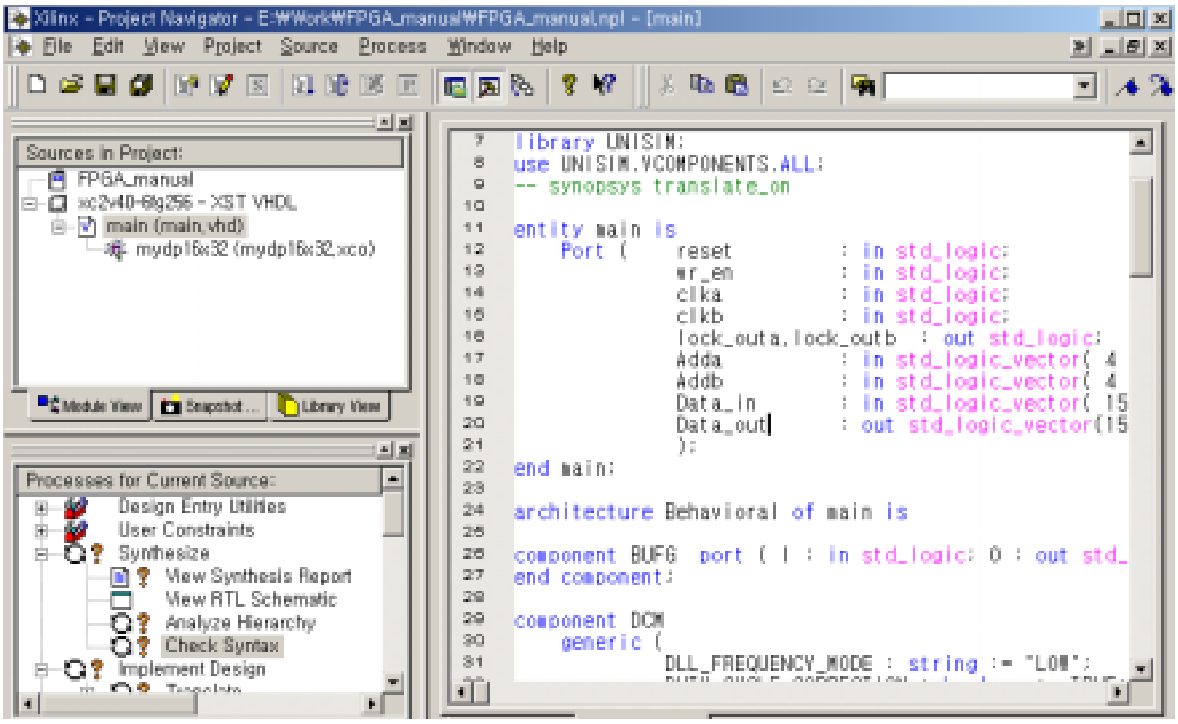
< 그림 7 > Define VHDL Source

만약 이전에 작성된 HDL Source를 이용할 경우에는 다음과 같은 메뉴를 이용합니다.
Project Navigator : Project -> Add copy of Source…를 이용합니다.



< 그림 8 > Add Copy of source

"New Source…" 를 이용한 Source의 작성을 위한 모든 준비 작업이 끝나면 아래의 그림처럼 Project Navigator가 준비됩니다.



< 그림 9 > New Source로 VHDL file 입력된 후 Project Navigator

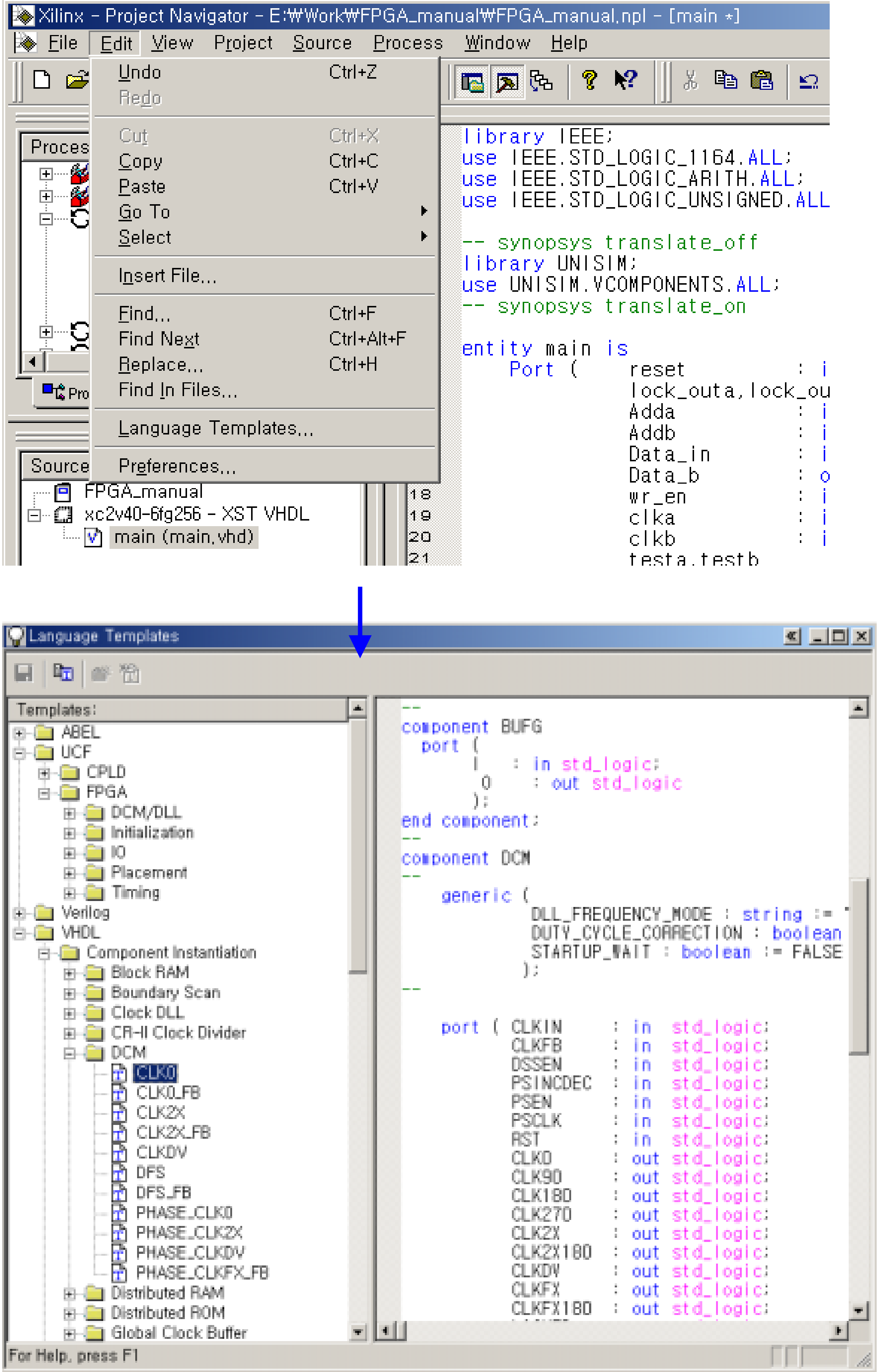
오른쪽 부분의 HDL source 창에 있는 main.vhd 파일의 코드를 완성하여야 합니다. 파일의 Architecture 부분에 User application 로직을 설계합니다. .

이 예제에서 사용될 ISE5.2i 의 기능은 “Language Templates”를 이용한 DCM function 구현과 Core Generator를 이용한 Dual Port Block Memory core를 작성할 것입니다.

3.3 Language Template 이용하기

Project Navigator는 VHDL (혹은 Verilog HDL) 사용자를 위한 기본적인 Language Templates를 제공합니다.

기본적인 HLD Standard 문법을 Language Templates를 이용하여 쉽게 설계를 할 수 있습니다.



<그림 10> DCM Instantiation from Language Templates

위와 같이 DCM - CLK0의 내용 중 필요한 부분을 복사하여 main.vhd에 붙여 넣기를 합니다. DCM 과 BUFG 모듈을 이용하여 clka 과 clkb에 대한 설계를 아래와 같이 완성합니다.

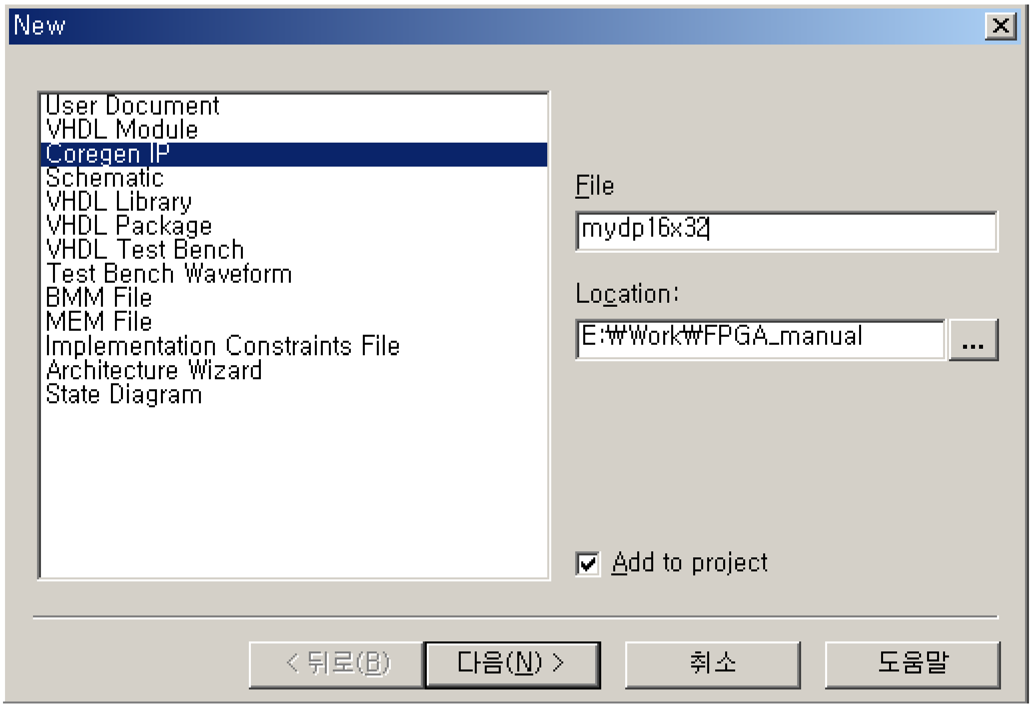
```
architecture Behavioral of main is
component BUFG   port ( I : in std_logic; O : out std_logic );
end component;

component DCM
     generic (
                DLL_FREQUENCY_MODE : string := "LOW";
                DUTY_CYCLE_CORRECTION : boolean := TRUE;
                STARTUP_WAIT : boolean := FALSE
               );
     port ( CLKIN       : in   std_logic;
              CLKFB       : in   std_logic;
              DSSEN       : in   std_logic;
              PSINCDEC   : in   std_logic;
              PSEN        : in   std_logic;
              PSCLK       : in   std_logic;
              RST         : in   std_logic;
              CLK0        : out std_logic;
              CLK90       : out std_logic;
              CLK180      : out std_logic;
              CLK270      : out std_logic;
              CLK2X       : out std_logic;
              CLK2X180   : out std_logic;
              CLKDV       : out std_logic;
              CLKFX       : out std_logic;
              CLKFX180   : out std_logic;
              LOCKED      : out std_logic;
              PSDONE      : out std_logic;
              STATUS      : out std_logic_vector(7 downto 0)
             );
end component;

signal GND : std_logic;
signal CLKa_W, clkb_w: std_logic;
signal CLKa_1X, clkb_1x: std_logic;

begin
```

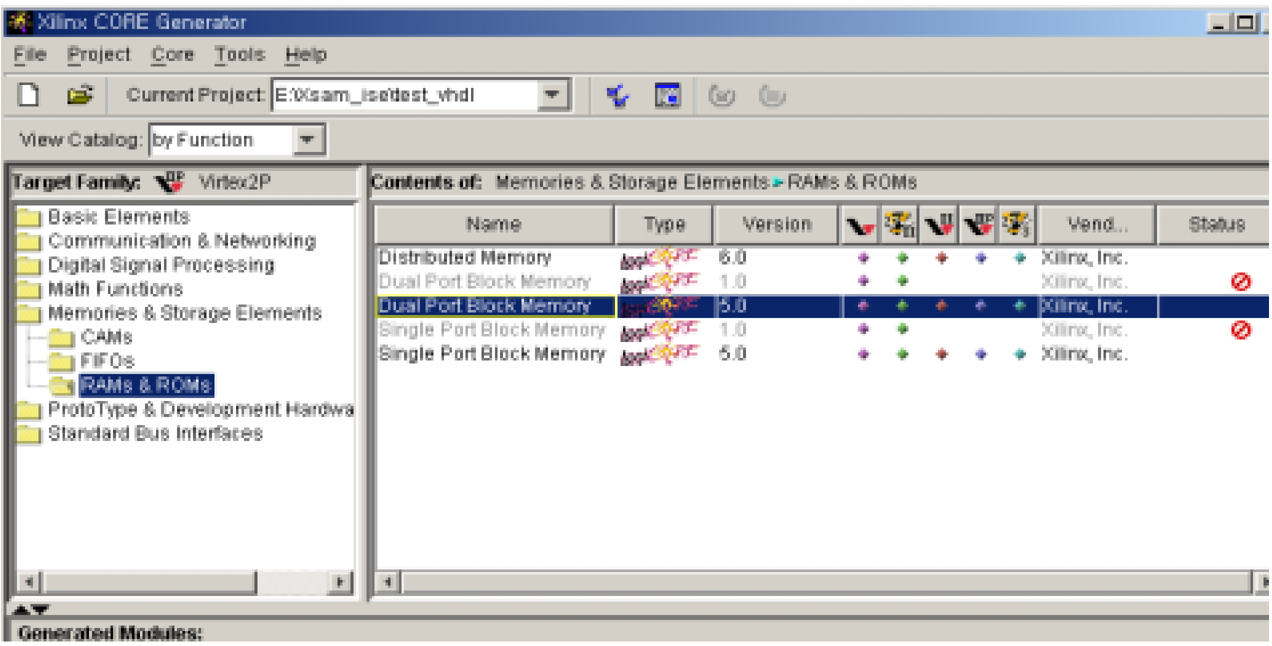
```
GND <= '0';

U_DCMa: DCM              -- DCM Instantiation   for clka
   port map (
                CLKIN =>      CLKa,
                CLKFB =>      CLKa_1x,
                DSSEN =>      GND,
                PSINCDEC => GND,
                PSEN =>       GND,
                PSCLK =>      GND,
                RST =>        reset,
                CLK0 =>       CLKa_W,
                LOCKED =>     LOCK_outa
               );

U_BUFGa: BUFG
   port map (      I => CLKa_W,
                   O => CLKa_1X
                   );
U_DCMb: DCM     -- DCM Instantiation for clkb
   port map (
                CLKIN =>      CLKb,
                CLKFB =>      CLKb_1x,
                DSSEN =>      GND,
                PSINCDEC => GND,
                PSEN =>       GND,
                PSCLK =>      GND,
                RST =>        reset,
                CLK0 =>       CLKb_W,
                LOCKED =>     LOCK_outb
               );

U_BUFGb: BUFG
   port map (      I => CLKb_W,
                   O => CLKb_1X
                   );
end Behavioral;
```

3.4 Core Generator 이용하기

Core Generator를 띄우기 위해 Project Navigator -> Project -> New Source에서 아래와 같이 Core Generator를 이용하여 DPRAM core를 작성합니다.

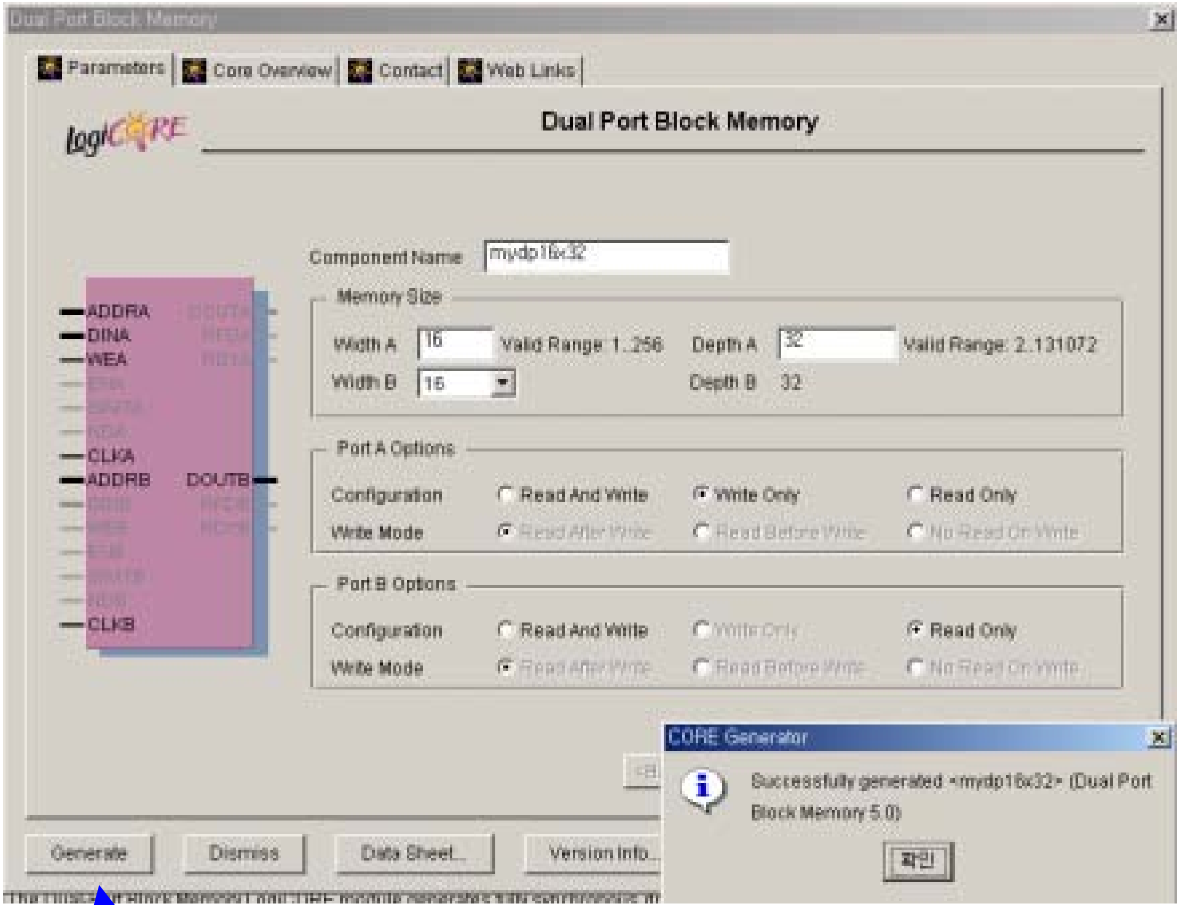
이 메모리 모듈의 기능은 서로 다른 두개의 clock이 입력되고, 한쪽은 입력 전용, 다른 쪽은 출력 전용으로 사용되는 모듈을 작성합니다.





<그림 11> Core Generator

Dual Port Block Memory를 Double click 합니다.

