XILINX®

BASIC TRAINING

For Beginners

Hyun Myung Electronics Co., Ltd.
Tel 02-3141-0147 / Fax 02-3141-0149
http://www.hmelec.co.kr

XS AT CE TP

XILINX®

BASIC TRAINING

Installation

Hyun Myung Electronics Co., Ltd.
Tel 02-3141-0147 / Fax 02-3141-0149
http://www.hmelec.co.kr

XS AT CE TP

Foundation Series Software Overview.

안녕하십니까? 저희 XILINX 사의 Foundation Software 를 사용하게 되신 점을 진심으로 축하합니다. 우선 저희 Tool 을 간략하게 설명하면, 모든 Design Entry Tool 과 Design Verification Tool, 그리고 Design Implementation Tool (Place and Route / Fitting)이 하나로 통합되어서 사용할 수 있는 Total Solution 을 제공할 수 있는 소프트웨어 입니다.

여러분이 디자인을 전형적인 방법을 택해서 Schematic Capture 로 하든, HDL Language 인 VHDL 또는 Verilog 그리고 ABEL 을 사용하시든, 이 모든 것을 혼합하여 사용하시든 (Mixed Design) 아무 문제없이 통합되고 동일한 Interface 의 디자인 환경을 제공하므로 단지 마우스를 원하는 동작에 맞춰 Click 만 하시면 디자인 설계서부터, 검증 그리고 구현을 단 시간 내에 하실 수 있습니다.

자 그러면 이 소프트웨어를 여러분의 컴퓨터에 설치해보도록 합시다. 먼저 갖고 계신 Foundation Series Software Package 에서 CD-ROM Package 를 찾도록 합시다. 여러분이 Foundation Series EXPRESS 또는 Foundation Series BASE-EXPRESS Package 를 가지고 있다면 CD-ROM Package 에 여섯장의 CD-ROM 이 들어 있을 것입니다. 간단하게 설명하면

1. **Design Environment Foundation Series 1.5 CD-ROM**
   Design Entry 와 Design Implementation 이 들어 있는 CD-ROM
2. **Documentation Foundation Series 1.5 CD-ROM**
   Xilinx 에서 제공하는 각종 Manual 및 Reference Guide Book 이 들어 있는 CD-ROM
3. **MasterClass Lite VHDL Tutorial Foundation Series 1.5 CD-ROM**
   VHDL 을 학습하여 배울 수 있도록 만든 Multi-media 지향의 VHDL 학습 CD-ROM
4. **CORE Generator**
   Xilinx 사의 모든 Core 에 대한 정보와 core program 을 담고 있는 CD-ROM
5. **Model*Sim* Evaluation CD-ROM**
   VHDL Simulator 인 Modelsim 을 평가할 수 있는 CD-ROM
6. **Active-VHDL Evaluation CD-ROM**
   VHDL Simulator 인 Active-VHDL 을 한달 동안 설치하여 사용해 볼 수 있는 CD-ROM

만일 여러분이 Foundation Series BASE Package 을 가지고 계신다면 CD-ROM package 에는 위의 1, 2, 3 의 CD-ROM 만이 들어 있을 것입니다.

Foundation Series Software Installation.

여러분의 컴퓨터 CD-ROM Drive 에 1) CD-ROM 을 넣습니다. 여러분 중에 복사본 CD-ROM 을 가지고 있으면 그 CD-ROM 을 CD-ROM Drive 에 넣습니다. 시스템이 자동으로 설치프로그램을 구동 시키지 않으면 윈도우 탐색기를 이용하여 CD-ROM Drive 로 이동하여 setup.exe 프로그램을 실행시키시면 됩니다.

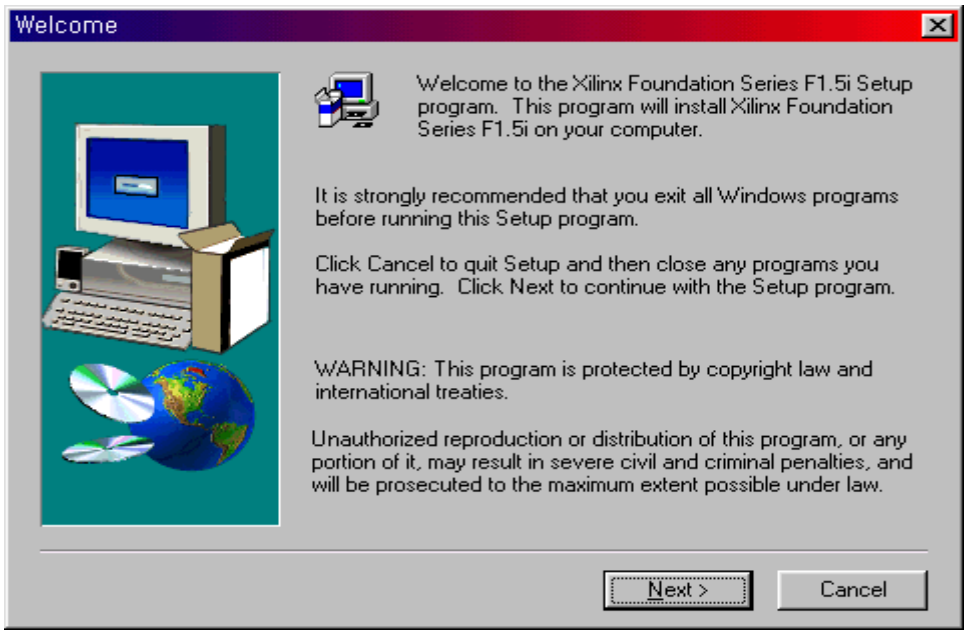


그림 1 Welcome Dialog Window

그림 1 이 나타나면 Next Button 을 눌러 설치를 계속 진행시킵니다.

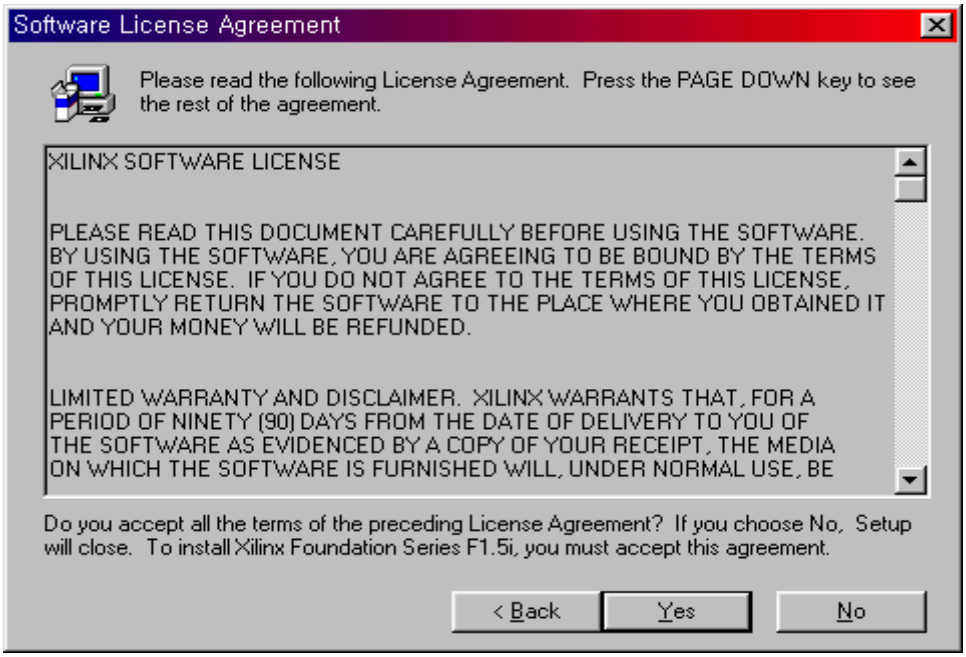


그림 2 Software License Agreement Window

그림 2 는 License Agreement 을 묻는 그림입니다. Yes Button 을 눌러 설치를 계속 합니다.

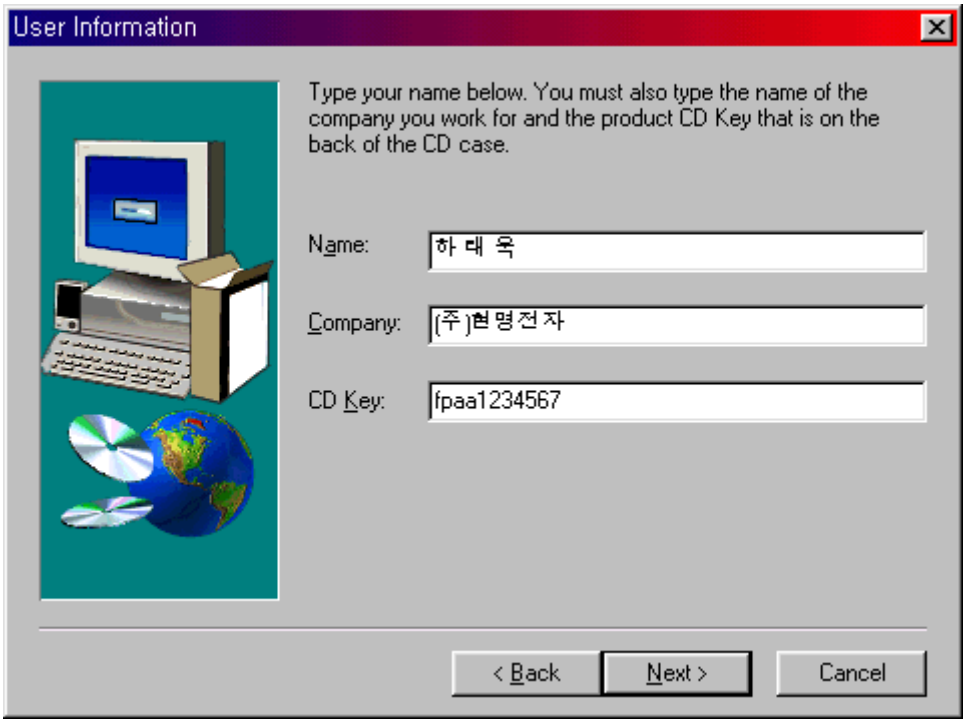


그림 3 User Information Window

그림 3 은 CD key 를 넣는 부분입니다. 여러분이 구입하신 CD-ROM Package 뒷면에 CD KEY 라고 흰색라벨이 붙어 있을 것입니다. 그 CD-KEY 를 입력해주시길 바랍니다.

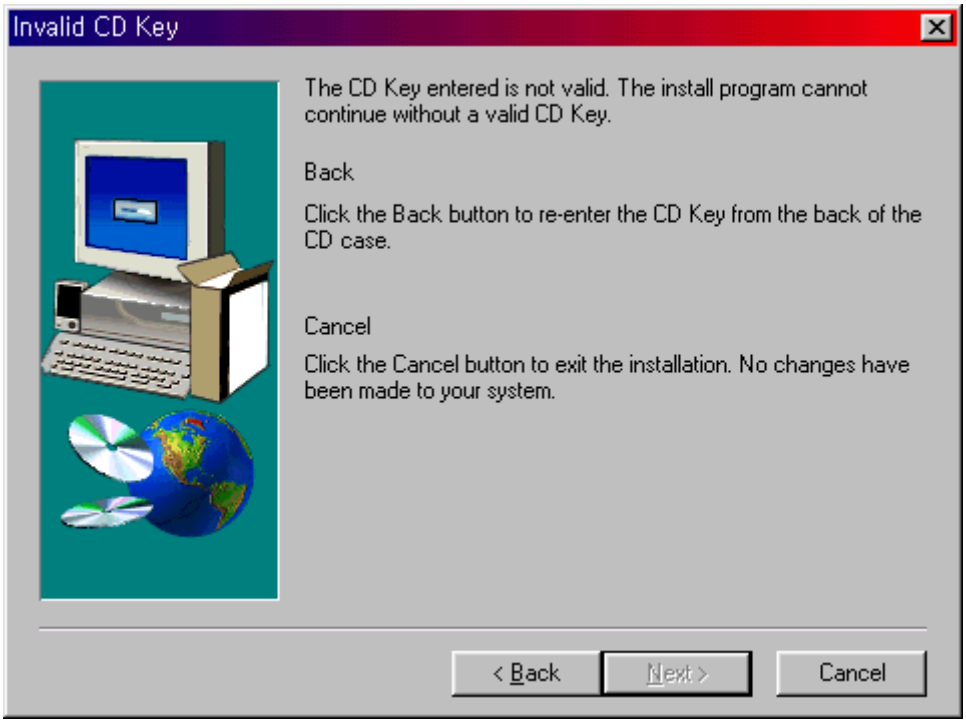


그림 4 Invalid CD Key Window

그림 4 는 올바른 CD Key 를 입력하지 못했을 때 나오는 화면입니다. Back button 을 눌러 다시 전 화면으로 돌아가서 CD-KEY 을 정정하여 입력하시길 바랍니다.

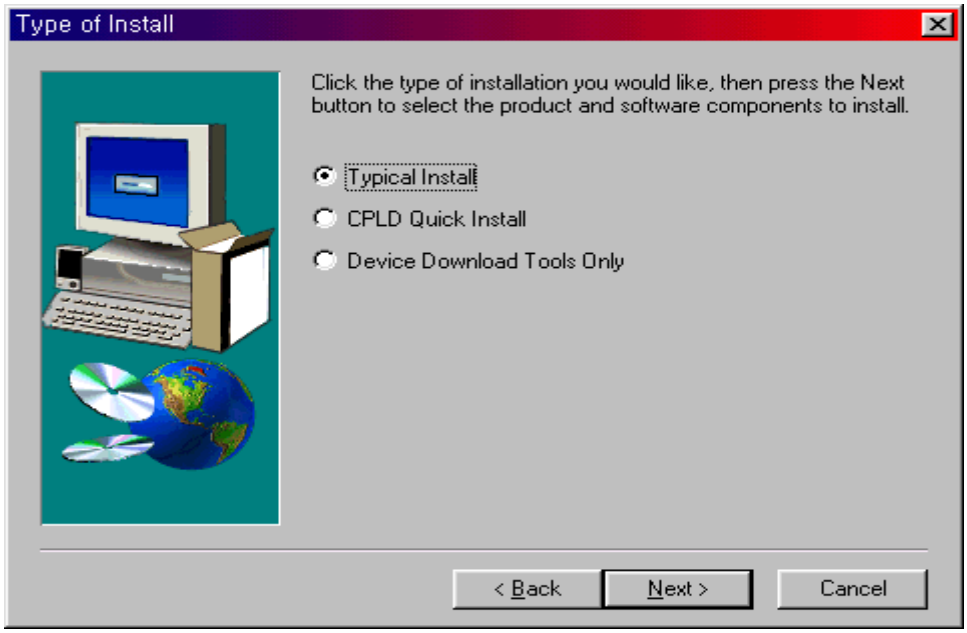


그림 5 Type of Install Window

적법한 CD Key 를 넣었으면 그림 5 의 화면으로 연결이 될 것입니다. Typical Install 은 Foundation Software 의 모든 부분을 선택하여 설치하는 것입니다. CPLD Quick Install 은 CPLD 에 관련된 부분만 선택하여 설치할 때 사용하면 됩니다. Device Download Tools Only 는 제품생산 라인처럼 전

체 프로그램은 필요 없고 Download Program 인 JTAG Programmer 만을 설치하여 XC9500 을 ISP 을 이용하여 Writing 할 때 설치합니다. 여기서는 Typical Install 을 선택하여 설치를 계속하기로 하겠습니다.