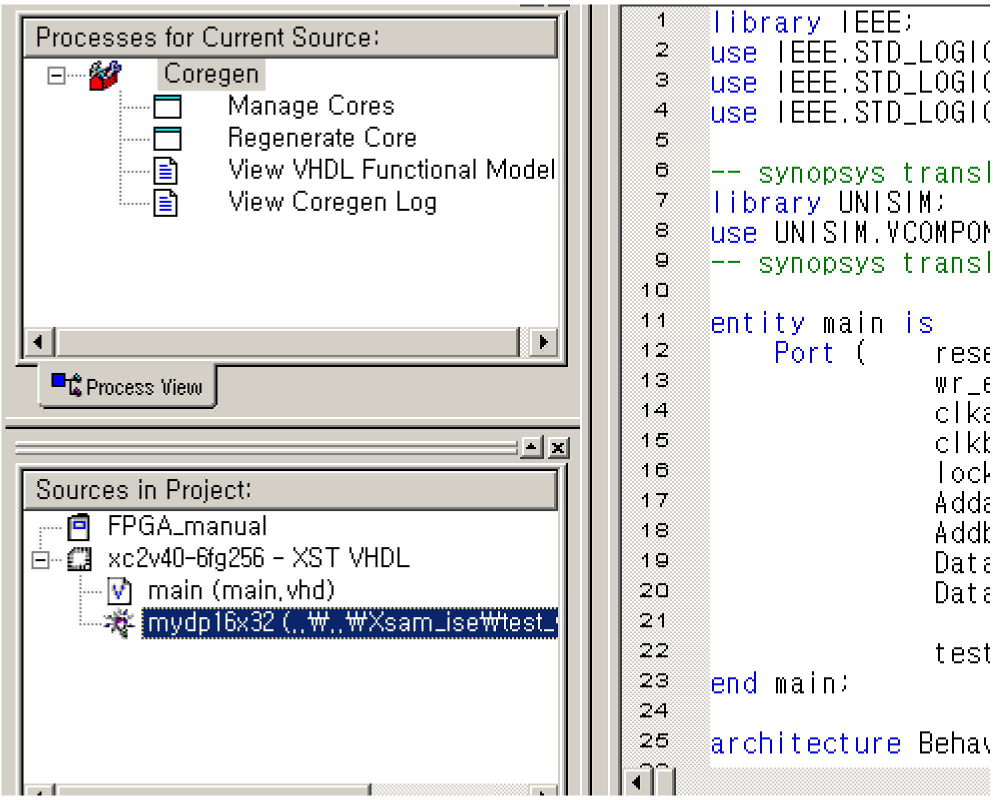


<그림 12>

위의 내용은 mydp16x32라는 이름의 DPRAM을 만들기 위한 Parameters를 넣습니다.
다음, Generate Icon을 선택하면, 16bit x 32depth의 1 input 1 output 포트를 가진 DPRAM core 가 만들어집니다.

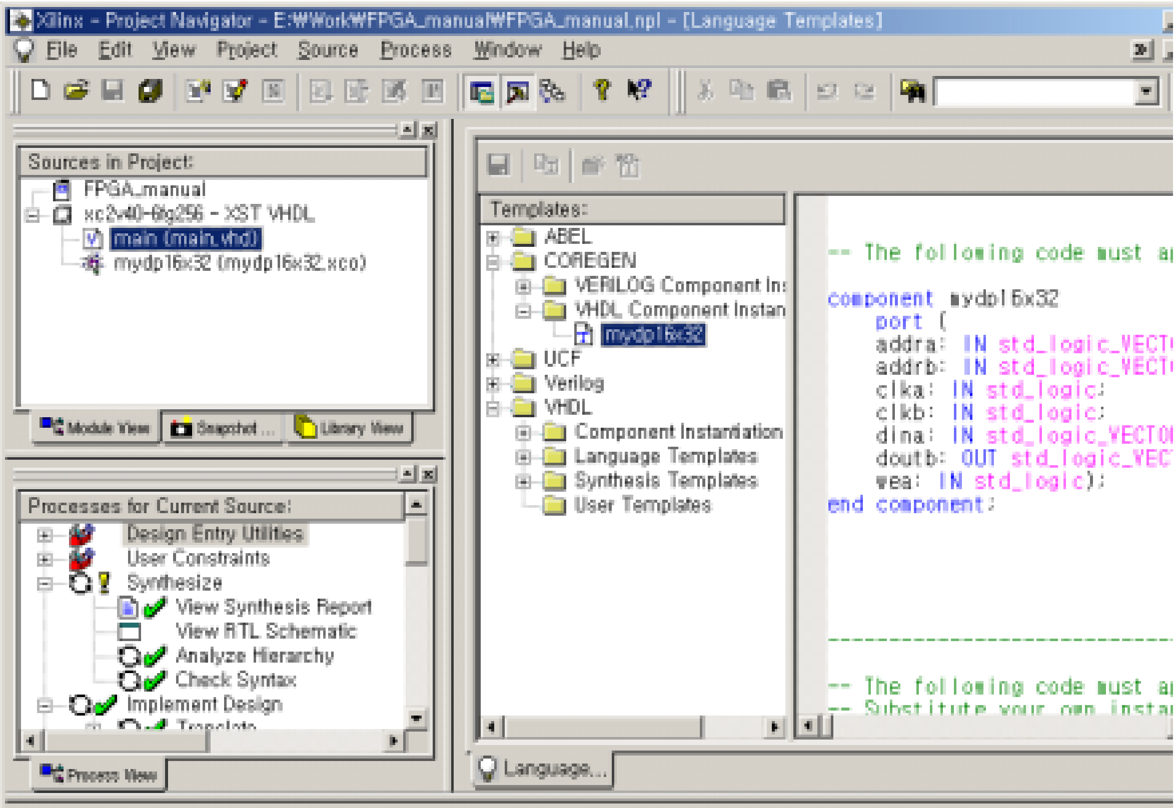
“확인” 을 선택하여 Core Generator를 이용한 DPRAM core의 생성작업은 끝냅니다.
이제 Coregenerator의 창을 닫습니다.

다시 Project Navigator를 보면, 지금 Core Generator에서 만들었던 DPRAM IP core가 아래 <그림 13>과 같이 “Source In project” 창에 나타나게 됩니다.



<그림 13>

Language Templates에 지금 만든 User Ipcore가 나타나지 않는다면 ISE Navigator를 다시 실행시킵니다.



만들어진 “mydp16x32” Core IP를 main.vhd 상에서 Port mapping하는 작업을 합니다.

위에서와 같이 “Language Templates” 의 Coregen을 보면 지금 까지 만들어진 mydp16x32 Core 의 Instantiation 을 위한 Information 을 볼 수 있습니다.

TOP file인 Main.vhd에 지금 작성한 mydp16x32를 Port Mapping 하기 위하여 Language Templates 의 component mydp16x32 부분과 port map 부분을 Main.vhd 파일에 복사한 후, 아래의 Source와 같이 Entity에서 선언된 Port와 “mydp16x32”와 Port mapping 시킵니다.

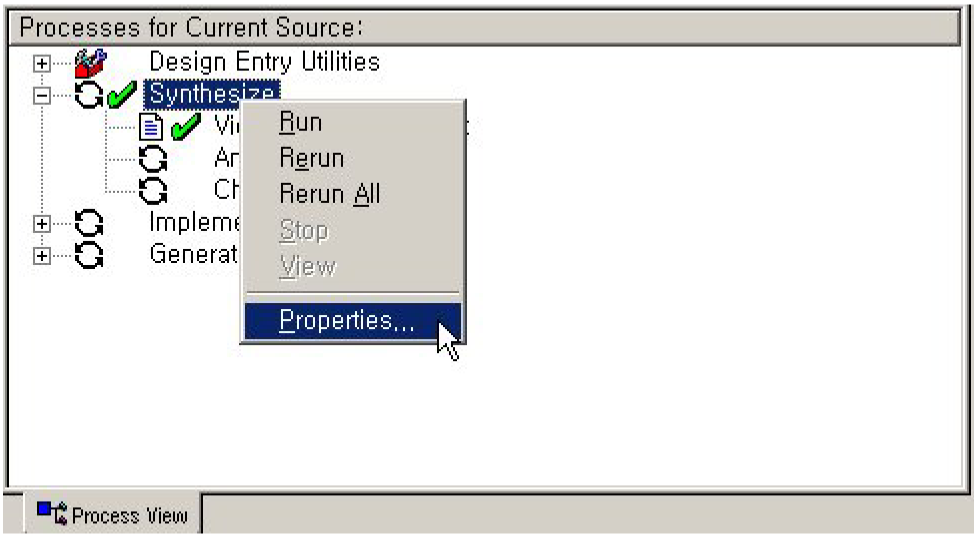
```
component mydp16x32
    port (
    addra       : IN std_logic_VECTOR(4 downto 0);
    addrb       : IN std_logic_VECTOR(4 downto 0);
    clka: IN std_logic;
    clkb: IN std_logic;
    dina: IN std_logic_VECTOR(15 downto 0);
    doutb       : OUT std_logic_VECTOR(15 downto 0);
    wea: IN std_logic);
end component;

begin

mem_blk : mydp16x32
port map (
            addra       => adda,
            addrb       => addb,
            clka        => CLKa_1X,
            clkb        => CLKb_1X,
            dina        => data_in,
            doutb       => data_out,
            wea         => wr_en);
```

이것으로서 기본적인 Source 작성이 끝났습니다. 다음 장에서는 이 source를 가지고 Synthesis와 Implementation을 수행하게 됩니다.

4. Synthesize 하기

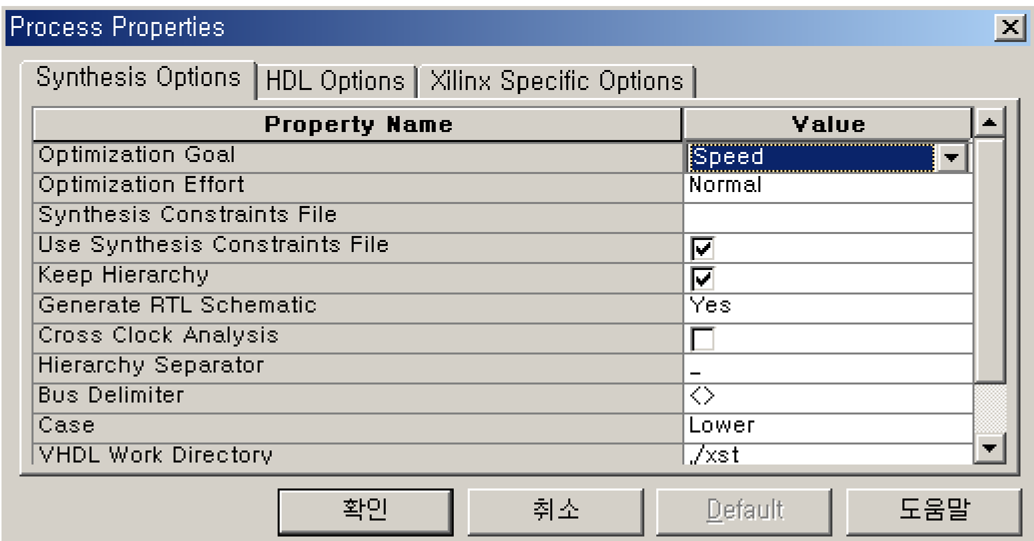
ISE 5.2i ISE에서 사용되는 Synthesis Tool은 XST(Xilinx Synthesis Technology) 입니다. XST 는 HDL design을 synthesize하고 .NGC 확장자를 가진 Netlist 파일을 생성합니다.

아래의 그림과 같이 Synthesize를 하기 위하여 Processes for Current Source 창의 Synthesize를 더블 클릭합니다. Synthesize Default Option을 변경하기 위한 방법은 다음과 같습니다. Synthesize에 마우스의 오른쪽 버튼을 클릭하고 Properties..를 선택하면....



< 그림 15 > Processes for Current Source창에서 Properties 선택

Process Properties가 뜹니다.



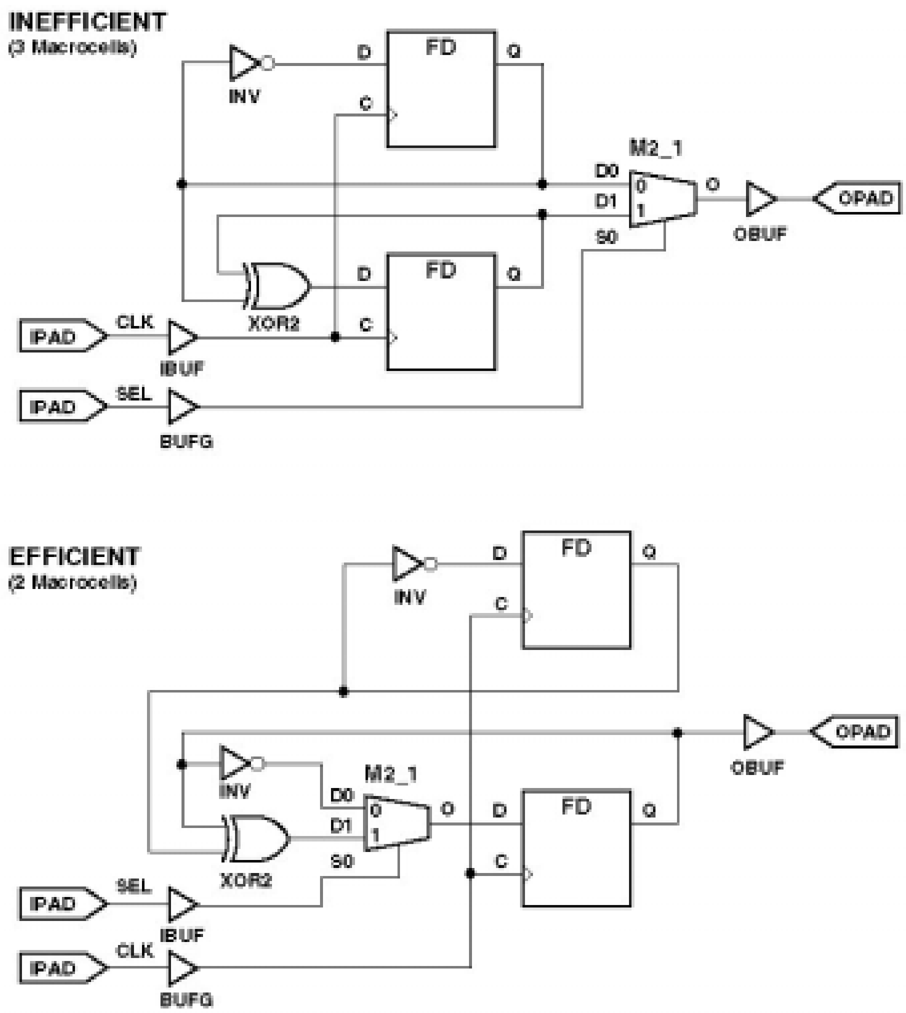
< 그림 16 > Synthesis Option Properties

Synthesis Options의 내용은...

- Optimization Goal 은 두 가지 value (Speed, Area)가 있는데 로직레벨을 축소하면서

speed위주로 최적화하는 방법으로서 예를 들어 CPLD의 경우 3개의 macrocell사용하는 것을 2개만 사용하여서 signal이 거치는 단계를 줄이는 방법이 있고 또 다른 방법으로 Area위주로 최적화하는 경우도 있습니다. .

Figure 3.8 Reducing Levels of Logic

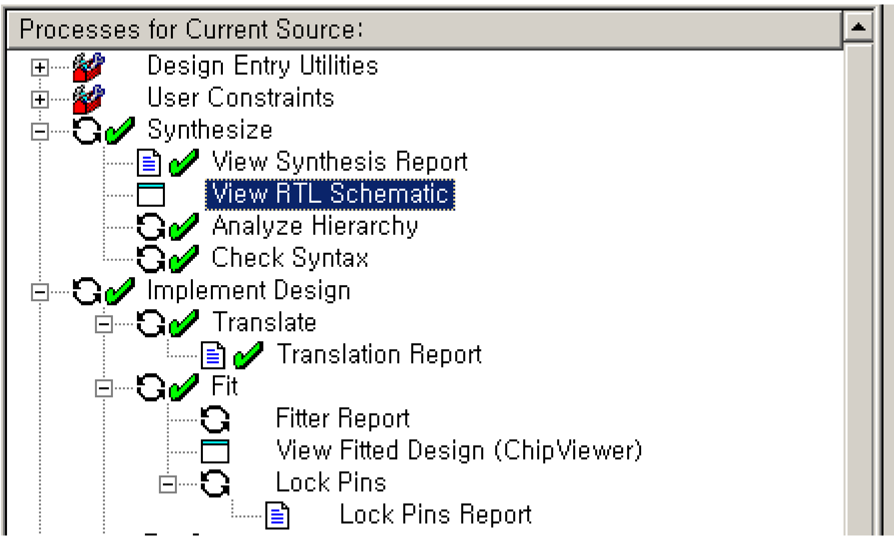


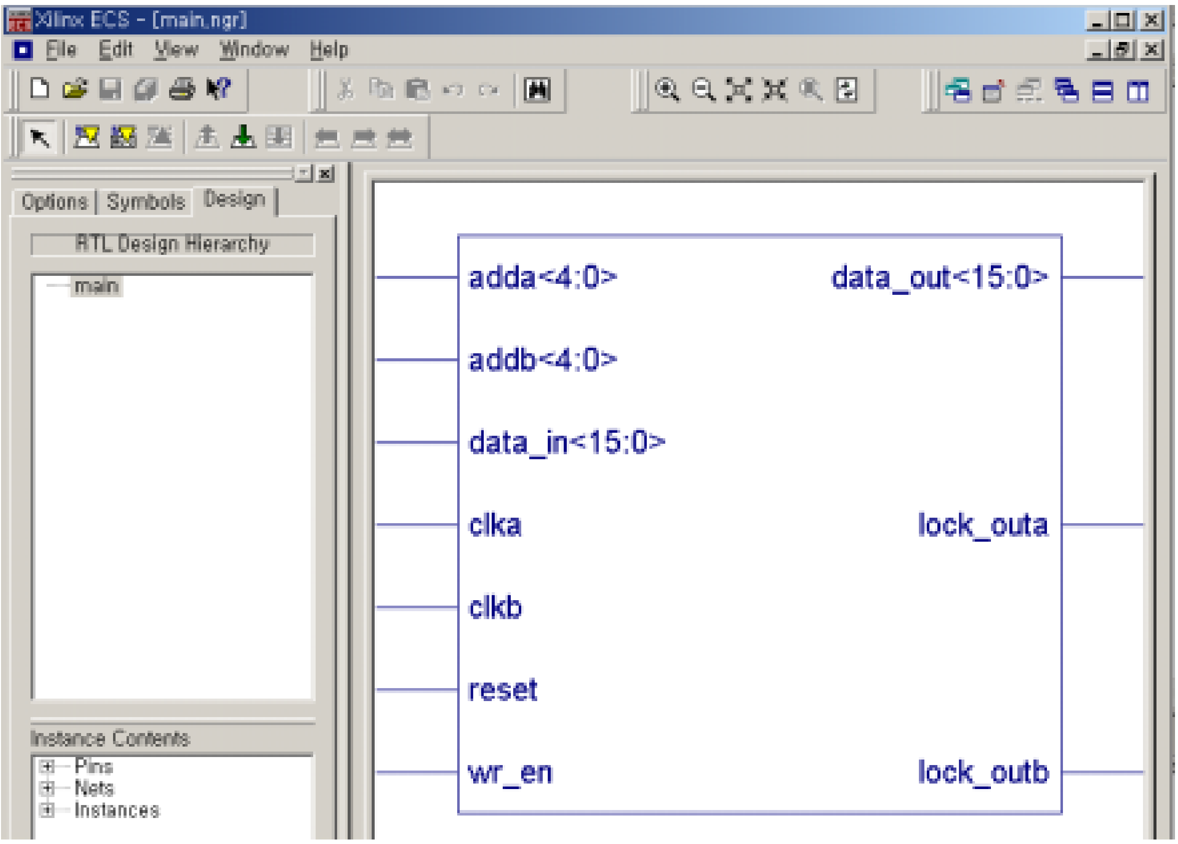
- Optimization Effort 는 synthesis optimization의 수준을 나타내며 default 로 Normal 이지만 high로 놓게 되면 multiple optimization 알고리즘을 이용하여 최적하지만 많은 Compile 시간이 걸립니다.

그 밖의 Properties 들은 Process Properties 의 도움말을 참조하여 대개의 경우는 Default 값으로 사용하는 것이 바람직합니다.

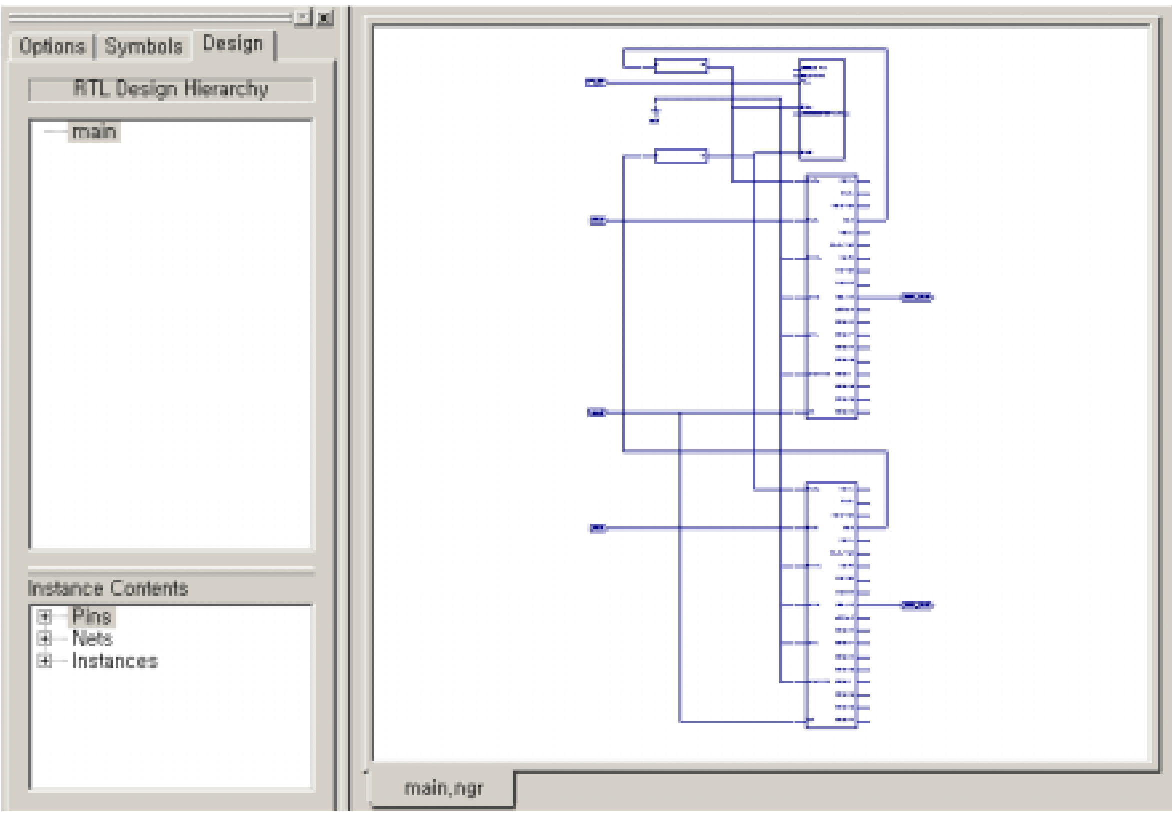
4.1 View RTL Schematic

HDL로 작성된 디자인을 쉽게 이해하기 위해 “View RTL Schematic”을 이용합니다.
전체 Top 디자인의 Block-diagram과 내부의 세부적인 Sub-diagram을 더블클릭 함으로서 자세히 볼 수 있습니다.





< 그림17 > View RTL Schematic

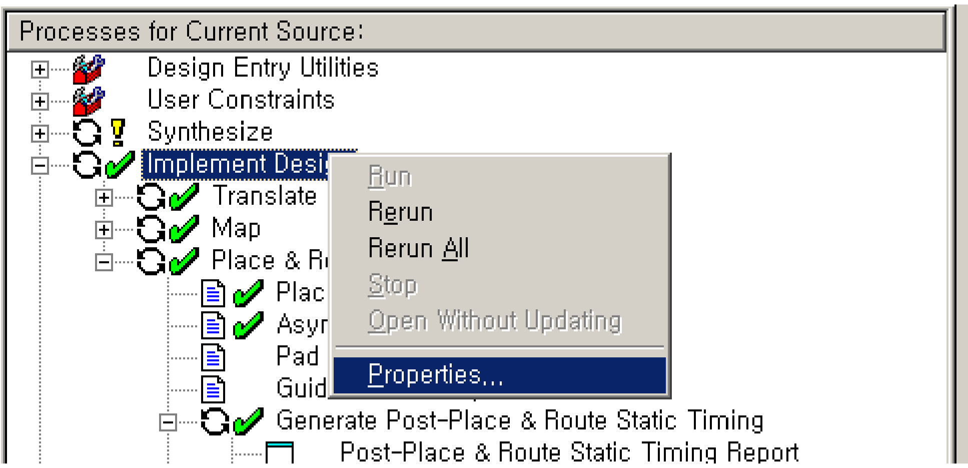


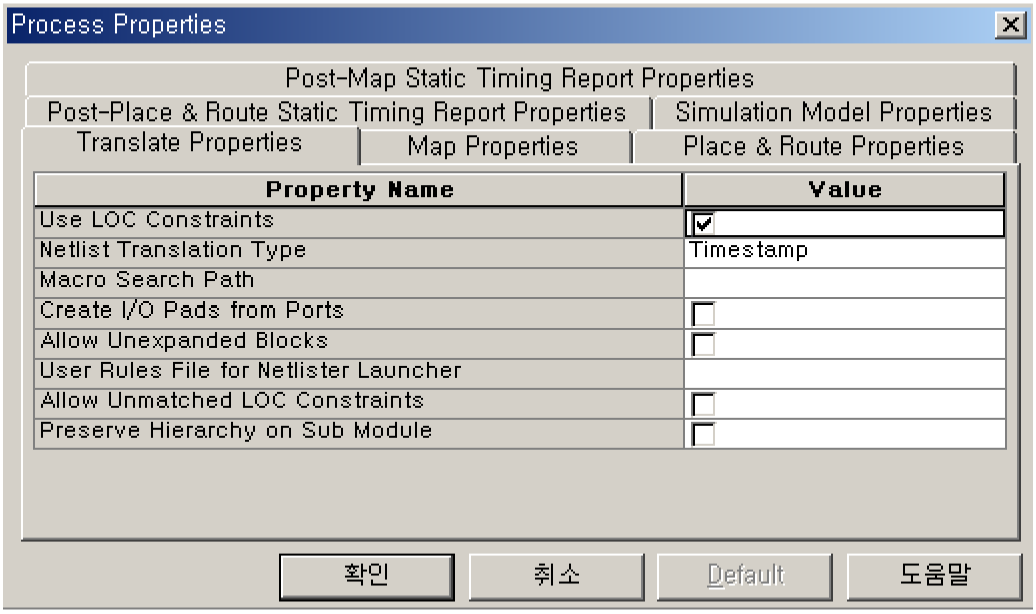
< 그림18> RTL schematic의 Sub diagram

각 블록을 클릭함으로서 Hierarchy subdiagram을 볼 수 있습니다.

5. Implementation 하기

CPLD 과 FPGA은 약간 다른 Implementation을 수행합니다. 이 장에서는 FPGA의 경우만 설명합니다. FPGA의 Implementation은 Translate와 Map, Place & Route 로 나눌 수 있습니다. Implementation의 Option을 변경하는 작업은 Synthesis에서와 같이 Implementation 아이콘을 오른쪽 마우스로 선택하여 바꿀 수 있습니다.





<그림20> Implementation Option

5.1 Translate

XST로 synthesis 해서 생성된 Netlist 인 .NGC파일을 가지고 아래와 같은 작업을 수행하여 .NGD 파일을 생성합니다.

Netlist Import

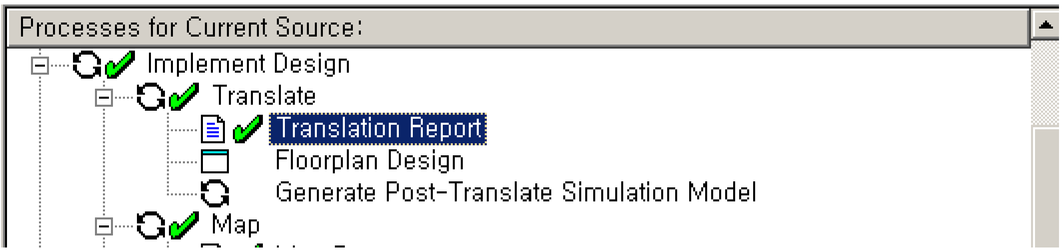
Optimisation

Translate to physical elements

Design Rule Check

Device Info (device, package, speed grade)

Utilisation Report



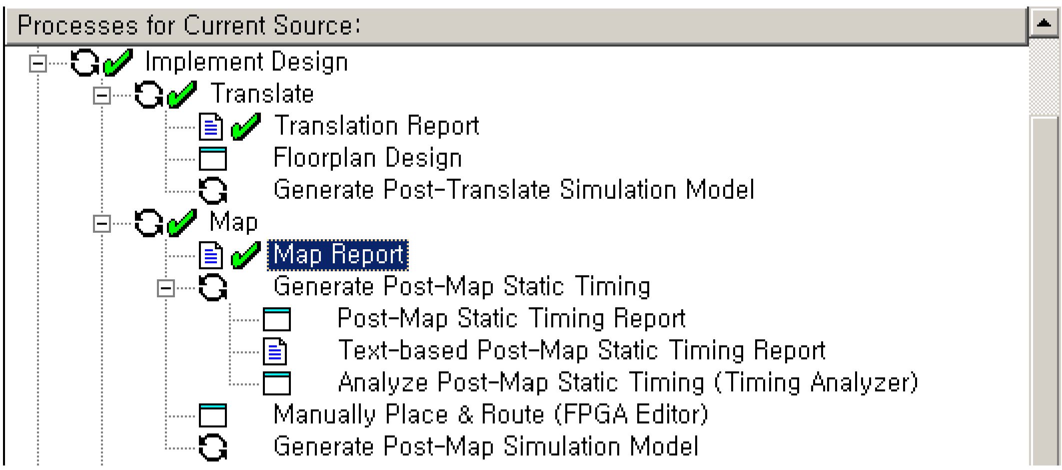
< 그림 20 > Implement Design 선택

이 과정을 수행하면서 발생한 Error 및 Warning 등의 Report를 해주는 파일이 NGDbuild Log파일인 .bld파일 안에 들어있고 이 .ngd파일은 Translation Report를 더블클릭하면 아래와 같이 볼 수 있습니다.

<그림 21> Translation Report 선택

5.2 Mapping

설계된 Logic Element들을 CLB나 IOB 같은 Physical Element들로 할당하는 과정입니다. 이러한 작업의 결과도 아래와 같이 Map Report을 선택하여 결과를 확인할 수 있습니다.



<그림 22> Map Report 선택

Map Report에는 사용된 내부 Resource 정보를 포함해서 사용된 CLB의 수 , 사용된 I/O 개수 ,각 Error message 등의 내용들이 들어있습니다. 이 Report는 설계자의 의도대로 설계가 제대로 되었는지 확인할 수 있는 중요한 Report입니다.

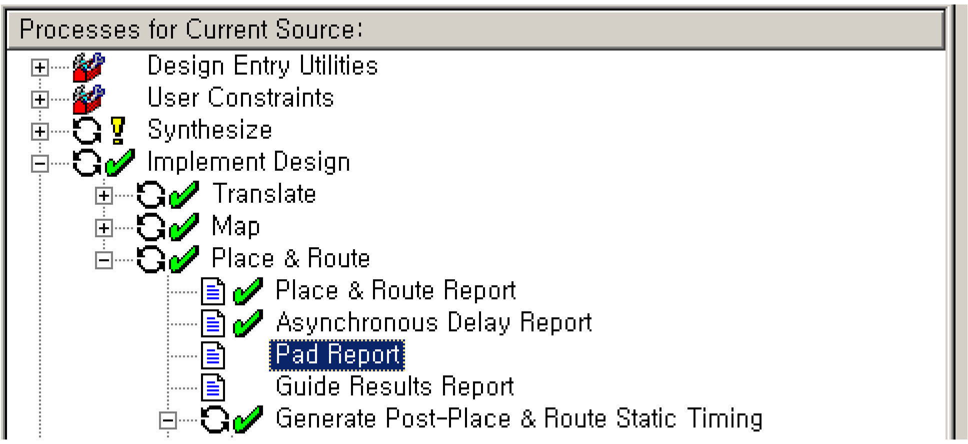
이 결과에는 Routing과 관련한 Delay 정보는 포함되어 있지 않습니다.

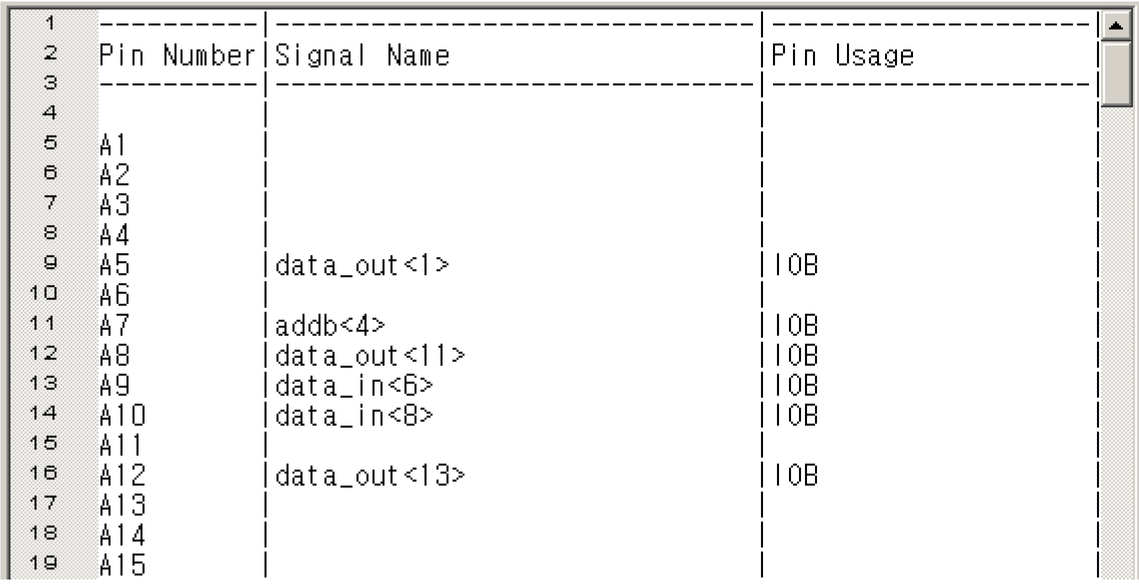
```
Release 5.2.02i - Map F.30a
Xilinx Mapping Report File for Design 'main'

Design Information
------------------
Command Line   : C:/Xilinx52/bin/nt/map.exe -quiet -p xc2v40-fg256-6 -cm area
-pr b -k 4 -c 100 -tx off -o main_map.ncd main.ngd main.pcf
Target Device  : x2v40
Target Package : fg256
Target Speed   : -6
Mapper Version : virtex2 -- $Revision: 1.4 $
Mapped Date    : Mon Jun 09 17:14:32 2003

Design Summary
--------------
Number of errors:      0
Number of warnings:    0
Logic Utilization:
Logic Distribution:
    Number of Slices containing only related logic:       0 out of       0    0%
    Number of Slices containing unrelated logic:          0 out of       0    0%
        *See NOTES below for an explanation of the effects of unrelated logic

   Number of bonded IOBs:                48 out of     88    54%
   Number of Block RAMs:                  1 out of      4    25%
   Number of GCLKs:                       2 out of     16    12%
   Number of DCMs:                        2 out of      4    50%

Total equivalent gate count for design:  79,542
Additional JTAG gate count for IOBs:  2,304
Peak Memory Usage:  53 MB

NOTES:
```

< 그림 23> Map Report의 내용

5.3 Place & Route

Place는 CLB, IOB 등 Logic Block을 Xilinx Device의 특정 위치에 할당하는 과정이고, Route는 Xilinx Device에 위치한 Logic Block들을 Interconnection Element들로 서로 연결하는 과정입니다.



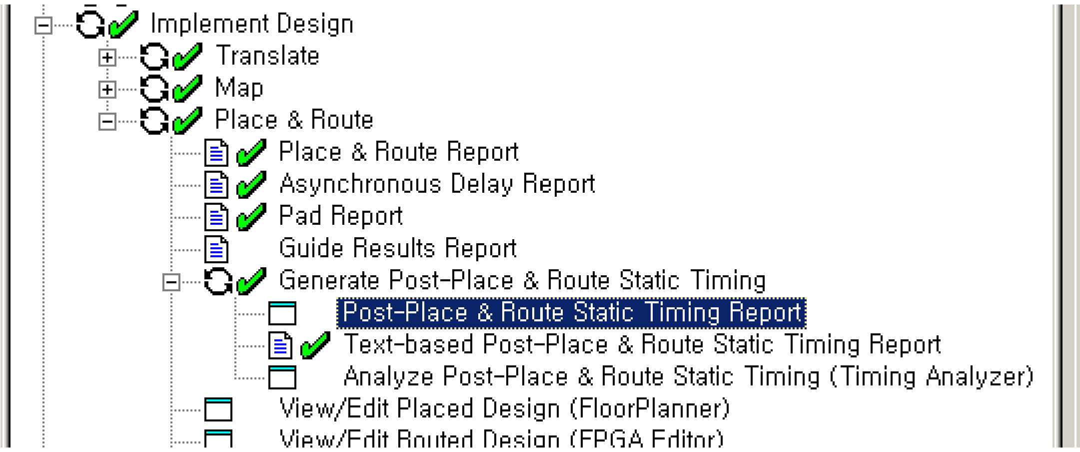


Pin Number	Signal Name	Pin Usage
A1		
A2		
A3		
A4		
A5	data_out<1>	IOB
A6		
A7	addb<4>	IOB
A8	data_out<11>	IOB
A9	data_in<6>	IOB
A10	data_in<8>	IOB
A11		
A12	data_out<13>	IOB
A13		
A14		
A15		

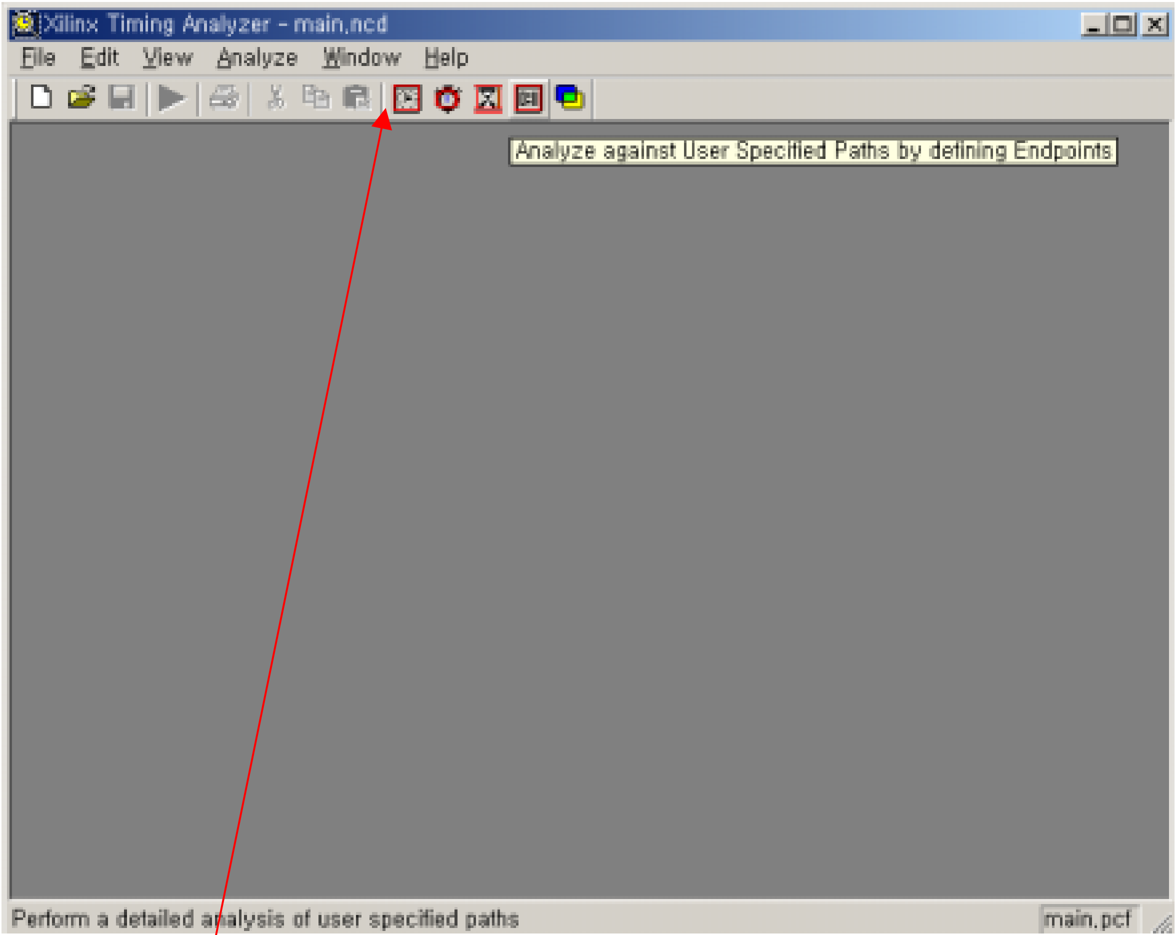
<그림24> Pad Report

설계자가 UCF 파일에서 설정한 Pin Assignment 정보가 제대로 적용되어 Implementation 되었는지를 확인할 수 있는 Report file 입니다.