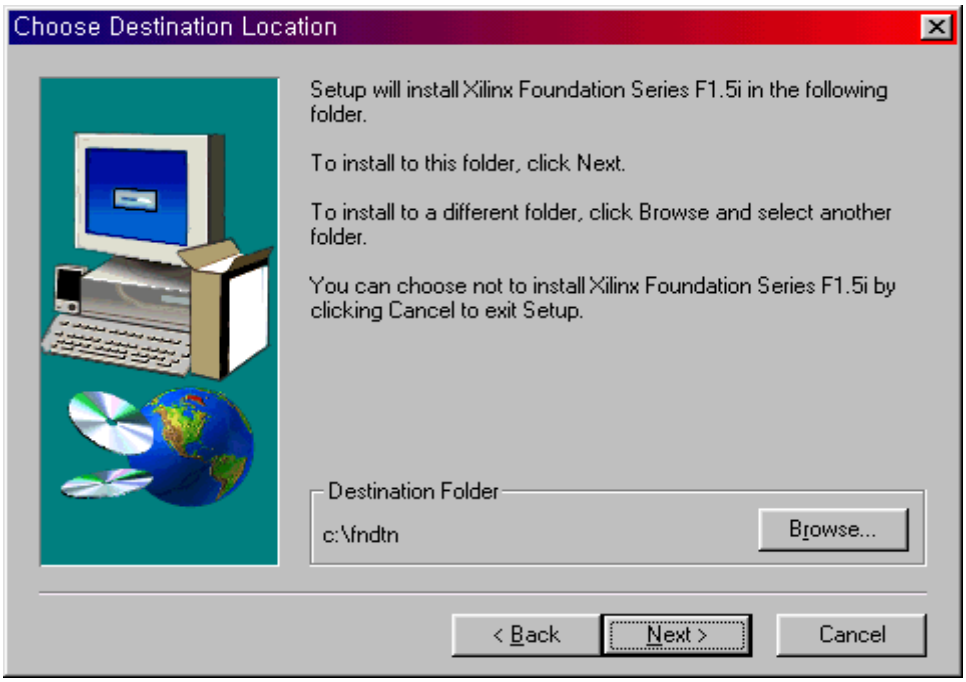


그림 6 Choose Destination Location Window

그림 6 의 화면에서 Default 로 프로그램이 설치될 장소를 C:\fndtn directory 로 결정을 합니다. 만일 여러분이 경로를 바꿔서 설치를 하시길 원하시면 Browse 버튼을 눌러 여기서 원하는 경로명을 적으시거나 선택하시기 바랍니다. 원하는 경로를 적고 나서 Next Button 을 눌러 설치를 계속합니다.

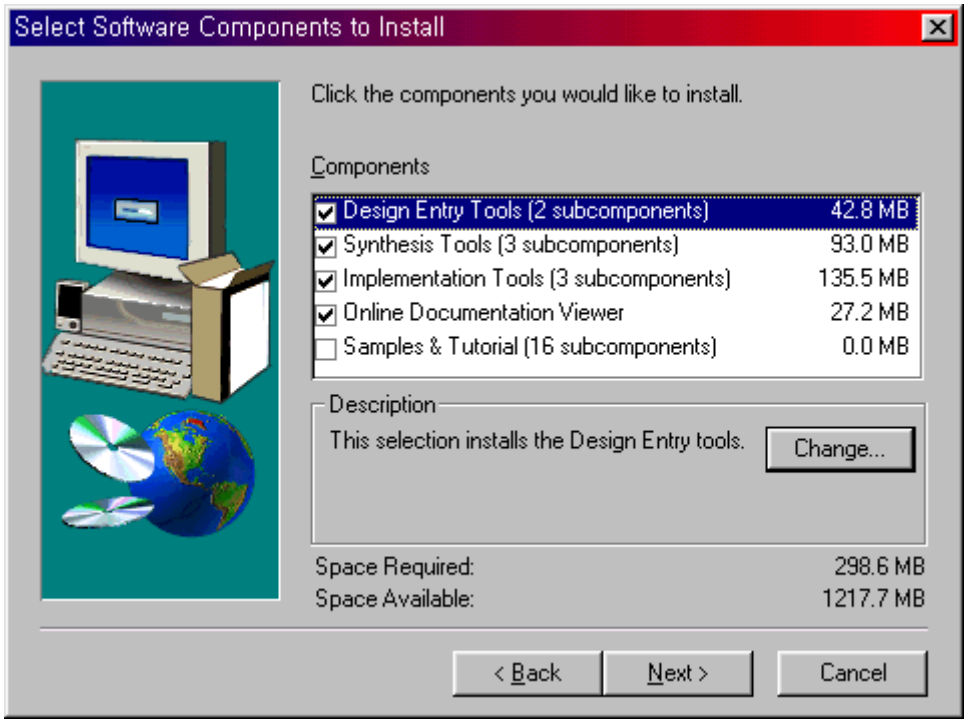


그림 7 Select Software Components to Install Window

그림 7 이 보이면 Samples & Tutorial 이 선택이 되어 있지 않을 것입니다. 가급적이면 선택하여 기본 예제를 참조할 수 있도록 하고 디자인 시 필요한 자료들을 활용할 수 있도록 합시다. Next Button 을 눌러 설치를 계속 합니다.