아래와 같은 Generate Post-Place & Route Static Timing의 기능은 디자인상의 Path에 대한 Timing information을 제공합니다. 이 정보는 가장 긴 Path delay 부터 가장 짧은 Path delay 를 표시해 줍니다.

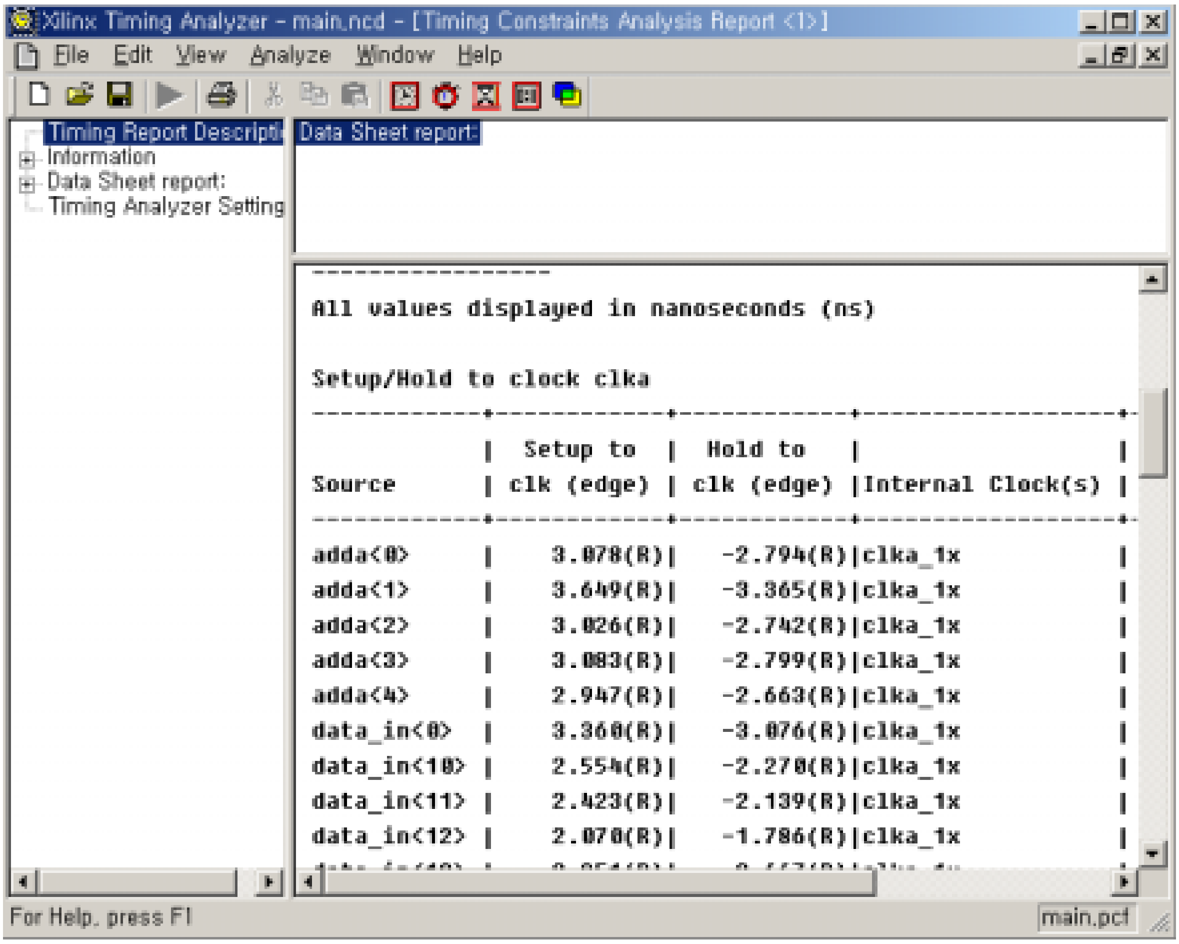


Timing Analyzer를 통해 좀 더 면밀한 Timing정보로 디자인을 분석할 수 있습니다.



<그림 25 > Timing Analyzer 창

"Analyze against Timing Constraints"아이콘을 클릭함으로 아래와 같이 Timing Constraint에 대해 실제로 fitting신호들의 시간정보를 볼 수 있습니다.



< 그림 26 > Timing Analyzer 창에서 Timing Constraints Analysis Report

이렇게 Implement를 마치고 정상적이면 녹색의 체크표시가 되고 설계자가 원하는 Timing Option에 만족하지 못하여 Error가 발생하면 빨간색으로 표시됩니다.

6. User Constraints (Pin Assignment)

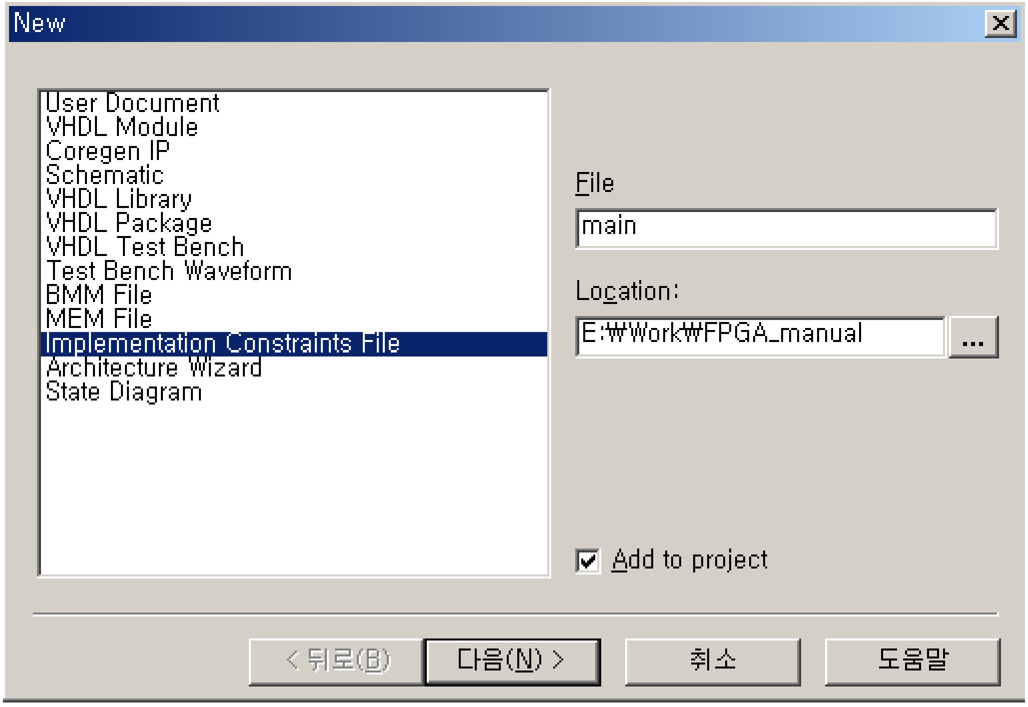
이 장에서는 지금까지 Auto Assignment 된 project의 여러 In/Out pin들을 설계자가 의도하는 Device의 Pin에 직접 설정하는 방법을 설명합니다.

Pin Assignment를 하는 방법은 여러 가지가 있습니다. “Edit Constraints(Text), Assign Package Pins, Create Constraints Editor를 이용하는 방법이 있습니다.

UCF 파일을 수정한 후 저장하게 되면, Implementation을 다시 실행해야 합니다.

6.1 UCF 파일 만들기

먼저 Project Navigator -> Project -> New Source... 를 선택하고 아래와 같이 작성한 하면,

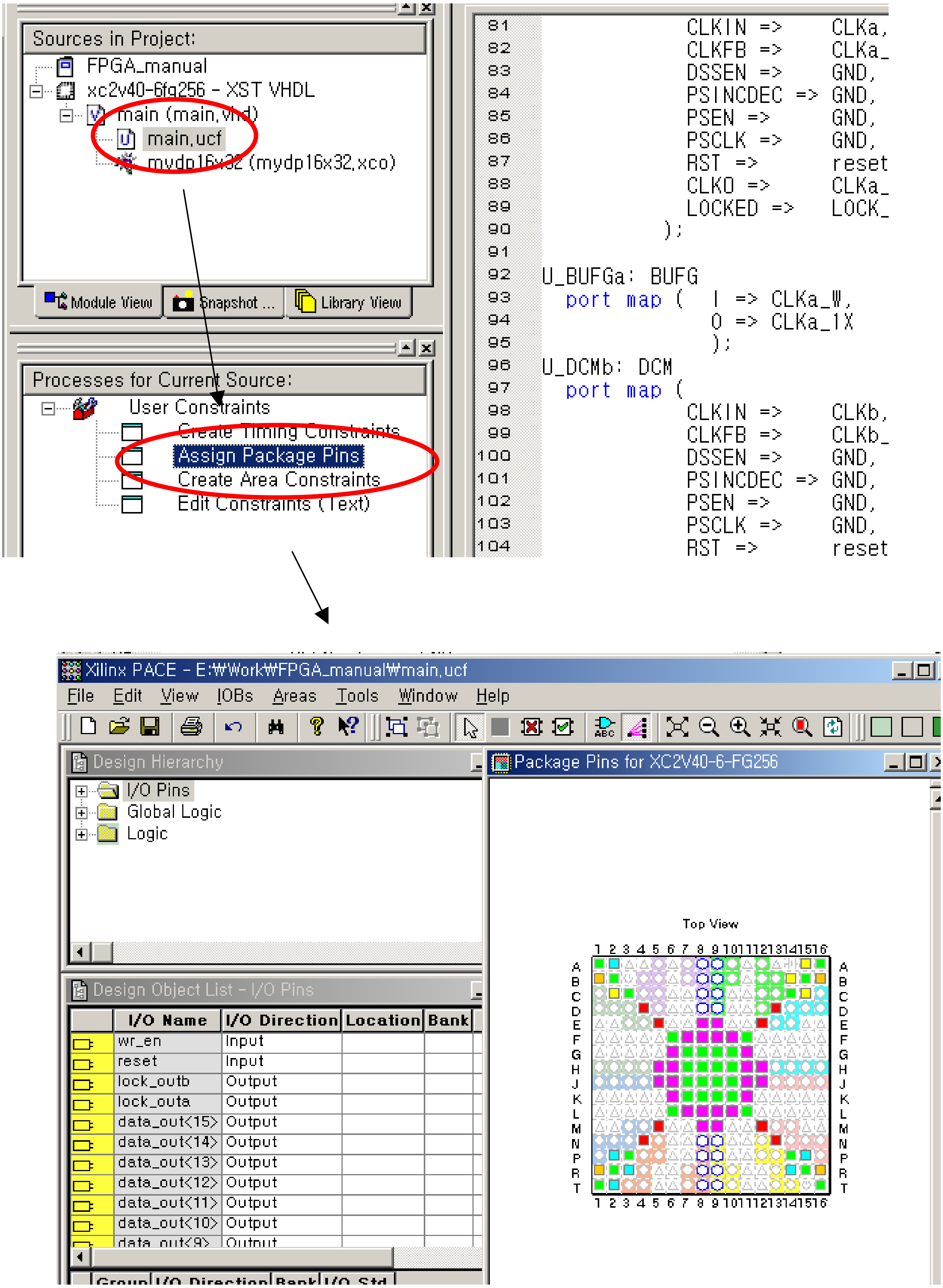


< 그림 27 > Implementation Constraints File 선택

아래와 같이 main.ucf 파일이 Source in Project 창에 만들어 집니다. 아직 이 UCF 파일 안에는 아무런 정보가 없는 빈 파일입니다. 이 파일에 Pin Assignment 정보를 포함한 각종 Timing Option을 작성합니다. 먼저 Xilinx 의 PACE를 이용하여 Pin Assignment를 하는 방법만 설명하겠습니다.

6.2 Assign Package Pins

Xilinx PACE를 실행하기 위해서는 아래와 같이 선택합니다.

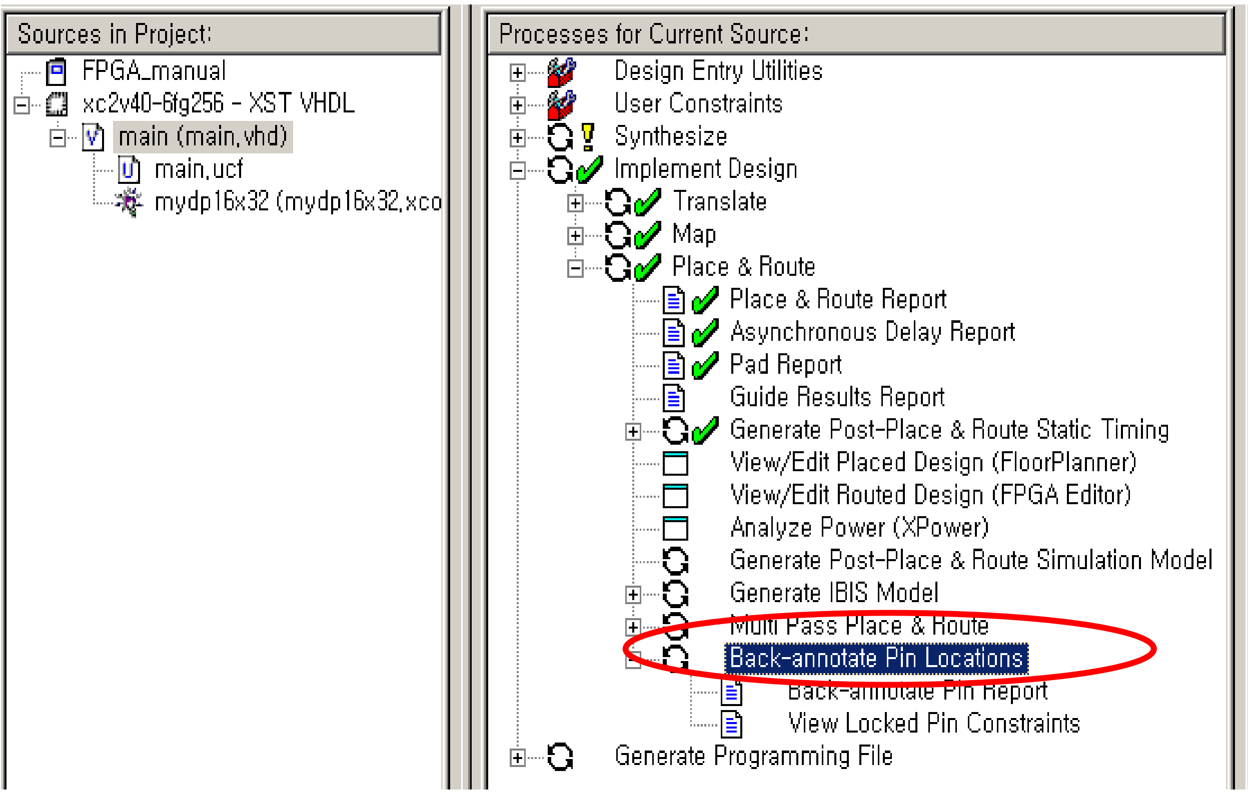


Pin Assignment를 하는 방법 중의 하나로서 GUI상에서 해당 Pin을 Drag and Drop 함으로서 Pin Assignment 하는 방법입니다.

<그림28> PACE 화면

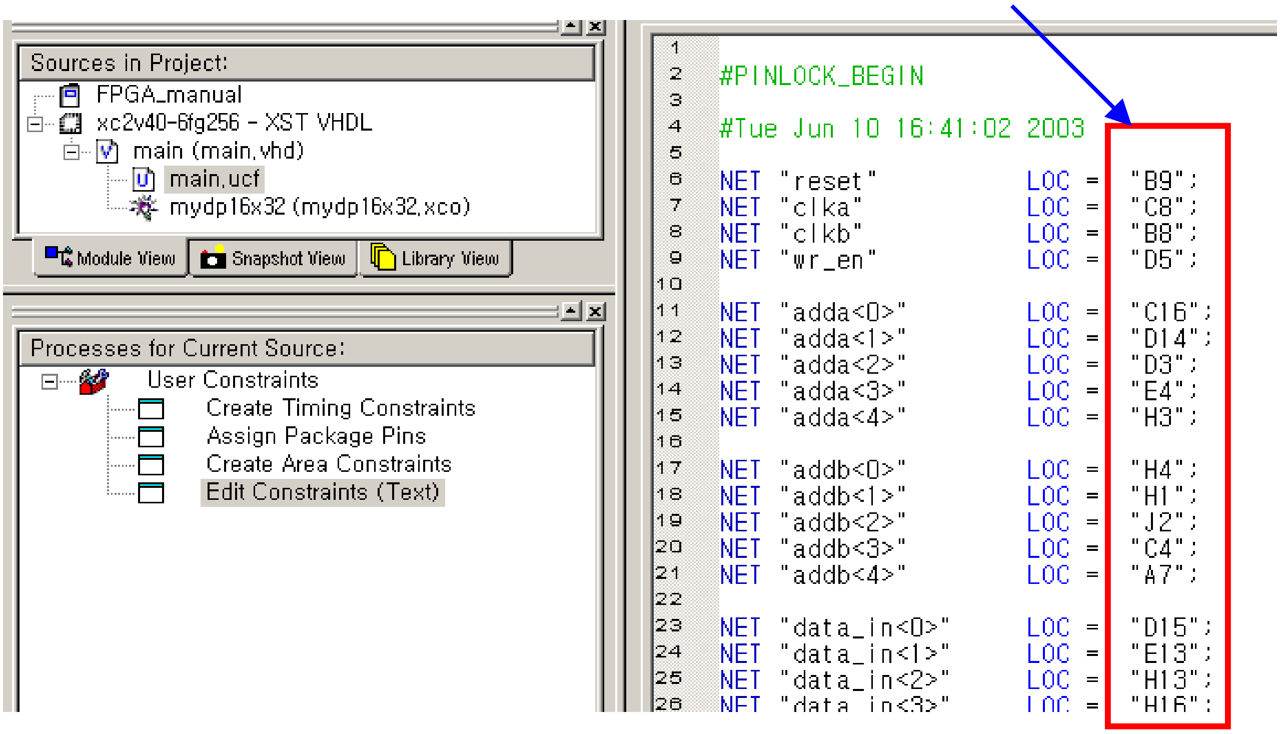
6.3 Back-annotate Pins Location

이 방법은 Implementation P & R 시 자동으로 만들어지는 Pin Assignment 정보를 이용하는 것입니다. 아래의 그림과 같이 Back-annotate Pin Locations 을 실행시키면,



<그림 29> Back-annotate Pin Locations

설계자는 이 곳의 Pin 정보만 수정하면 됩니다.



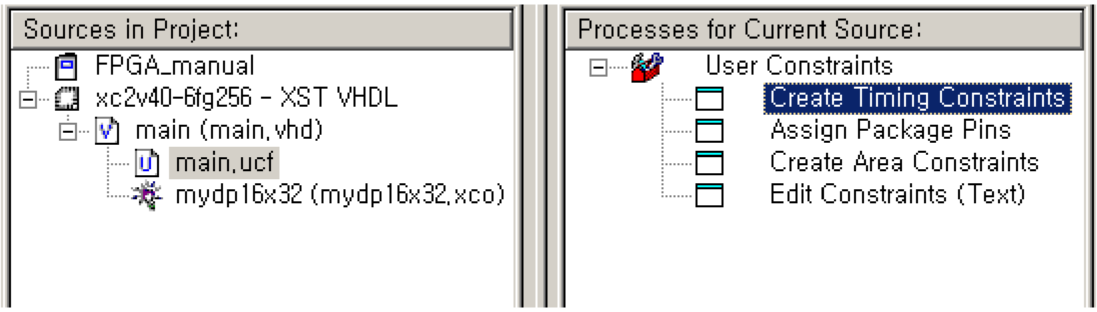
이와 같이 main.ucf 파일에 Pad Report 에 자동으로 만들어진 Pin Assignment 정보가 저장됩니다. 정해진 Pin 번호를 수정한 후 저장합니다.

6.4 Edit Constraints (Text)

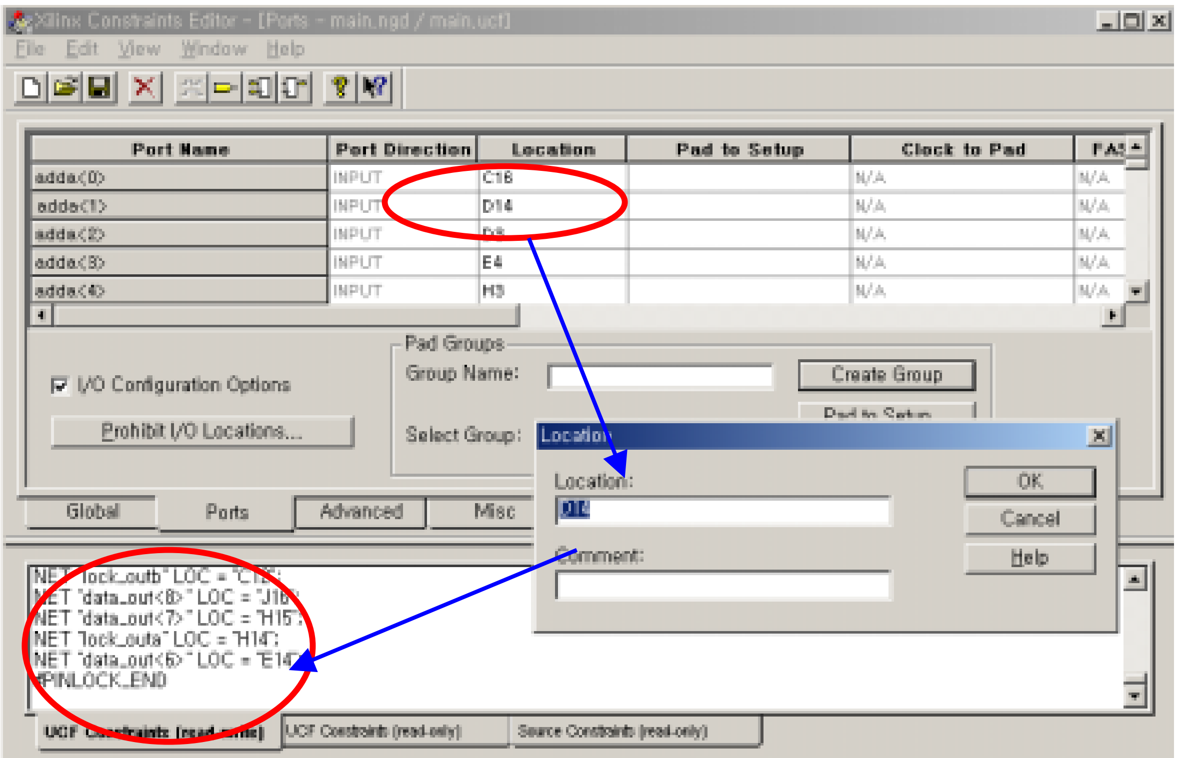
Edit Constraints를 실행하면 main.ucf 파일의 Text editor가 실행됩니다. UCF 파일의 문법에 맞게 직접 작성합니다.

6.5 Create Timing Constraints

Constraints Editor 이용하여 Pin Assignment를 하는 방법입니다. 아래와 같이 Create Timing Constraints를 실행합니다.



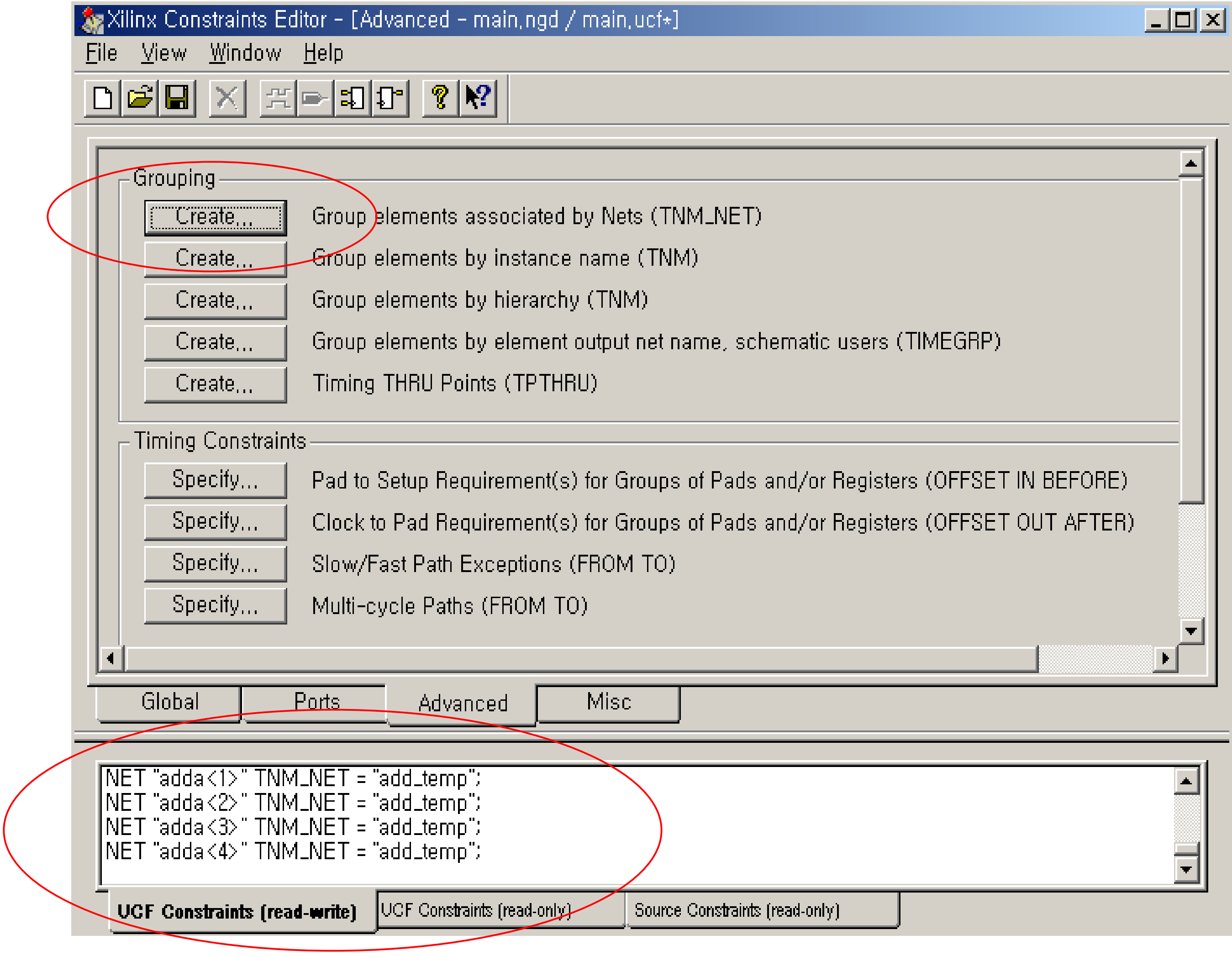
해당 핀의 란을 선택 & 핀 정보 입력



< 그림 30 > Constraints Editor

위의 그림과 같이 직접 핀 정보를 해당 란에 직접 입력하고 저장하면, UCF 파일에 자동으로 저장됩니다.

설계 중에 사용할 수 있는 모든 Timing Option들은 Constraint Editor를 이용해서 사용할 수 있습니다.



작성된 Timing Constraint 정보는 저장과 동시에 자동으로 main.ucf 파일에 Update 됩니다.

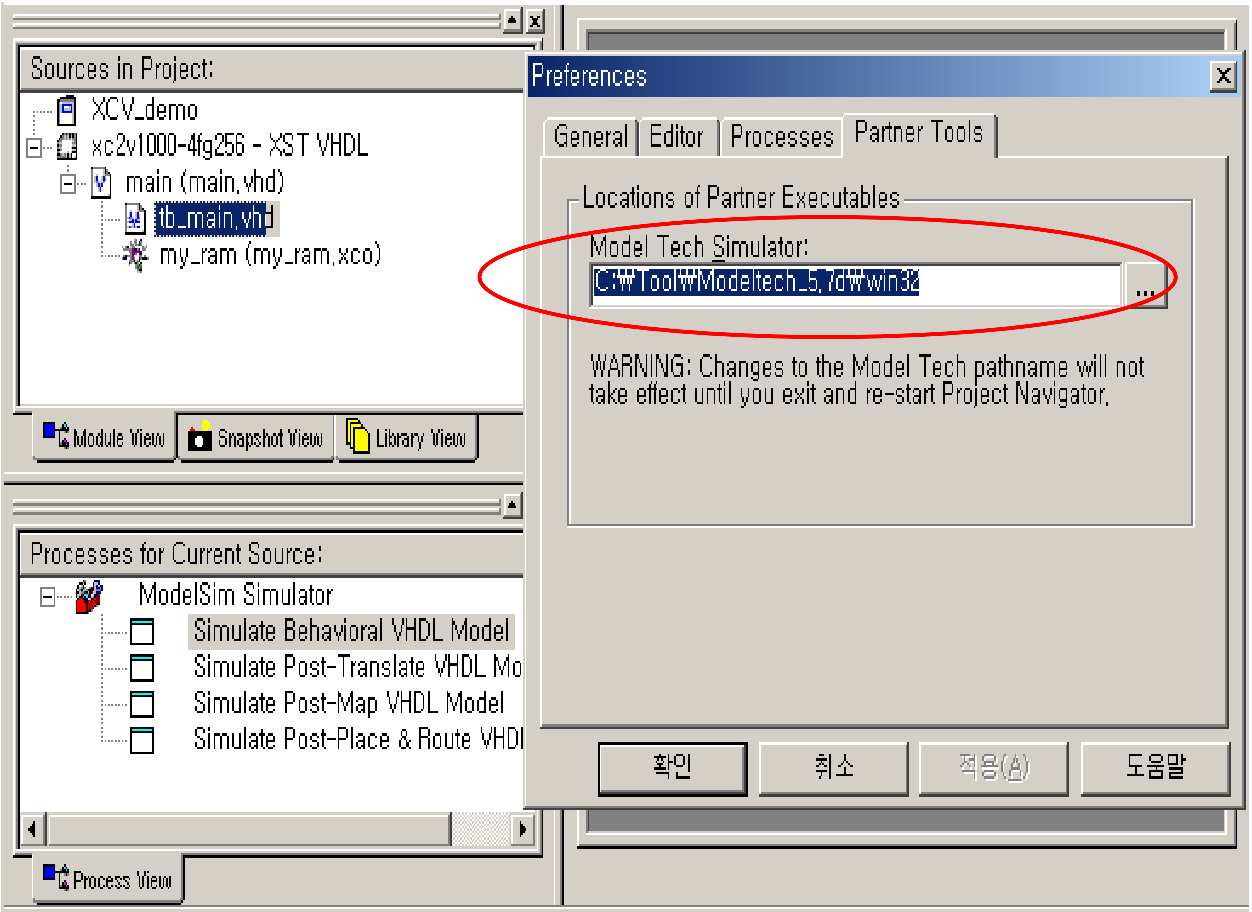
7. Modelsim Simulation

Modelsim simulator는 크게 3가지 종류로 나눌 수 있습니다. Xilinx에서 무료로 제공하는 Modelsim MXE starter version이 있습니다. 이 버전을 사용하기 위해서는 MXE_5.6e_Full_installer.exe 를 설치합니다. Modelsim 사용에 필요한 License는 프로그램 설치 후 자동으로 Xilinx Home page에서 등록하여 License 파일을 받거나, Windows 시작메뉴 -> 프로그램 -> Modelsim XE v5.6a -> Submit License Request를 선택하여 받을 수 있습니다. License 파일은 메일을 통하여 즉시 받을 수 있습니다. 이 버전은 Source의 line 수의 제한이 있습니다. 다른 하나는 Modelsim MXE (Modelsim Xilinx Edition) 이며, Modeltech에서 나오는 Modelsim PE/SE로 나눌 수 있습니다.

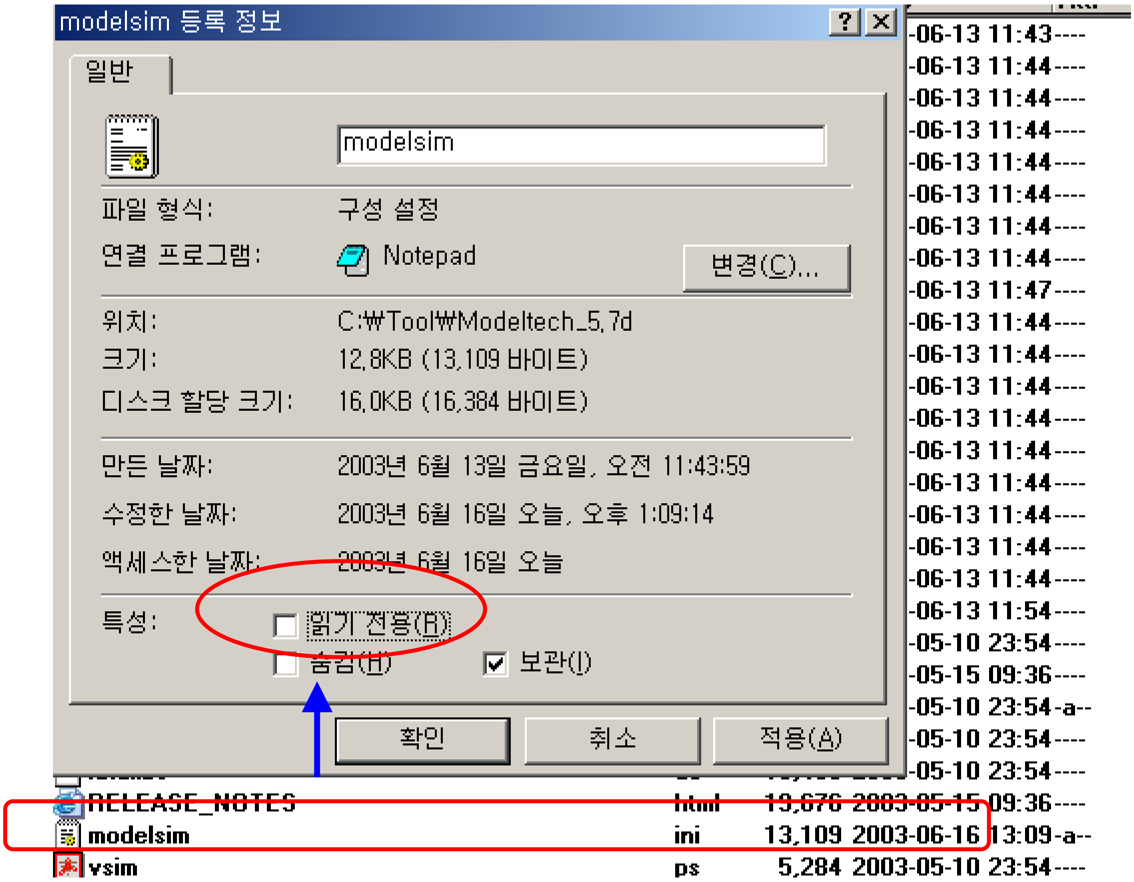
7.1 Modelsim 사용 환경 설정

Project Navigator에서 Modelsim을 연결하기 위해서는 메뉴의 Edit -> Preference를 선택하고 Partner Tools에서 아래와 같이 Modelsim Xilinx Edition version 의 실행 파일의 디렉 토리 경로가 정확히 설정되어있는지를 확인하여야 합니다.

-> 아래의 그림은 Modelsim SE 를 사용합니다.



Xilinx에서 제공하는 Modelsim MXE Version이 아닌 일반 Modelsim SE or PE를 사용할 경우에는 Modelsim에서 Xilinx device를 인식할 수 있도록 Xilinx device Library를 Compile 해주어야 합니다.
우선 Modelsim이 설치되어 있는 Directory에서 Modelsim.ini 파일의 등록정보 “읽기 전용”을 Uncheck 합니다.



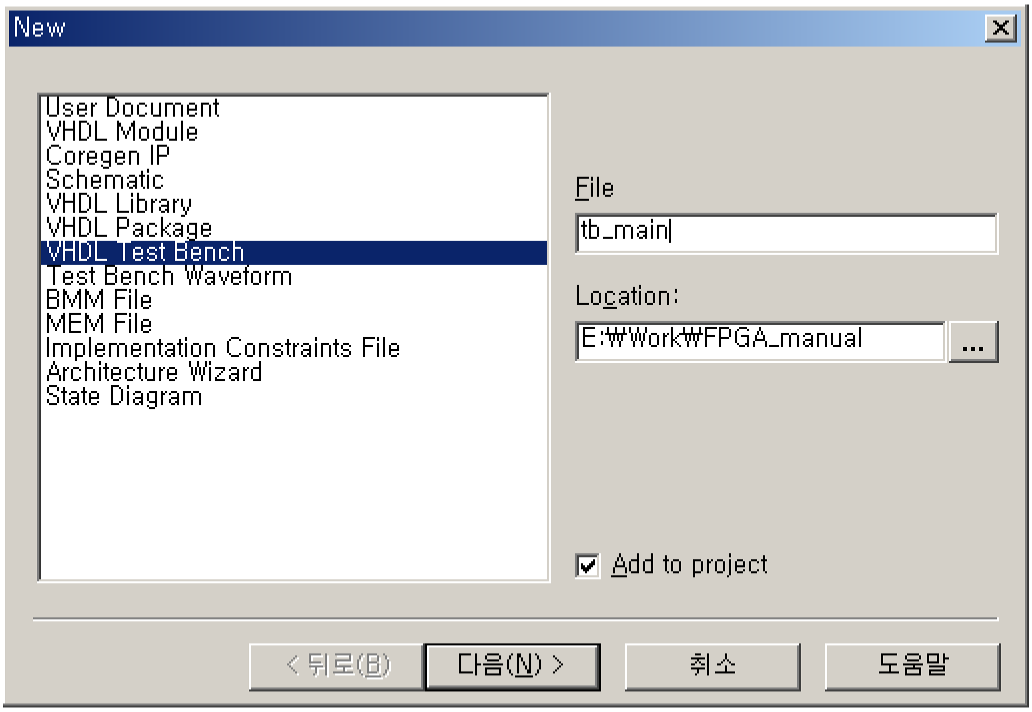
Dos command 창을 띄운 뒤, 아래와 같이 명령문을 실행합니다.

compxlib -s mti_se -f all -l all -o c: \ modeltech_5.7d \ xilinx_libs

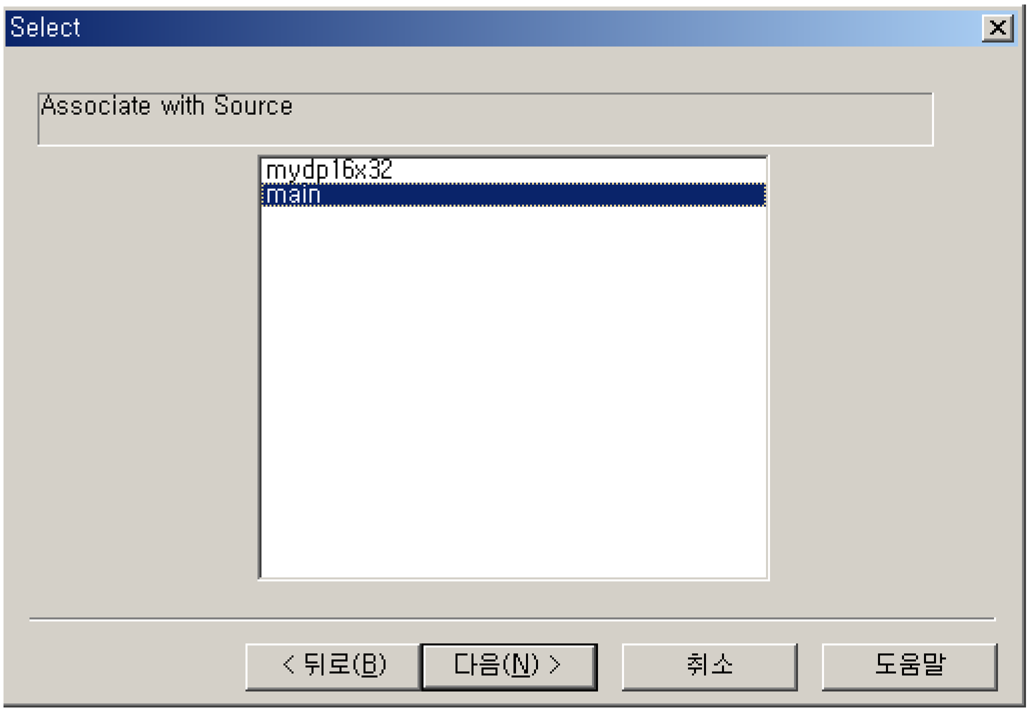
PC 사항에 따라 몇 십분 혹은 그 이상이 걸릴 수도 있습니다.
이것으로써 Modelsim에서 Simulation을 실행하는데 필요한 Xilinx library를 모두 Compile 하였습니다 .