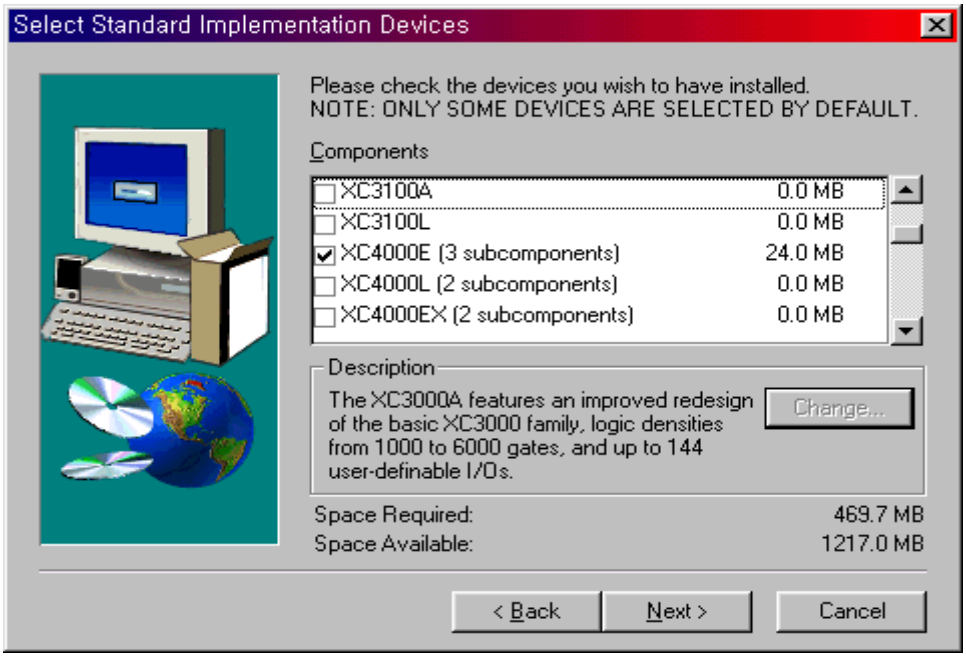


그림 8 Select Standard Implementation Devices Window

그림 8 은 사용할 디바이스를 선택하는 부분입니다. 가이드 라인을 정해보면 만일 기존에

XC3000 Family 를 사용하셨으면 XC3000 을 선택하시면 됩니다만 현재로서는 거의 선택할 일이 없을 것으로 생각됩니다. 오른쪽 상하 이동 버튼을 눌러 원하는 디바이스 Family 를 선택하시길 바랍니다. 기본적으로 설정된 것을 이용하시면 별 다른 무리가 따르지 않을 것이나 저희가 추천하는 디바이스 제품군은 XC4000XL, XC4000XLA, XC9500, XC9500XL, SPARTAN, SpartanXL 입니다. XL 또는 XLA 로 표시된 제품군은 Power Voltage 가 3.3Volt 입니다. 그외 디바이스는 5Volt 전원을 사용합니다. 설치를 원하는 제품군들을 선택하셨으면 Next Button 을 눌러 설치를 계속합니다.

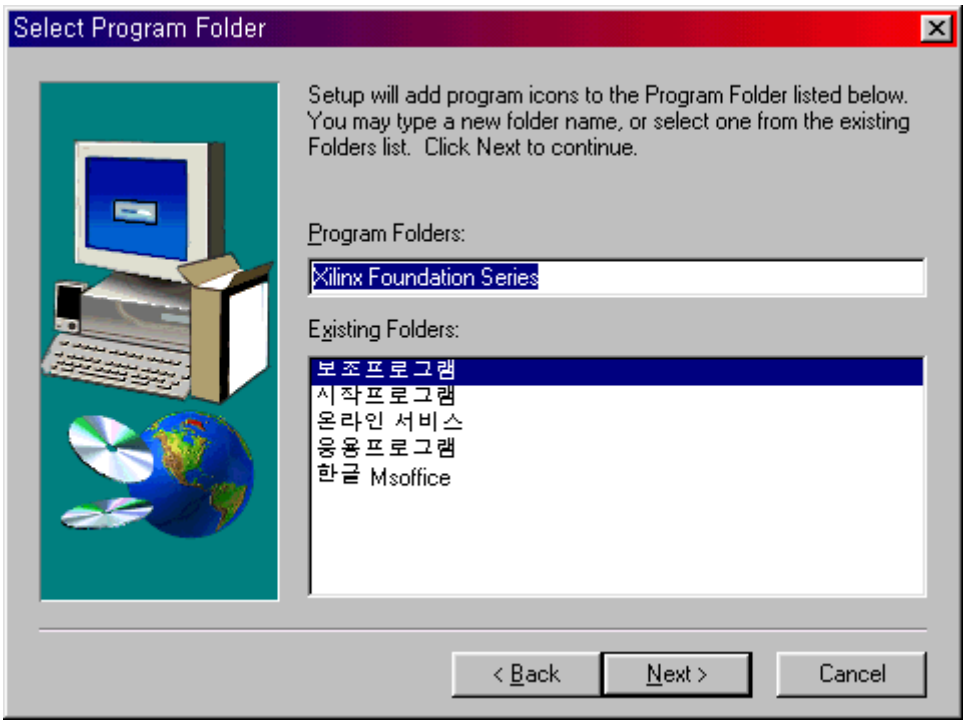


그림 9 Select Program Folder Window

그림 9 는 시작단추의 프로그램 그룹에 표시할 Program Folder 이름을 정의하는 곳입니다. 원하시는 이름을 넣어 주시길 바랍니다. Next Button 을 눌러 설치를 계속 합니다.

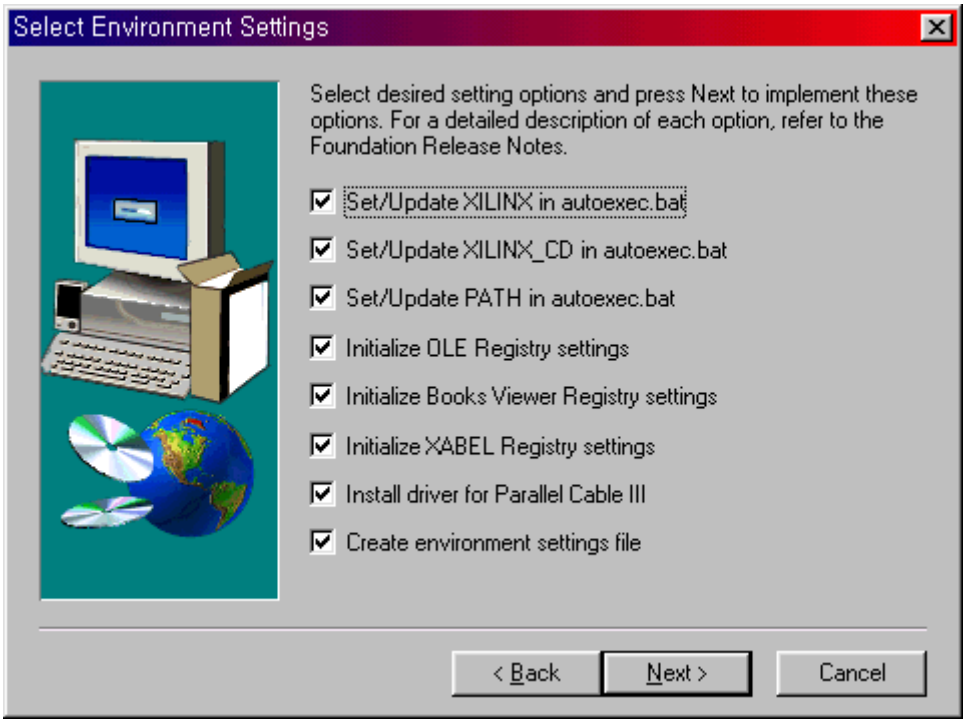


그림 10 Select Environment Settings Window

그림 10 은 시스템 환경변수를 설정하는 것입니다. 처음 설치하시는 사용자는 바로 Next Button 을 눌러 설치를 계속 진행하시고, 만일 컴퓨터에 Foundation Series Software 를 한번 설치한 적이 있으면 맨위의 세 항목은 선택하시지 마시길 바랍니다.