7.2 Making Test Bench for Modelsim

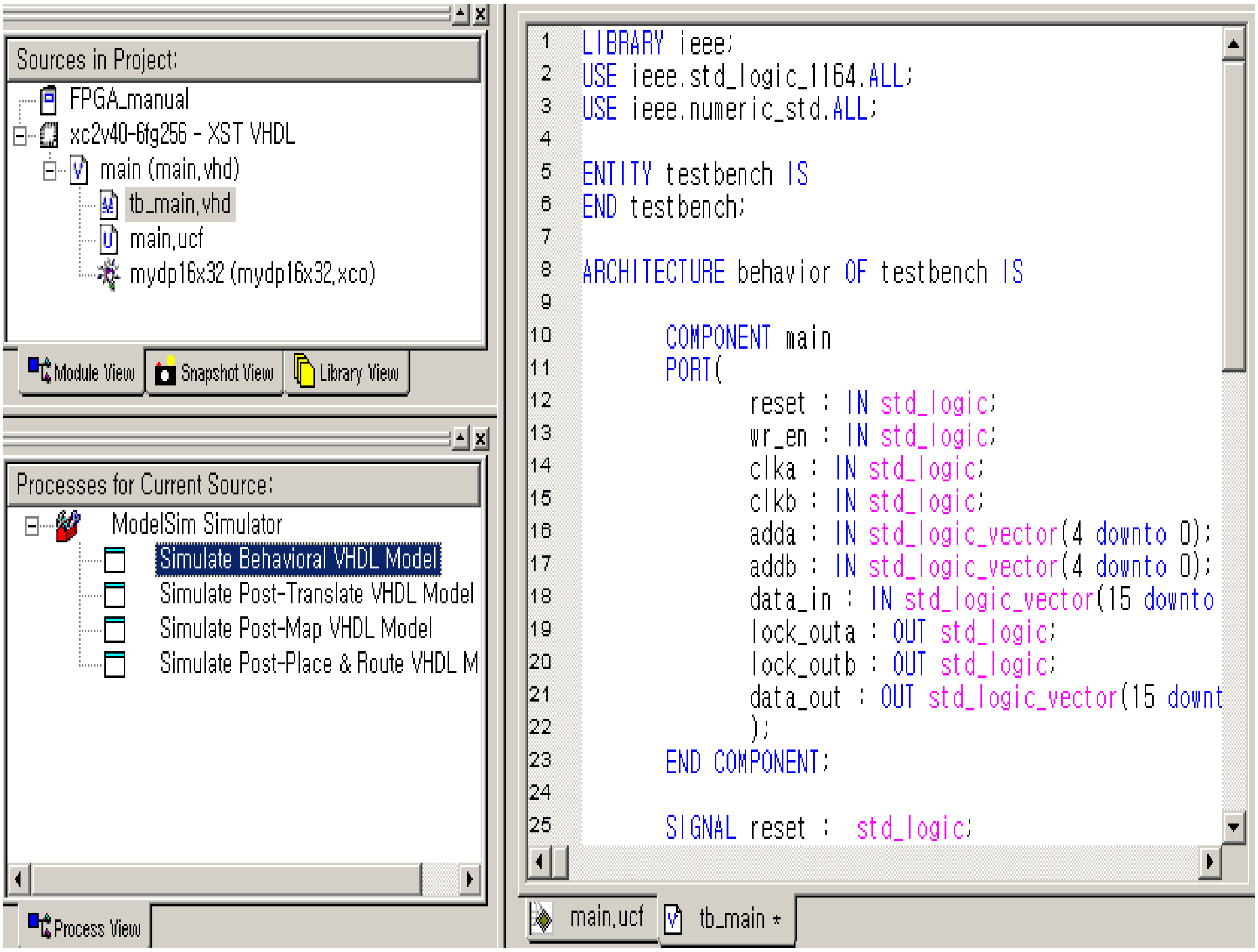
Project Navigator에서 Project -> New source... 를 선택한 후, 아래와 같이 “VHDL Test Bench를 선택합니다.



다음과 같은 Window가 나타나면, 마침 button을 Click합니다.
만약 여러 개의 entity가 존재할 경우 TOP_entity를 선택합니다.



Test Bench 파일의 초기 작성이 끝났으면, 아래와 같이 “tb_main.vhd 파일을 완성합니다.



Project Navigator창에서 tb_main.vhd에 커서를 위치 시키면 Processes for Current Source창에 Modelsim simulator가 나타납니다.

각 단계 별로 Simulation을 할 수 있습니다.

7.3 Making Test-Bench file

아래와 같이 Test bench file을 작성합니다.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
USE ieee.std_logic_unsigned.all;

ENTITY testbench IS
END testbench;

ARCHITECTURE behavior OF testbench IS

	COMPONENT main
	PORT(
		reset : IN std_logic;
		wr_en : IN std_logic;
		clka : IN std_logic;
		clkb : IN std_logic;
		adda : IN std_logic_vector(4 downto 0);
		addb : IN std_logic_vector(4 downto 0);
		data_in : IN std_logic_vector(15 downto 0);
		lock_outa : OUT std_logic;
		lock_outb : OUT std_logic;
		data_out : OUT std_logic_vector(15 downto 0)
		);
	END COMPONENT;

	SIGNAL reset :   std_logic := '1';
	SIGNAL wr_en :   std_logic := '0';
	SIGNAL clka :   std_logic := '0';             --** 초기값 설정**
	SIGNAL clkb :   std_logic := '0';
	SIGNAL lock_outa :   std_logic;
	SIGNAL lock_outb :   std_logic;
	SIGNAL adda :   std_logic_vector(4 downto 0);
	SIGNAL addb :   std_logic_vector(4 downto 0);
	SIGNAL data_in :   std_logic_vector(15 downto 0);
	SIGNAL data_out :   std_logic_vector(15 downto 0);
```

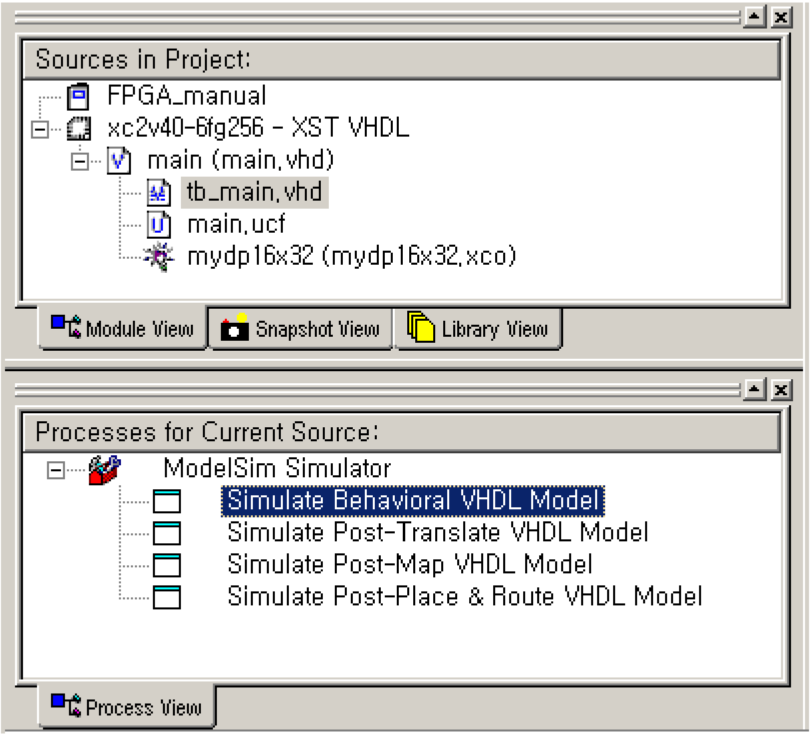
```
        signal data_temp : std_logic_vector(4 downto 0);

BEGIN

        uut: main PORT MAP(
                reset     => reset,
                wr_en     => wr_en,
                clka      => clka,
                clkb      => clkb,
                lock_outa => lock_outa,
                lock_outb => lock_outb,
                adda      => adda,
                addb      => addb,
                data_in   => data_in,
                data_out  => data_out
        );

reset <= '0' after 200 ns;              --## 위에서 초기값을 “1”로 설정했기 때문에
                                        --##  100 ns 후에 rst 값을 1 -> 0으로 변경시킴
clka <= not clka after 10 ns;           --## 10 ns 마다 Event 가 발행, Clock Period = 20 ns
clkb <= not clkb after 10 ns;

tb0 : process
        begin
        wr_en <=  '1'; wait for 150 ns;
        wr_en <=  '0'; wait for 150 ns;
end process;

tb1 : PROCESS
    BEGIN
         if now = 0 ns then
                data_temp <= (others => '0');
                wait for 20 ns;
        else
                data_temp <= data_temp +2 ;
                wait for 20 ns;
         end if;
    END PROCESS;
data_in <= "00000000000" & data_temp;
```

```
tb2 : PROCESS
    BEGIN
        if now = 0 ns then
                  adda <= (others => '0');
                  wait for 20 ns;
       else
                  adda <= adda +1 ;
                  wait for 20 ns;
        end if;

    END PROCESS;

tb3 : PROCESS
    BEGIN
        if now = 0 ns then                    --## 140 ns 가 지난 후, addb의 counter 값을
                  addb <= (others => '0');     --## 1 씩 증가시킴
                  wait for 140 ns;
       else
                  addb <= addb +1 ;
                  wait for 20 ns;
        end if;

    END PROCESS;

END;
```

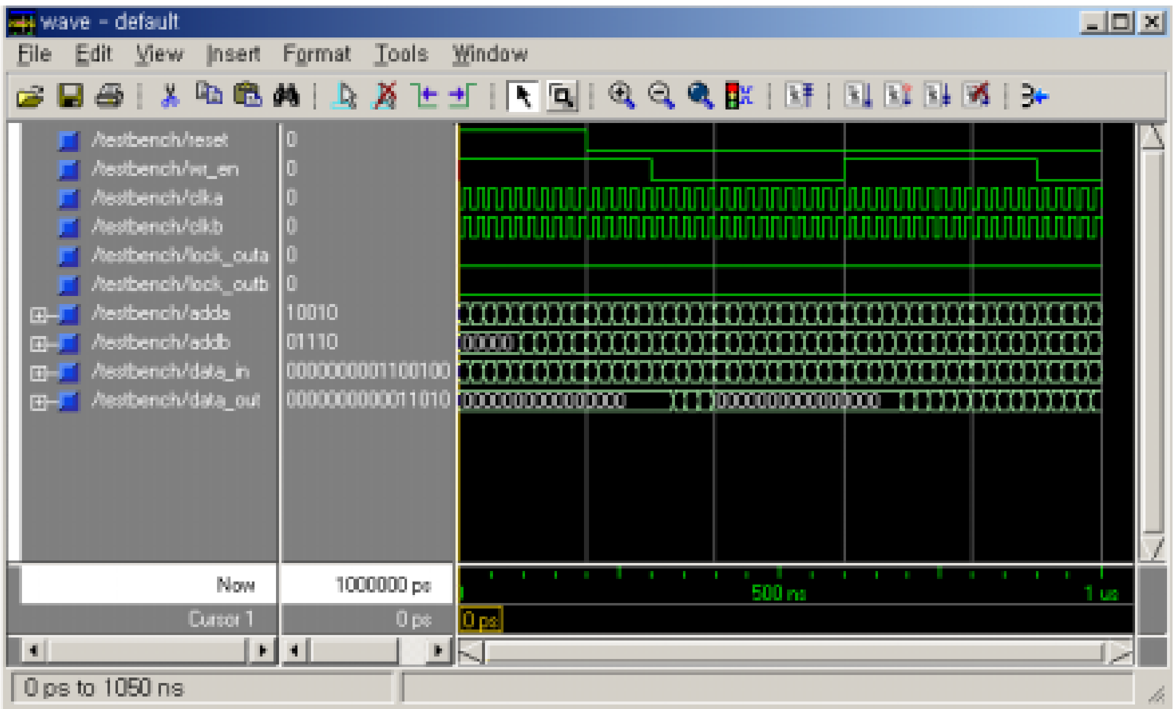
7.4 Modelsim Simulation

Test Bench file이 완성되었으면 Project Navigator의 Processes 창에서 “Simulate Behavioral VHDL Model”을 실행시킵니다. 참고로 이 Simulation은 Delay 정보가 들어 있지 않은 Functional (Language level) Simulation 입니다. Timing Simulation을 하기 위해서는 “Simulate Post-Place & Route VHDL”을 실행합니다.

이번 Lab에서는 Functional simulation 을 하겠습니다.

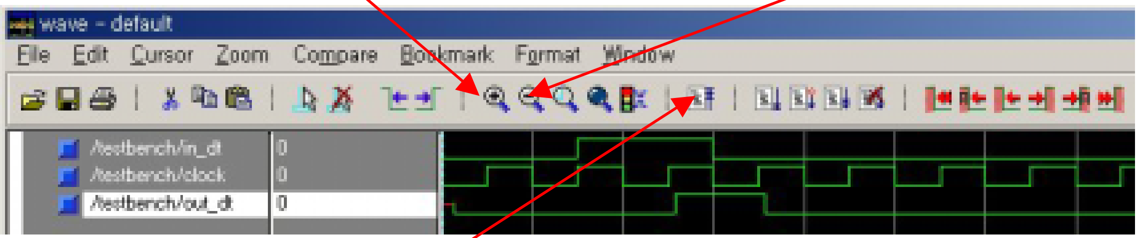


위의 Simulate Behavioral VHDL Model을 클릭하여 Modelsim simulator를 실행합니다.



- 확대(Zoom in 2X)하려면 이곳을 클릭

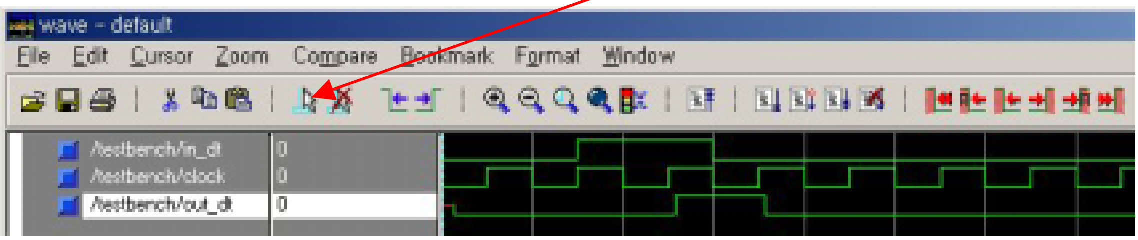
- 축소(Zoom out 2X)하려면 이곳을 클릭



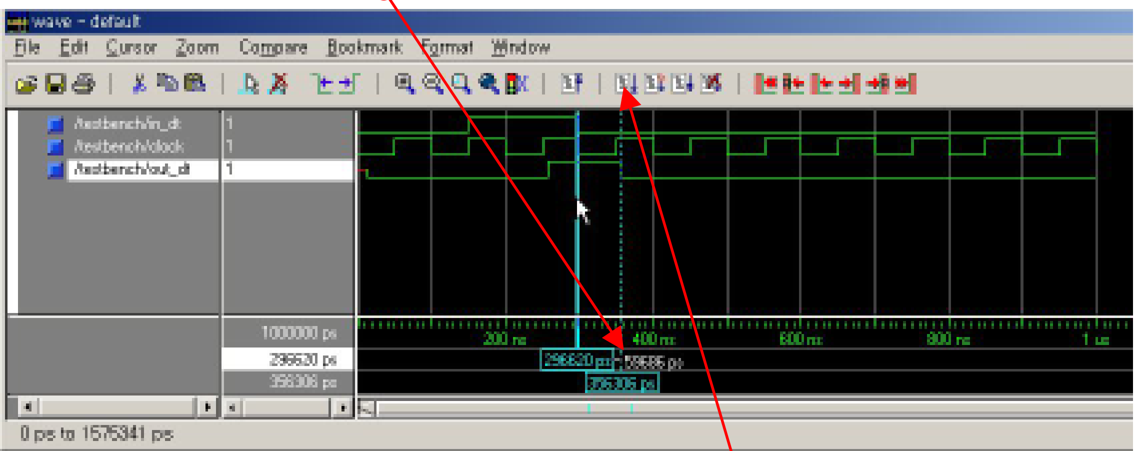
- 다시 Resimulation하려면 이곳을 클릭

- Delay를 확인하기 위해 또 하나의 지시자가 필요합니다. Add Cursor Icon을 Click합니다.

-

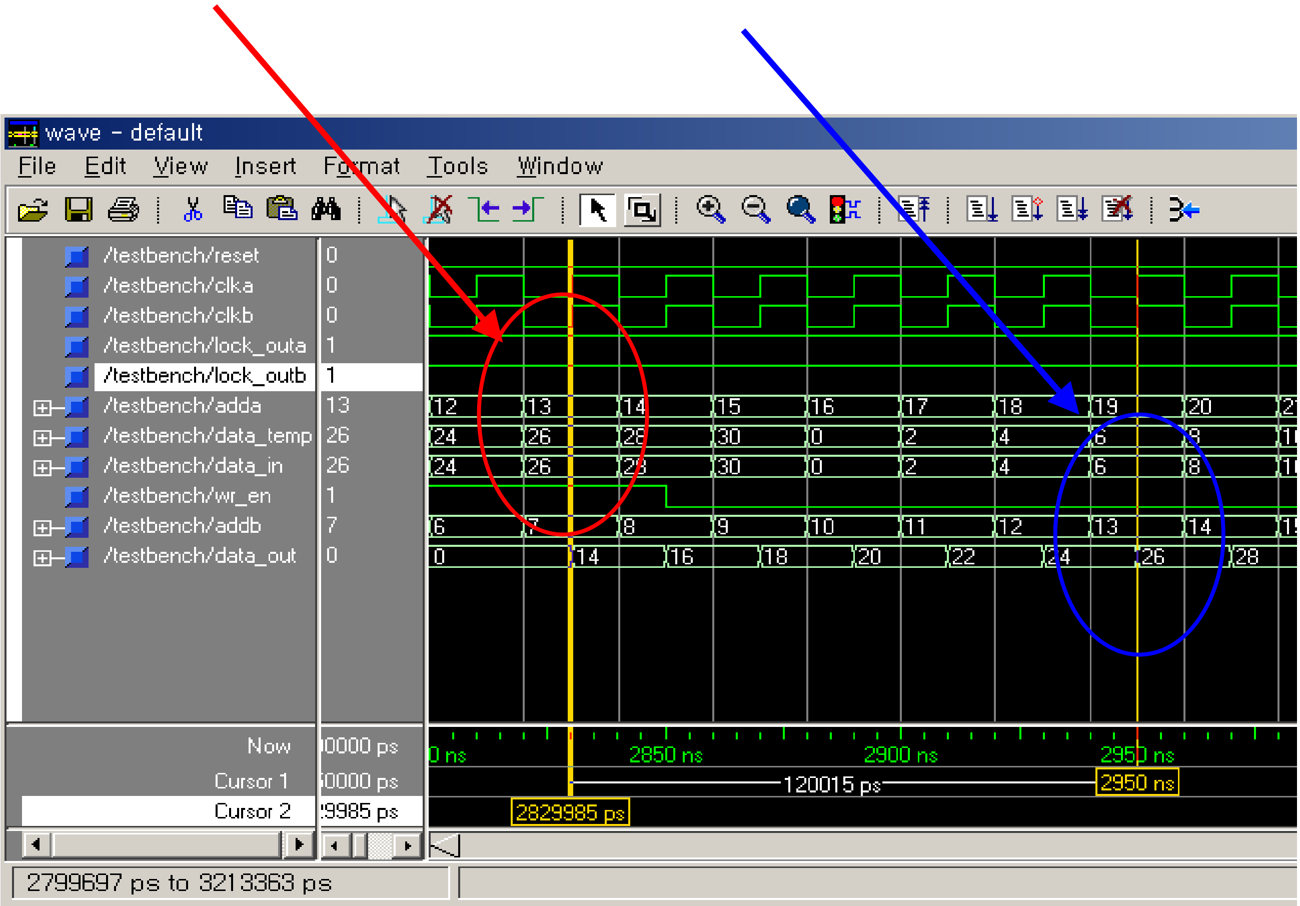


- cursor를 통해서 delay를 확인한다.



Run

지금까지 설계한 디자인이 맞는지를 Modelsim waveform 상에서 확인합니다. 아래의 그림은 Write address 때의 Data_in 값과 Read Address 때의 Data_out 값을 비교 한 값입니다



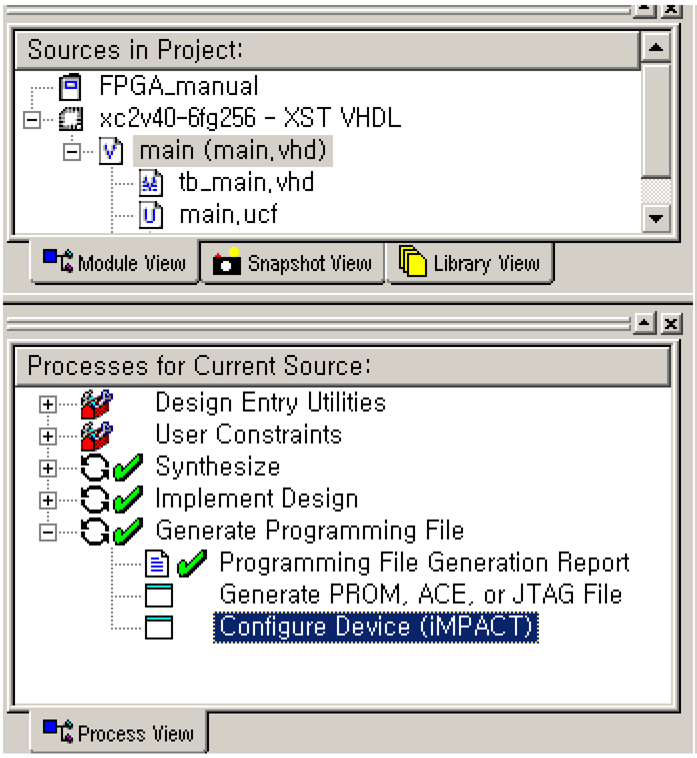
8. Downloading

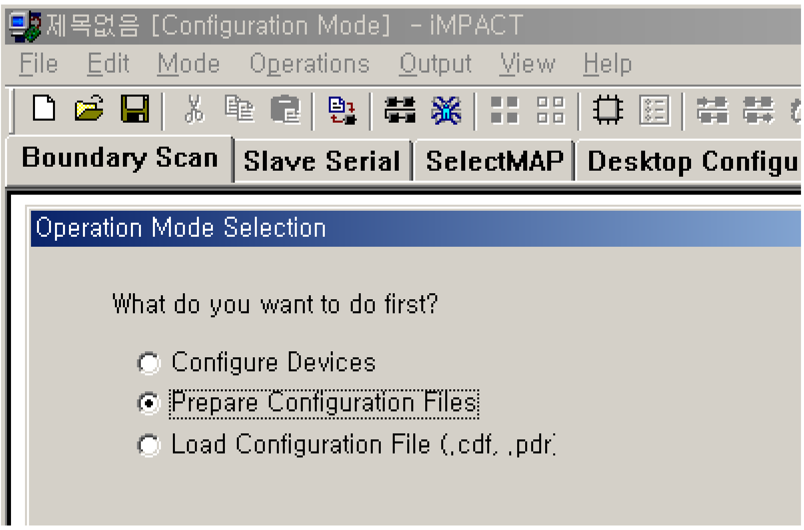
이 장에서는 설계한 디자인의 모든 검증이 끝난 후, Downloading file을 만들기 위한 작업을 설명합니다. 합니다. Bit 파일을 PROM에 맞는 MCS 파일로 만드는 과정과 MCS 파일을 이용하여 PROM에 downloading 하는 과정을 설명합니다.

** PROM을 만드는 이유는 Implementation 과정을 거치면서 만들어진 BIT 파일은 XC2V40을 Target으로 만들어졌습니다. 그러나 이 파일이 저장되고 Configuration과정이 이루어지는 디바아스는 XC2V40이 아니라 XC18v00 혹은 XC17v00이라는 PROM입니다. 따라서 XC2V40용으로 만들어진 Bit 파일은 PROM에 맞는 Format으로 변경해야만 하는 것입니다. ***

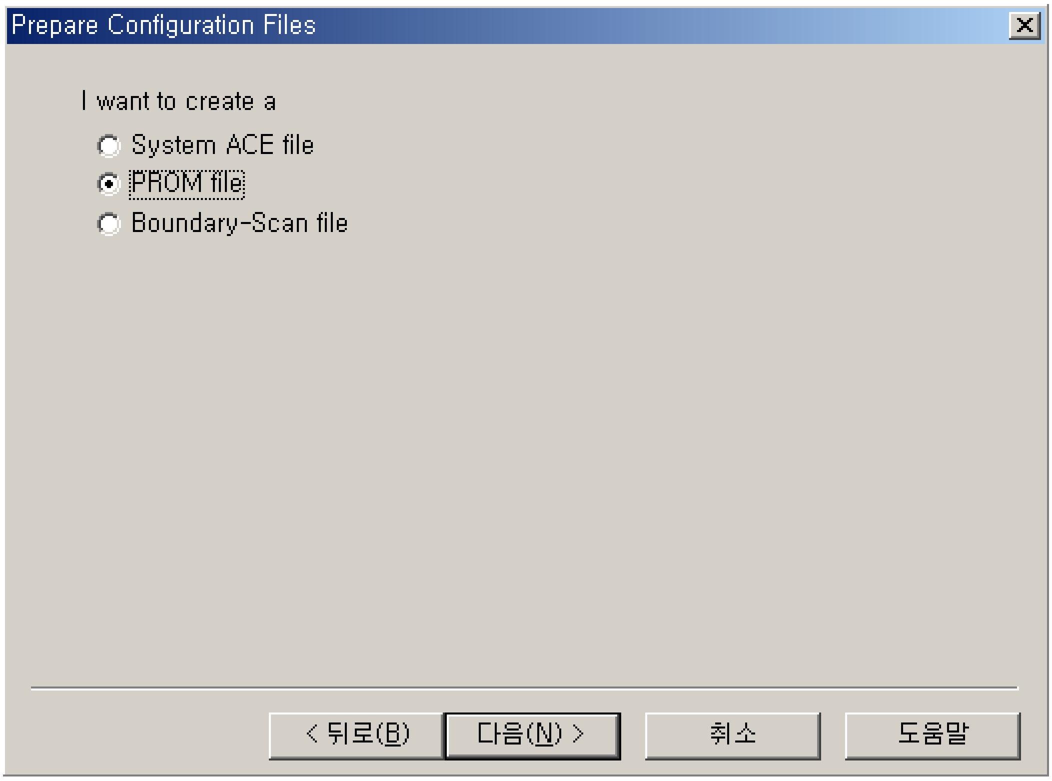
8.1. PROM 파일(MCS) 만들기

Project Navigator에서 아래와 같이 Configure Device (iMPACT)를 실행합니다.

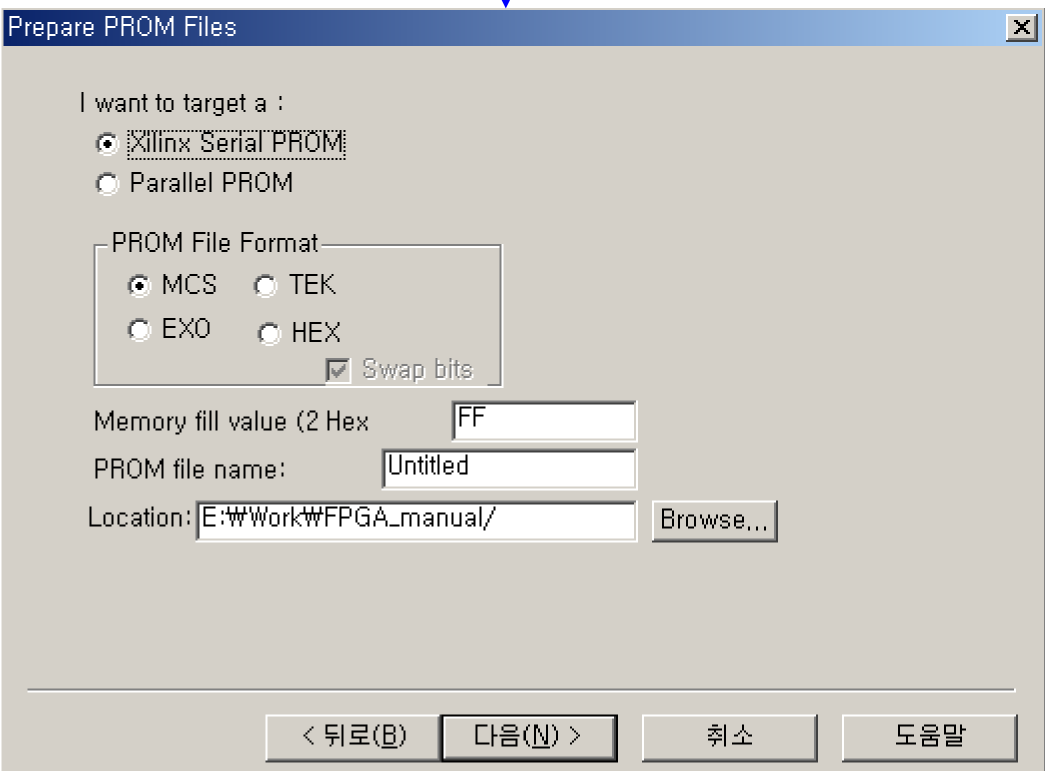




위와 같이 PROM 파일을 만들기 위하여 “Prepare Configuration File”을 선택합니다.
JTAG 파일을 이용하여 디바이스에 직접 Downloading할 때는 “Configure Devices”를 선택.

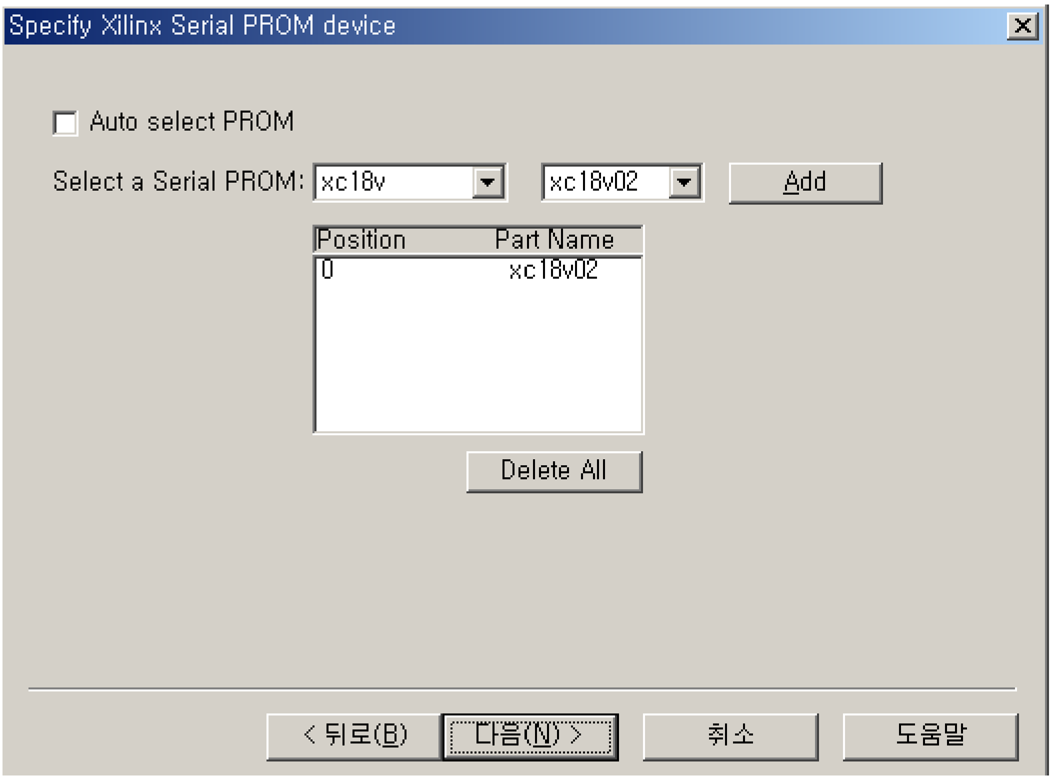


SystemACE file은 다음 Version에서 설명하겠습니다.

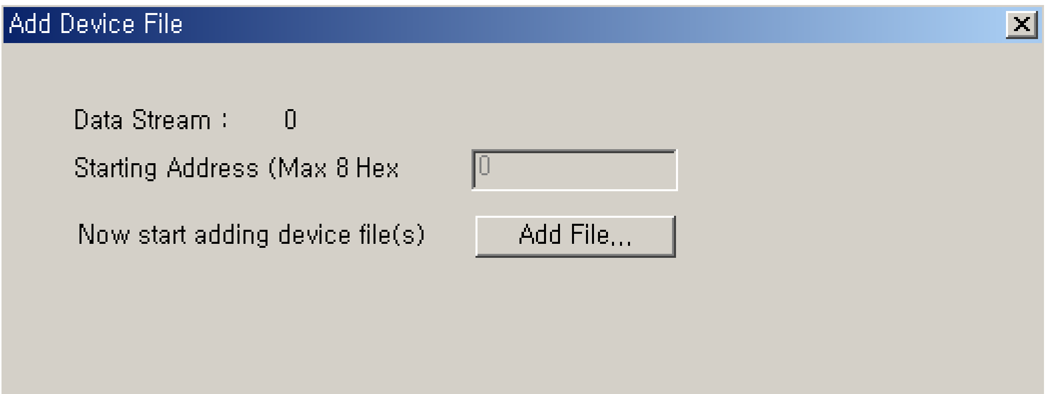


해당되는 항목이 없으면 다음을 선택

Memory fill value는 일정한 Checksum 값을 유지하기 위해 "FF"로 합니다.

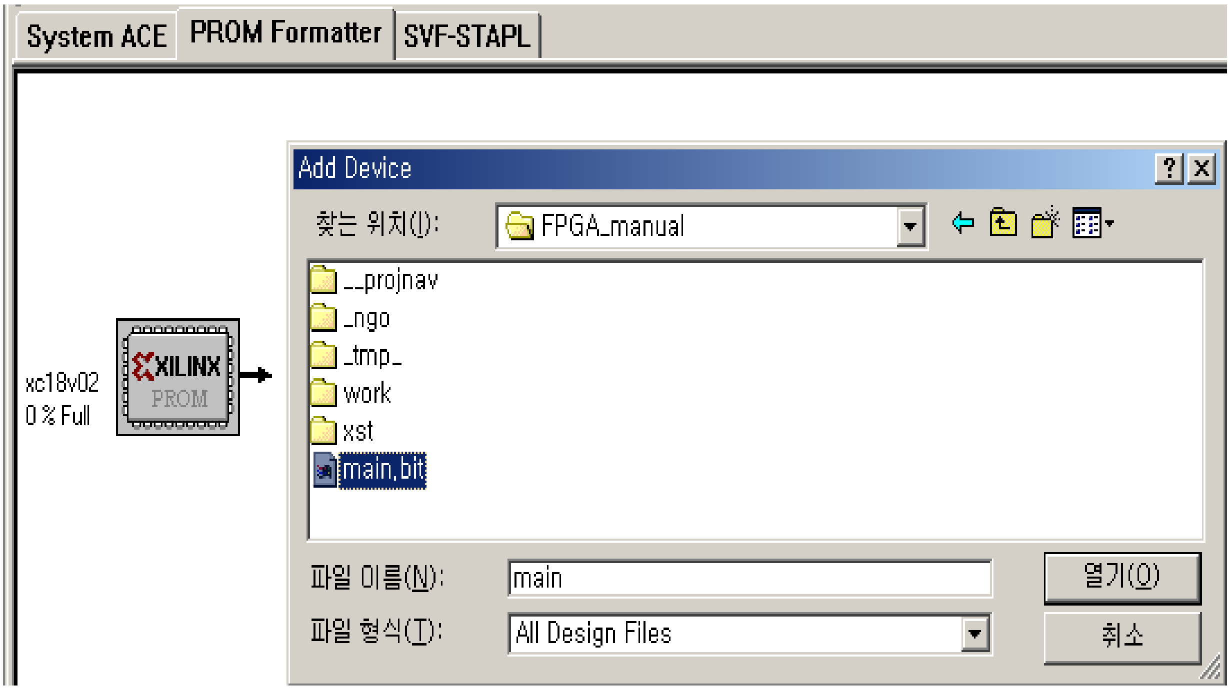


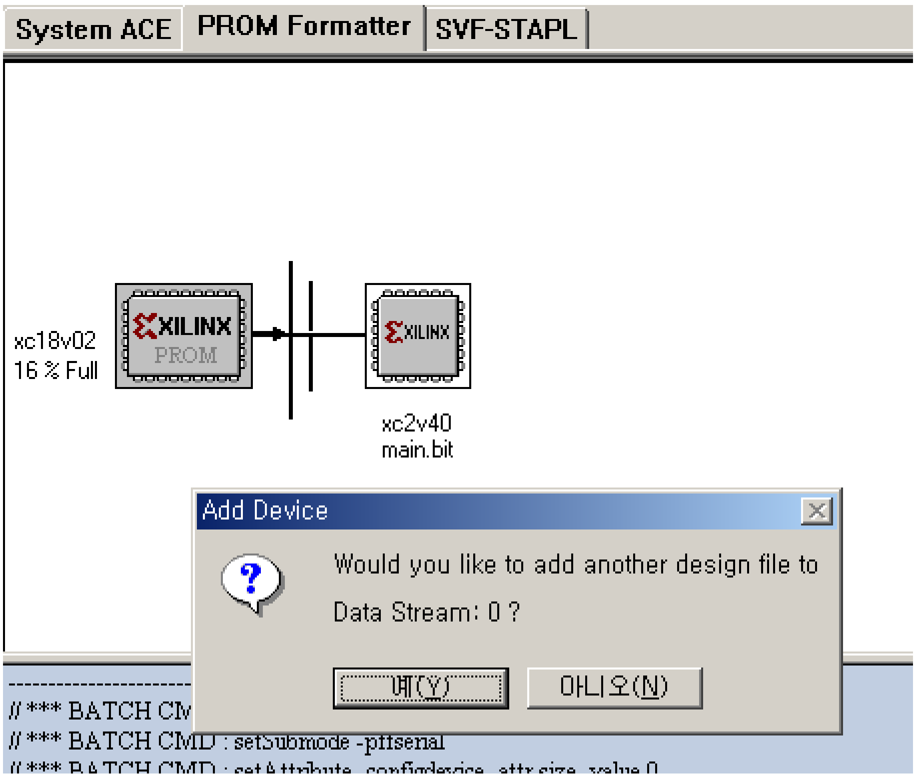
Targeting 될 PROM을 선택한 후 “Add”를 합니다. 만약 여러 개의 PROM이 사용되는 경우라면 “Add”를 이용 하여 PROM을 추가합니다. 이번 LAB에서는 XC18V02-PC44 한 개를 사용하기 때문에 위에서와 같이 XC18V02 PROM 한 개만은 Add 합니다.



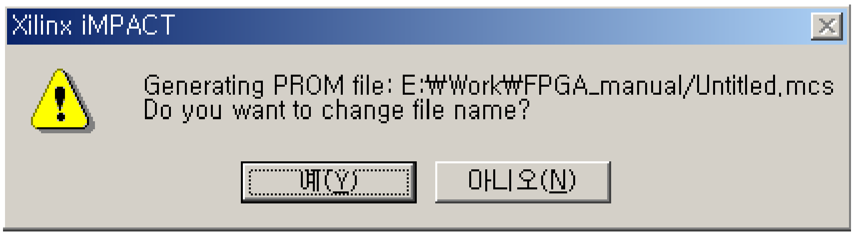
“Generate Programming File 후에 만들어진 BIT 파일은 선택하기 위하여 위의 “Add File” 아이콘을 선택합니다.

아래의 그림은 Generate Programming File 과정에서 만들어진 main.bit 파일을 선택하는 것을 설명합니다.

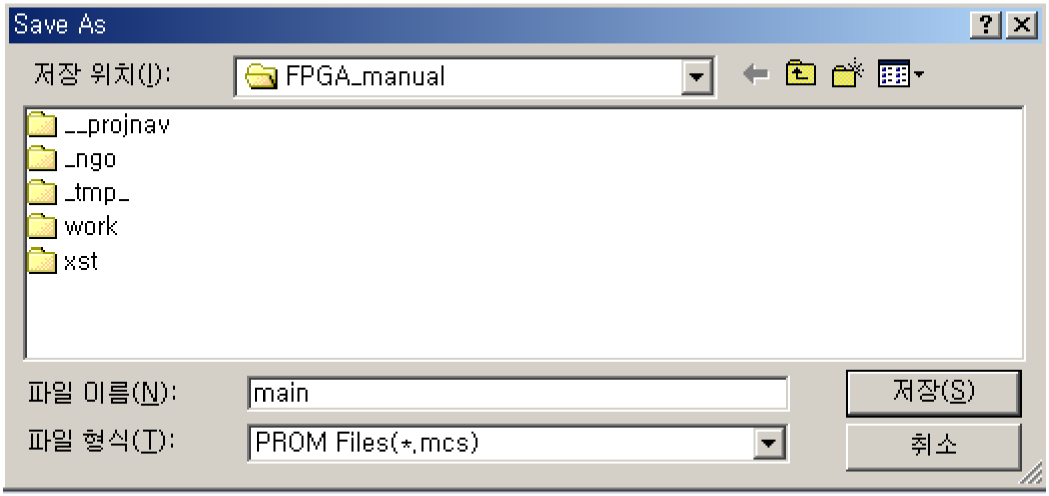




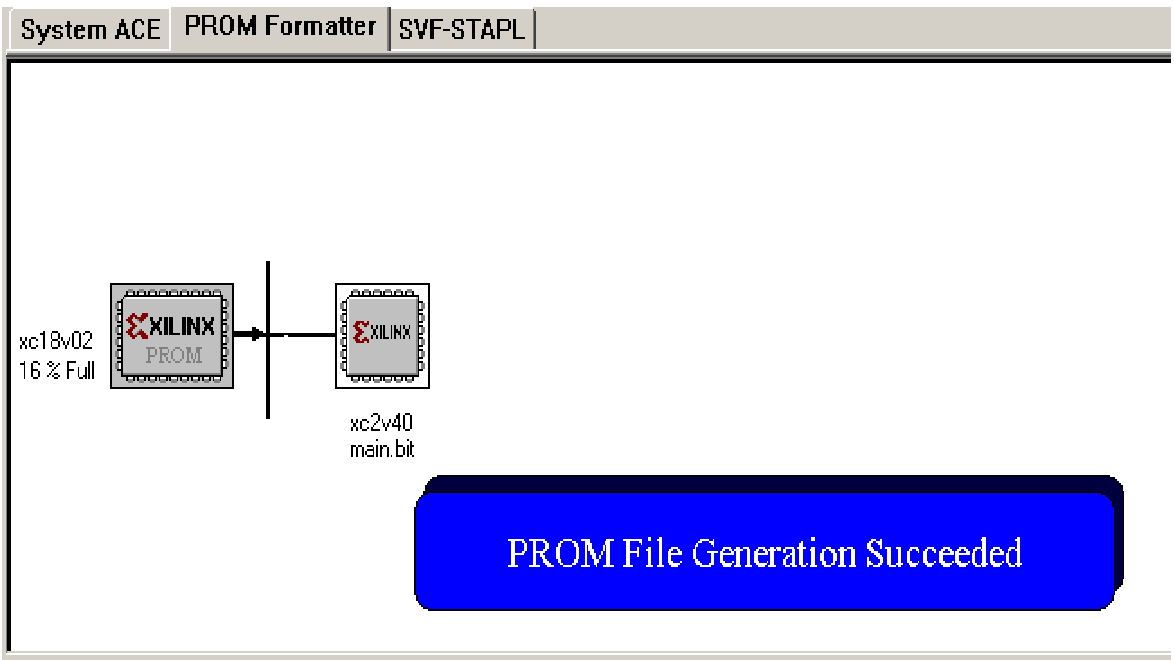
Main.bit 파일을 선택한 후 위와 같은 화면이 뜨게 됩니다. 이것은 여러 개의 FPGA가 Daisy chain으로 연결되어 있을 경우 사용하며, File 추가를 위한 창이 뜹니다.
여기에서는 "아니오"를 선택합니다.



예를 선택.

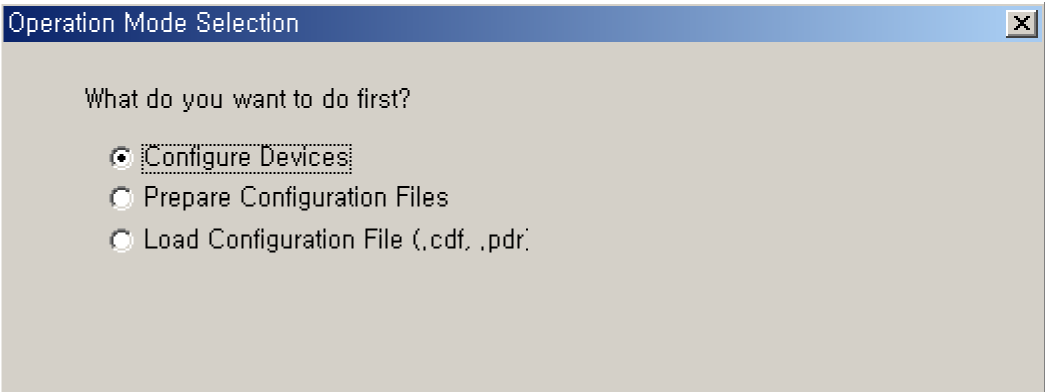


저장을 선택합니다. 이것으로서 main.mcs 파일이 만들어 지게 됩니다.

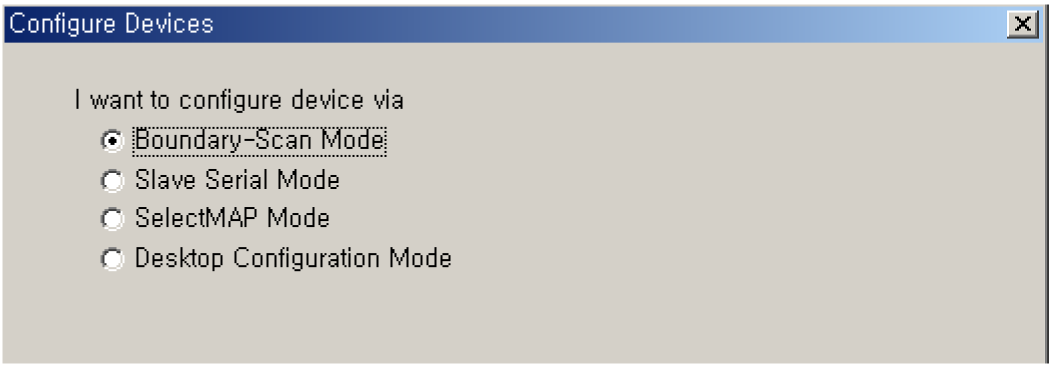


8.2. Configuration into the board

이 LAB의 마지막 단계로서 위에서 만들어진 main.mcs 파일을 가지고 downloading cable 을 가지고 Downloading 하는 과정을 설명합니다.
PC와 Target board를 download cable로 연결한 후, Target board의 전원을 넣습니다.



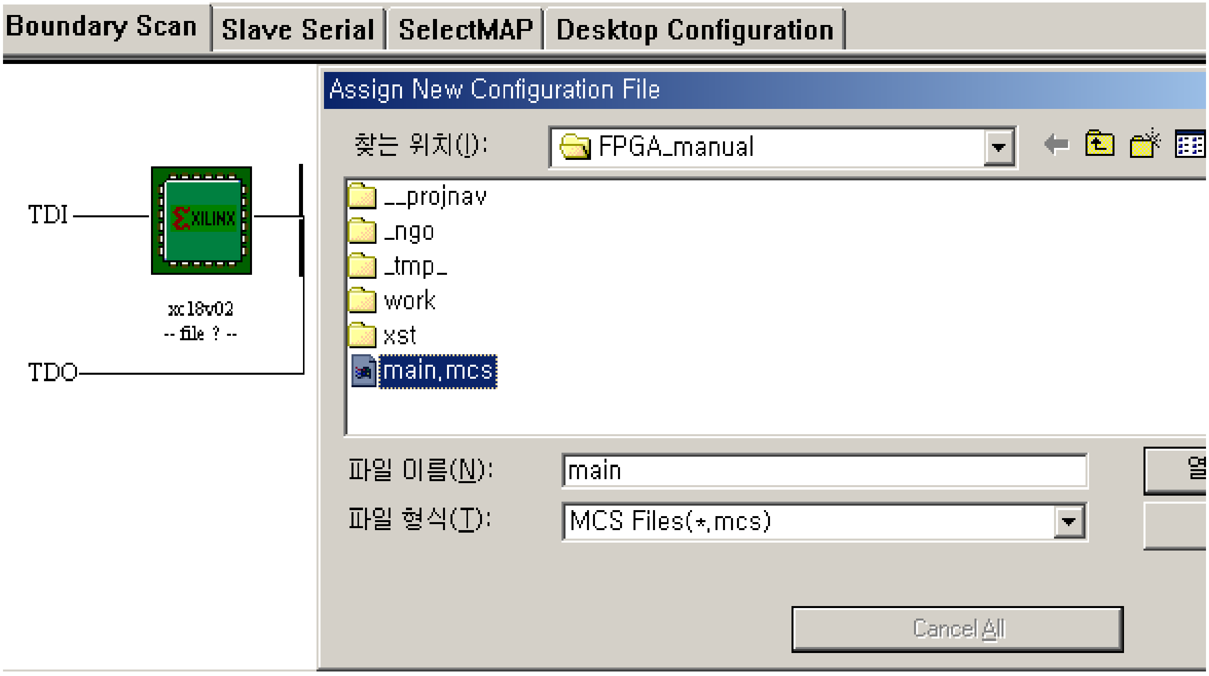
이번 단계에서는 "Configure Device"를 선택합니다.



아래와 같은 화면이 나올 때 까지 다음을 계속해서 선택합니다.

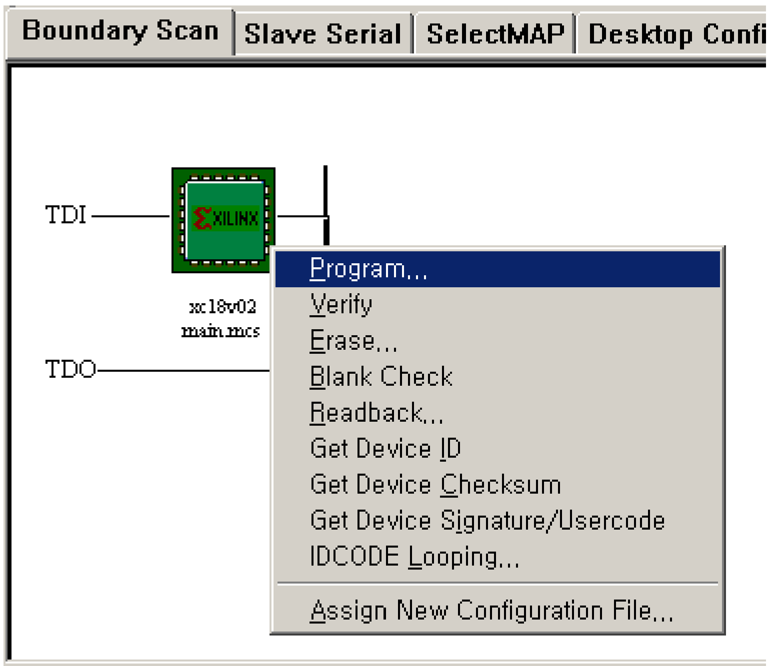
*** iMPACT의 디바이스 아이콘에 XC18V02가 나타나지 않는다면 PC와 Target board가 Hardware 적으로 연결이 제대로 이루어지지 않았다는 것을 의미합니다. 이 때는

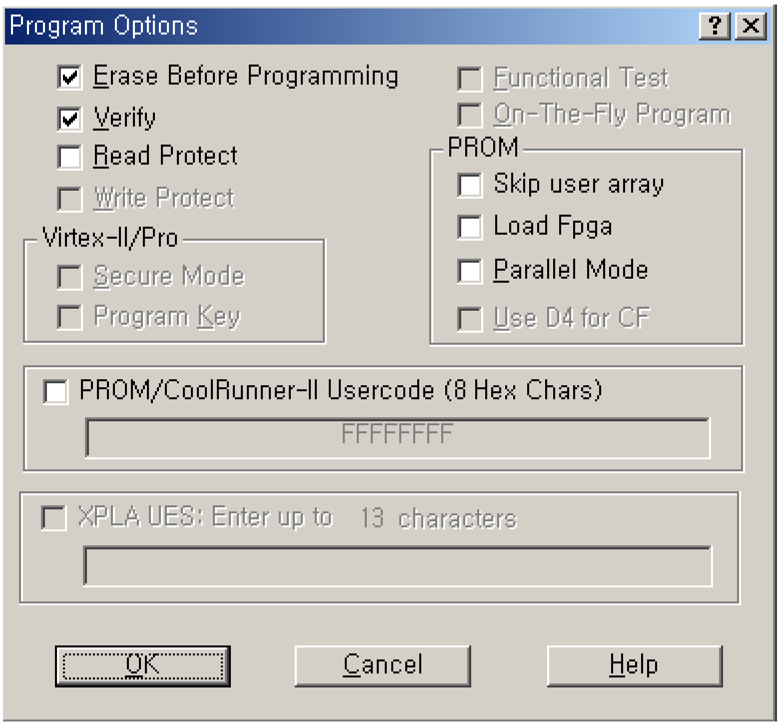
1. Cable 이 정상적으로 연결되었는지(Cable의 Vcc와 GND 가 연결되었는지).
2. Cable 고장 여부를 확인하기 위해서 2개 이상의 Cable로 Cross check를 합니다.
3. PC의 Parallel port(Printer port)가 이상 유무를 확인하기 위하여 다른 PC에서 Tool을 설치한 후 검사합니다. .



모든 준비가 완료되었으면 실제로 보드상에서 Programming을 실시합니다.

IMPACT 화면상에 있는 PROM icon을 마우스 오른쪽 단추로 선택하면 아래와 같이 한 후, Programming을 선택합니다.





이것으로서 MCS 파일을 PROM에 downloading하는 과정을 마치게 됩니다.

9. Testbench Examples

Case1 : 초기화 신호 생성