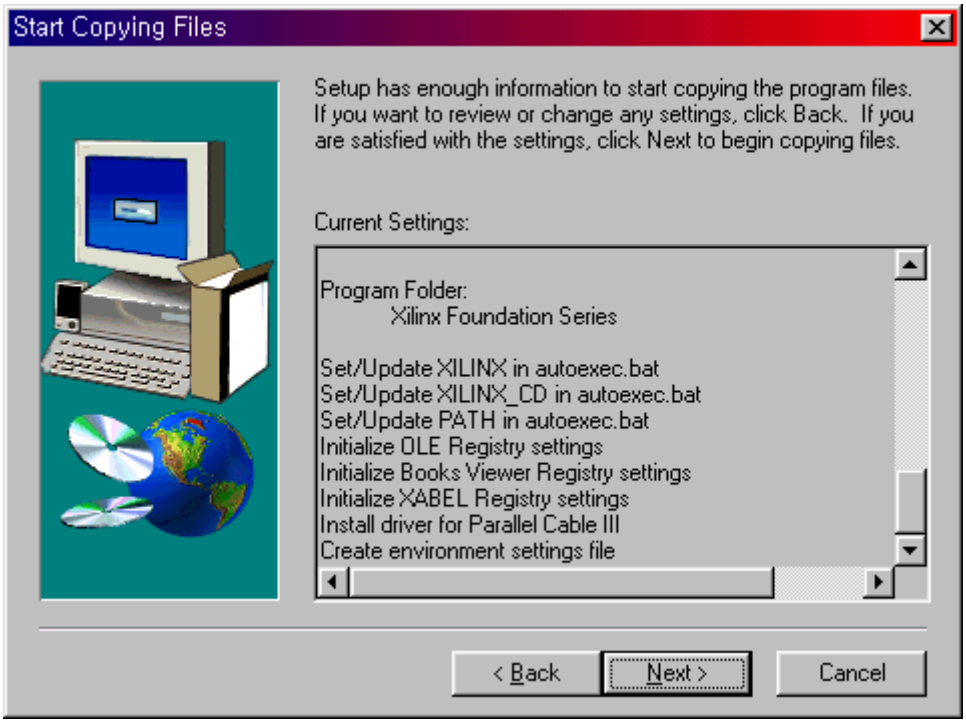


그림 11 Start Copying Files Window

그림 11 은 여러분이 선택한 항목들을 다시 한번 보여주고 설치를 시작해도 좋은지를 묻는 그림입니다. 선택한 사항이 문제가 없으면 Next Button 을 눌러 Foundation Series Software 가 설치되도록 하고 그렇치 않을시 Back Button 을 눌러 수정할 부분으로 이동후 수정하시길 바랍니다.

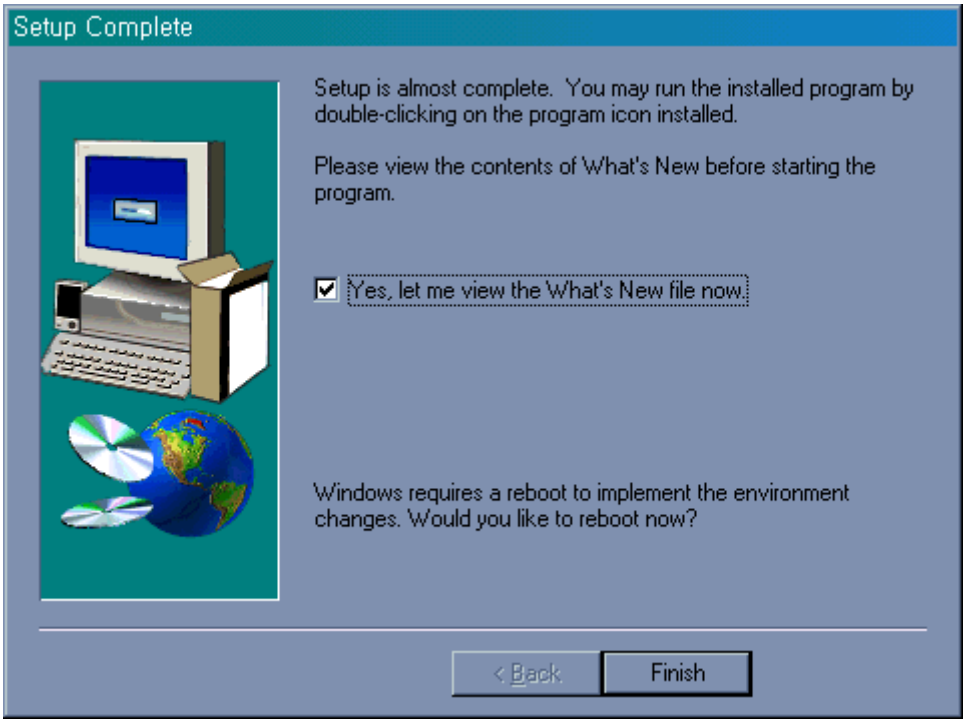


그림 12 Setup Complete Window

약간의 시간동안 설치프로그램이 파일복사 작업을 수행합니다. 모든 복사작업과 시스템 환경설정 작업이 끝나면 그림 12 가 나타나며 설치가 거의 완료되었음을 알려줍니다. 여기서 여러분의 현재 이 소프트웨어의 새로워진 점들을 보시길 원하면 체크박스를 선택하시길 바랍니다.

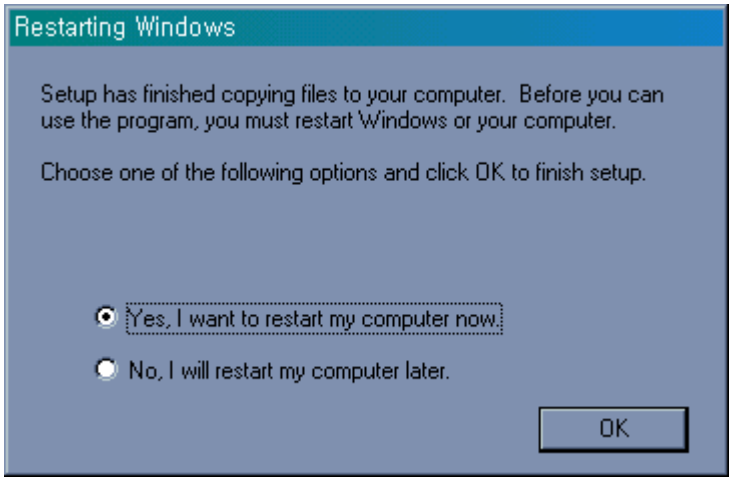


그림 13 Restarting Windows

그림 13 은 설치가 완전히 끝났고 변경된 시스템 환경변수를 사용하여 프로그램을 시작하려면, 시스템을 재시동하길 원하는 대화창입니다. 반드시 "Yes, I want to restart my computer now."를 선택

하여 재시작을 하시길 바랍니다.

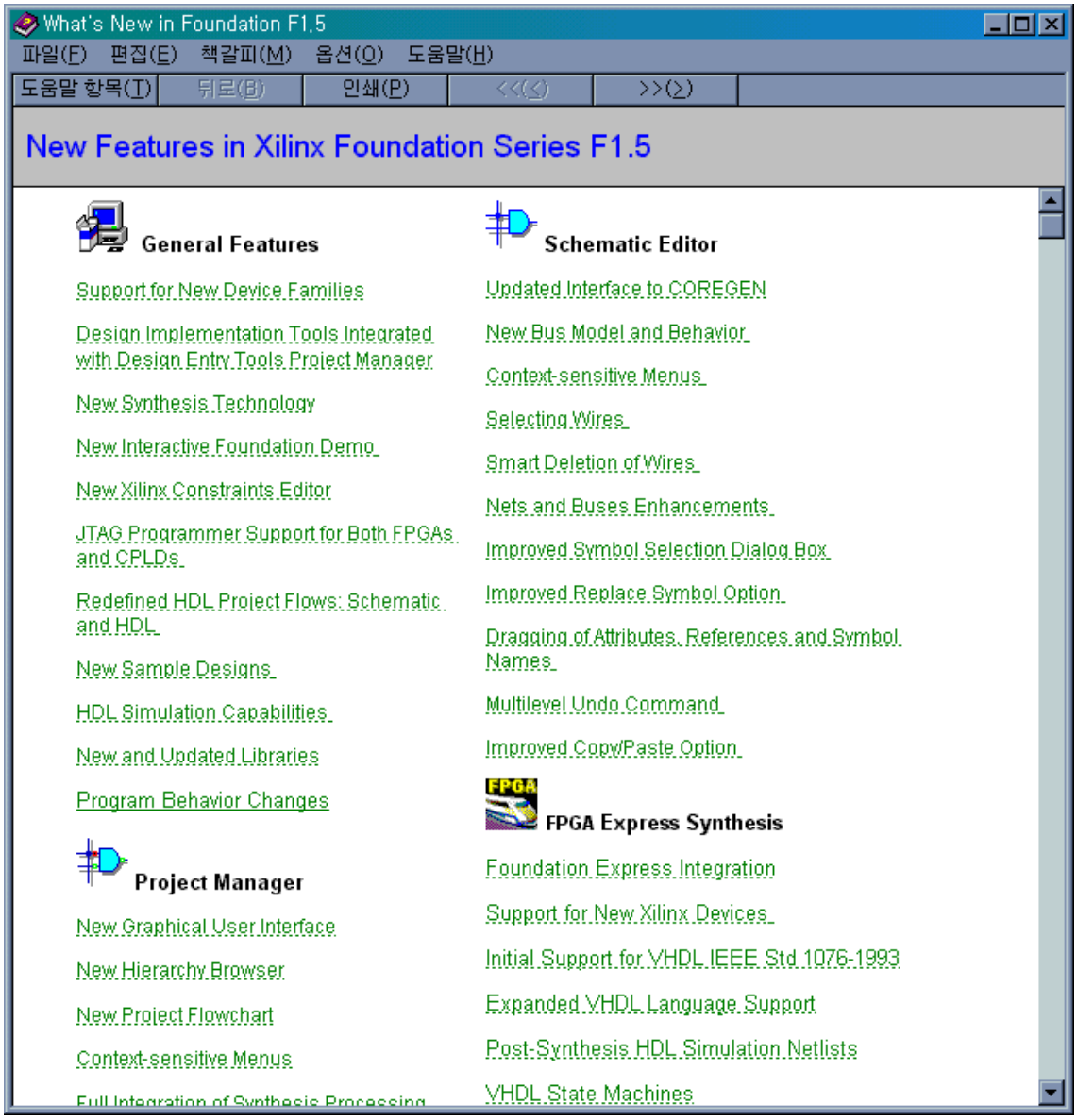


그림 14 What's News in Foundation F1.5 Window

그림 14 는 현재 소프트웨어의 새로워진 면들을 항목별로 정리한 화면입니다. 필요하신 항목을 선택하여 원하는 정보를 얻을 수 있습니다.

후기

이상으로 Xilinx Foundation 1.5 Software 설치에 관한 설명을 마칩니다. 설치 시 궁금한 점이나 의문사항이 있으시면 아래로 연락 주시길 바랍니다.

㈜현명전자
전화 : 3141-0147
팩스 : 3141-0149
응용 기술부
하 태욱 : twha@hmelec.co.kr
차 영근 : ykcha@hmelec.co.kr
양 창우 : cwyang@hmelec.co.kr
이 경득 : kdlee@hmelec.co.kr

1998 년 2 월 15 일 Revision 1…

XILINX®

BASIC TRAINING

F1.5i Day 1 Labs

Hyun Myung Electronics Co., Ltd.
Tel 02-3141-0147 / Fax 02-3141-0149
http://www.hmelec.co.kr

XS AT CE TP

Contents

소개 (Introduction)

본 실습교재에는 Foundation Series 1.5를 사용하여 XACT*step* M1을 지원하는 프로젝트를 만드는 방법이 설명됩니다. 설계 단계별로 중요한 부분들의 효율적인 설명을 위해서, 여기서는 실제로 하나의 프로젝트를 직접 만들어 갈 것입니다.

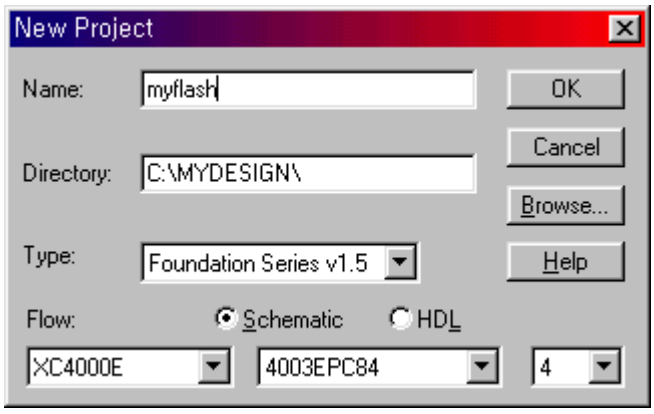
여기에서 주어지는 각 단계별 명령들을 잘 따라 한다면, 예제로 주어지는 FLASH design 이라고 정의한 프로젝트를 만들 수 있습니다.

예제 프로젝트에 주어지는 Chip의 Pin locations을 적용하시면 XILINX 4000 DEMO board에서 직접 설계된 회로를 검증하실 수 있습니다.

다음 장부터는 편의상 존칭은 생략합니다.

프로젝트 만들기 (Creating a New Project)

1. 바탕화면에 위치한 **Foundation Series** 아이콘을 더블 클릭하여 Project Manager를 시작한다.
2. **Project Manager**의 **File 메뉴** 중 **New Project…..**를 선택한다.
3. **New Project….** 창에서 Name항목에는 MYFLASH라고 기입하고, Directory는 사용자가 원하는 경로를 선택하면 된다. 여기서는 mydesign directory를 선택한다. Type항목은 Foundation Series v1.5을, Family항목은 XC4000, Part항목은 4003EPC84, 그리고 Speed항목은 4를 선택한다. Design Flow는 Schematic을 선택한다. 설정을 마친 후 **OK** 버튼을 클릭하면 새로운 프로젝트가 생성된다.



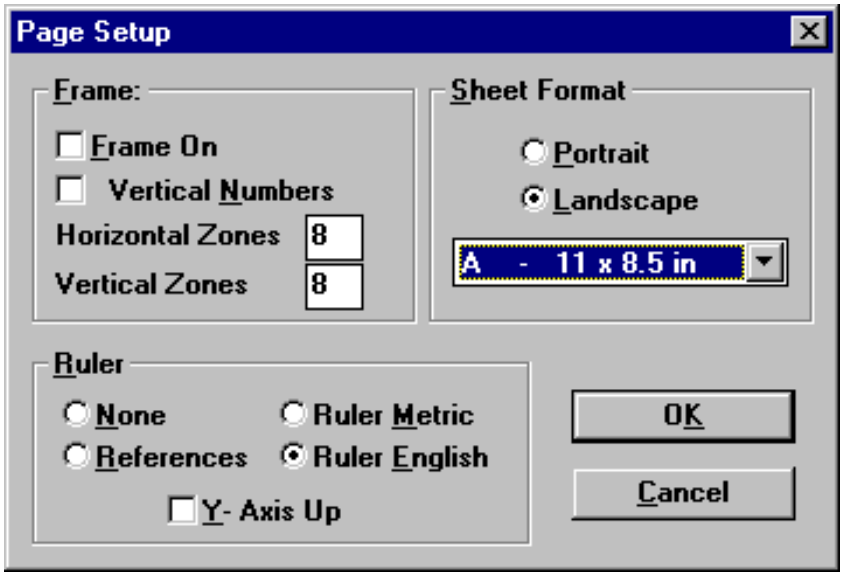
4. 새로운 프로젝트가 만들어 졌으면 아래그림에 나오는 것처럼 **Hierarchy Browser**에 라이브러리들이 생성이 될 것이다.