```
ENTITY testbench IS
END
ARCHITECTURE behavior OF testbench IS
  COMPONENT xxx
  PORT(reset1 : IN STD_LOGIC;
        reset2 : IN STD_LOGIC;
        ....);
  END COMPONENT;
  Signal reset1, reset2 : STD_LOGIC := ‘0’;
BEGIN
  reset1 <= ‘0’, ‘1’ AFTER 100 ns;
  reset2 <= ‘0’, ‘1’ AFTER 100 ns, ‘0’ AFTER 3000 ns, ‘1’ AFTER 3100 ns;
END;
```

Case2 : 반복적 특성의 신호 생성

Clock 과 같이 주기적으로 변화되는 입력 신호

```
ENTITY testbench IS
END
ARCHITECTURE behavior OF testbench IS
   COMPONENT xxx
   PORT(clock1 : In STD_LOGIC;
                  clock2 : In STD_LOGIC;
                  clock3 : In STD_LOGIC;
                  clock4 : In STD_LOGIC;
                        ...);
   END COMPONENT;
   CONSTANT half_period   : TIME := 50 ns;
   CONSTANT full_period   : TIME := 100 ns;
   SIGNAL clock1, clock2, clock3, clock4: STD_LOGIC := ‘0’;

BEGIN
   clock1 <= NOT clock1 AFTER 50 ns;
   clock2 <= NOT clock2 AFTER full_period/2;

   clk_gen3:process
   begin
      wait for full_period/2;      clock3 <= '1';
      wait for full_period/2;      clock3 <= '0';
   end process;

clk_gen4:process
   begin
      wait for half_period*2;      clock4 <= '0';
      wait for half_period*2;      clock4 <= '1';
   end process;
END;
```

Case3 : Random하게 변화하는 신호의 생성

Case3-1 : 자기가 원하는 값을 random 하게 발생 시키는 경우.

```
  SIGNAL sig_c : STD_LOGIC_VECTOR(2 DOWNTO 0);
BEGIN
  PROCESS
  BEGIN
    sig_c <= "000";     WAIT FOR 100 ns;
    sig_c <= "001";     WAIT FOR 100 ns;
    .....
    sig_c <= "110";     WAIT FOR 100 ns;
    sig_c <= "111";     WAIT FOR 100 ns;
  END PROCESS;
SIGNAL d_in : STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
  d_in <= "0000", "0001" after 20 ns, "0010" after 40 ns,
                "0011" after 60 ns , "0100" after 80 ns,
                "0101" after 100 ns, "0110" after 120 ns,
                "0111" after 140 ns;
….
  SIGNAL sig_a, sig_b   : STD_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN
  sig_a <= "00010101";
  sig_b <= "00001101";
```

Case3-2 : 임의의 값을 random 하게 발생하는 경우

```
SIGNAL sig_a   : STD_LOGIC_VECTOR(3 DOWNTO 0) := “0000”;
BEGIN
  PROCESS
  BEGIN
    sig_a <= sia_a + 1;
    WAIT FOR 100 ns;
  END PROCESS;
```

10. Revision Table

2003. 6. 25   ISE Foundation 5.2i Manual   ver1.0 Release