- **MYFLASH**는 프로젝트 라이브러리이다. 이 라이브러리는 현재 프로젝트에서 User가 작성하는 모든 Macro와 심볼들의 정보를 저장하는데 사용한다.
- **X4000E**는 XC4000E devices의 사용 시 필요한 시스템 라이브러리이다.
- **SIMPRIMS**는 Post - layout Netlists의 Simulation(Timing Simulation)에 필요한 시스템 라이브러리이다.

Schematic 시작하기 (Starting Schematic Editor)

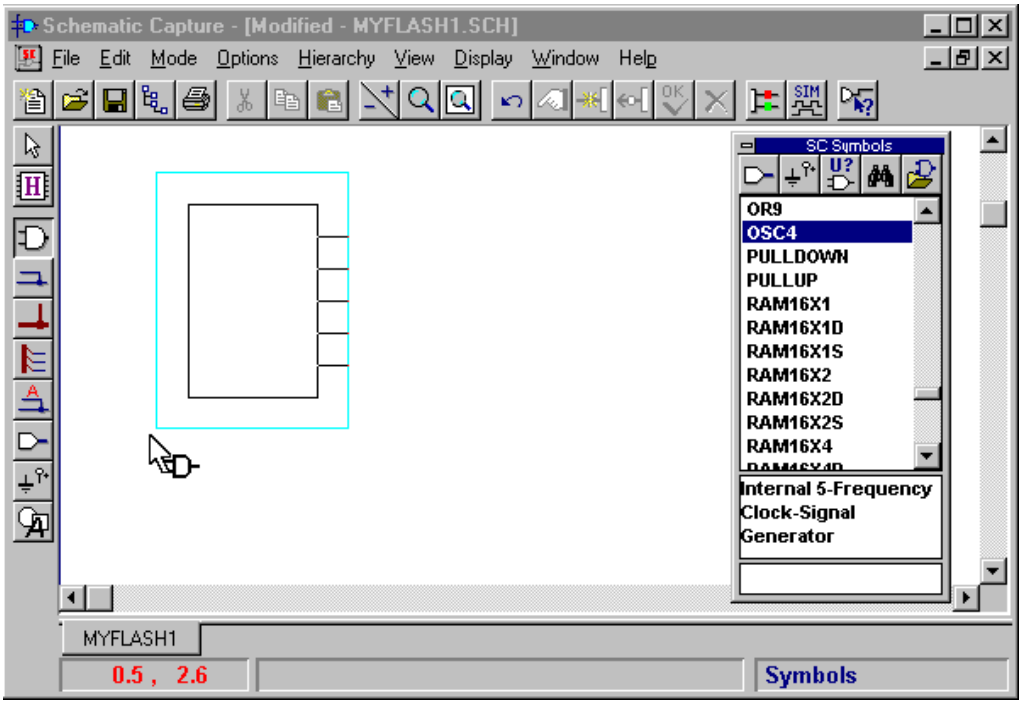
1. **Project Manager**상의 Flow diagram에 있는 Design Entry안의 and symbol icon 버튼을 클릭한다.

2. Schematic Editor에 MYFLASH1.SCH라는 이름으로 새로운 Schematic sheet가 열린다.
3. **File 메뉴**에 있는 **Page Setup**명령을 선택한다.
4. **Page Setup** 창이 화면에 나타나면, 그 중 **Sheet Format**부분을 Landscape, size A 11 x 8.5 in으로 선택한다.

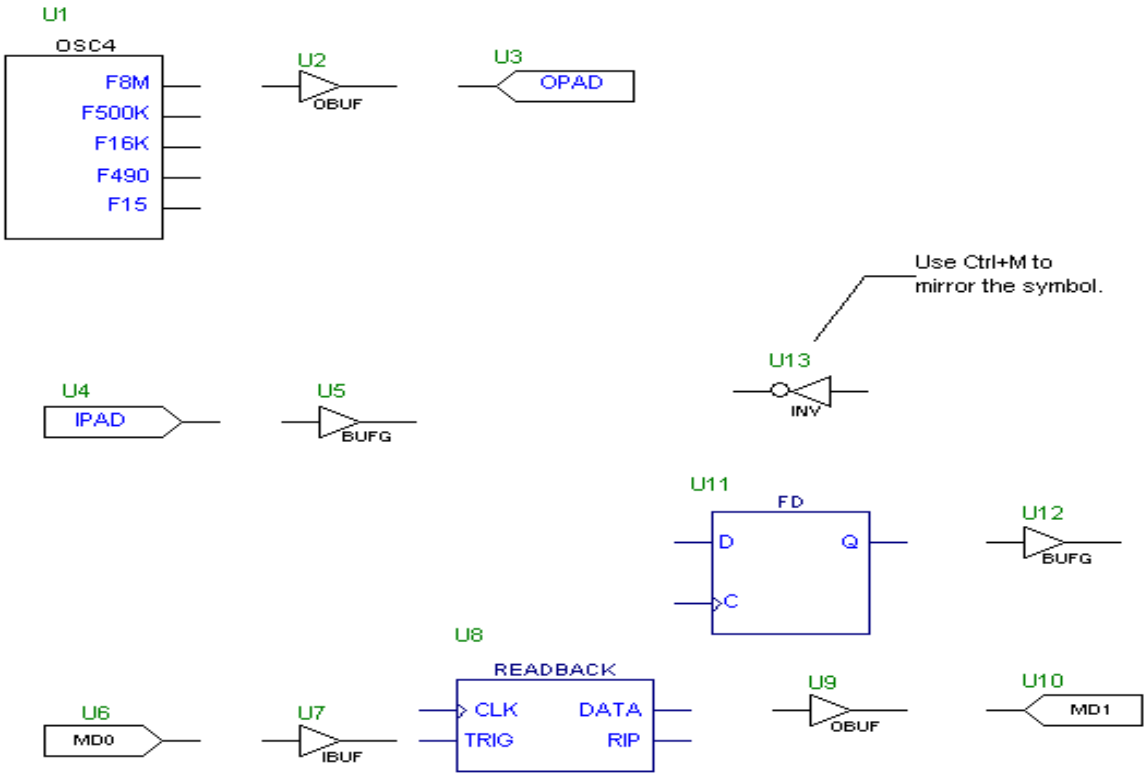


심볼 배치하기 (Placing Symbols)

1. 왼쪽에 위치한 Schematic 툴 바 중  버튼을 클릭한다.
2. 화면에 나타난 심볼 toolbox의 아래 입력란에 OSC4라고 입력한다.
3. 왼쪽으로 Cursor를 이동하면 Cursor를 따라서 선택한 심볼의 외형이 나타난다.



4. Mouse의 왼쪽버튼을 클릭하면 현 위치에 심볼이 배치된다.
5. 위의 순서를 반복하여 아래 그림처럼 각 심볼들을 배치한다:



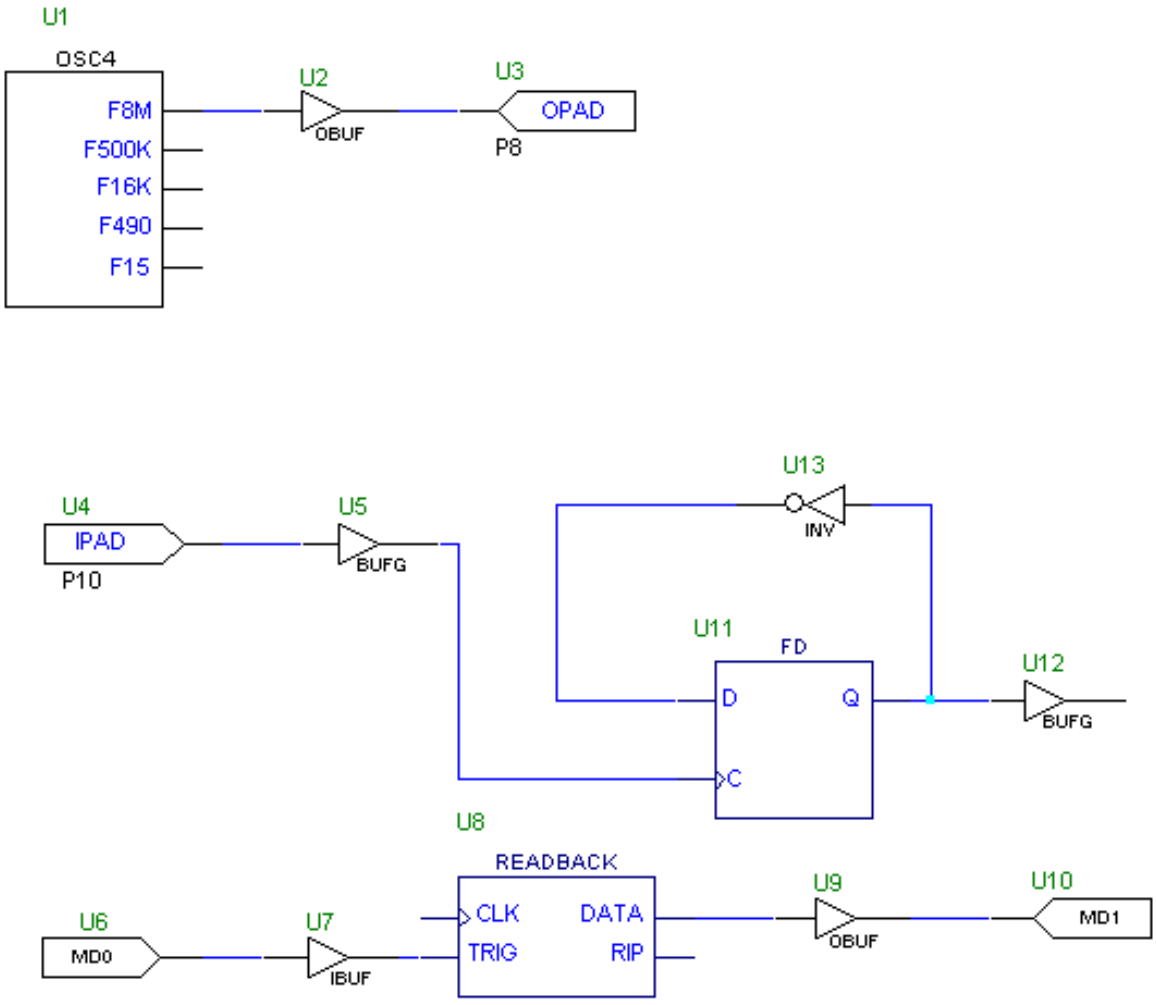
NOTES:

- 심볼을 배치하기 전에 **Ctrl+M** 키를 누르면 심볼의 좌우를 바꿀 수 있는데, 여기에서는 **INV** 심볼에다가 적용을 해보자. (**Ctrl+L** 키는 심볼의 회전에 사용한다.)
- 만약 심볼을 잘못 선택했다면, **Esc** 키를 눌러서 마우스 포인터에 붙어있는 잘못된 심볼을 없애버릴 수 있다.
- 임의의 심볼이 선택된 상태에서 포인터가 화면상의 경계에 가까이 다가가면 schematic page는 자동으로 스크롤이 된다.
- 전체 페이지를 보려면 **PgDn** 키를 누르고, 바로 전 상태로 돌아가려면 **PgUp** 키를 누른다. 이 키들은 키보드의 숫자키 부분에 있다.
- 잘못하여 틀린 심볼을 배치했다면, **Del** 키를 눌러서 삭제할 수 있다.
- 심볼 toolbox를 닫으려면, 왼쪽 편 툴 바에 있는  버튼을 누른다.
- Schematic의 특정부분을 확대하려면, 상단 툴 바에 있는  버튼을 누른다. 그런 다음 포인터로 확대할 영역을 사각형모양으로 드래그한다. 선택이 완료되면 눌러진 마우스 버튼에서 손을 뗀다.

- 전체화면으로 돌아가려면, 상단 툴 바에 있는  버튼을 누른다.
  현화면의 **Refresh**는, **F10** 키를 누른다.

선 연결하기 (Drawing Wires)

1. 선을 그리려면, 왼쪽 툴 바에 있는  버튼을 누른다.
2. 다음에는, 선이 시작될 핀 위에서 클릭을 하고 선이 종료될 다른 핀 위에서 클릭을 한다.
3. 기존에 그려진 선의 끝을 클릭하거나 다른 선을 시작하면, 자동적으로 junction(접합점)이 생성된다.
4. 위에서 설명한 순서대로 아래에 보이는 그림처럼 선들을 그린다.

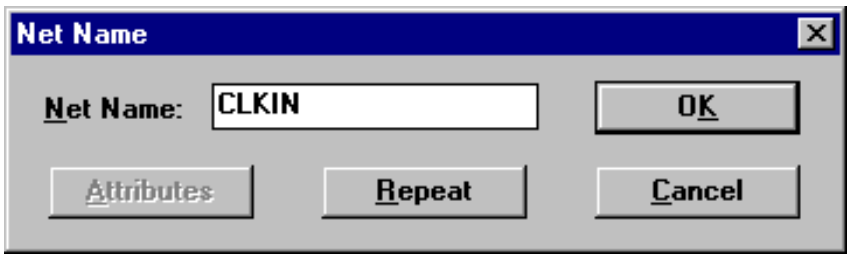


NOTES:

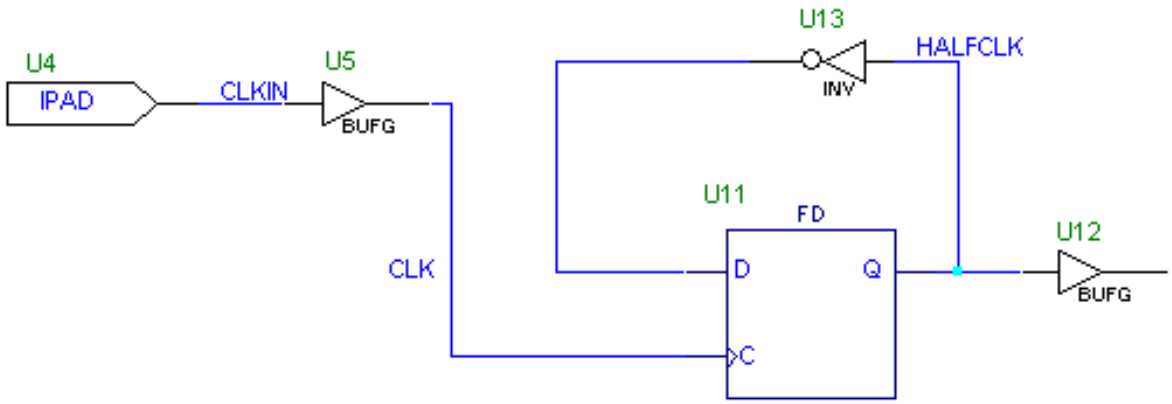
- 선 그리기 모드에서 빠져나가려면 **Esc** 키를 누른다.
- 선을 지우려면 **Esc** 키를 누르고, 삭제할 선을 클릭하고 **Del** 키를 누른다.
- Schematic을 저장하려면 상단 툴 바에 있는  버튼을 누른다.

네트 이름 주기 (Naming Wires)

1. 네트에 이름을 부여하려면, 왼쪽 툴 바에 있는  버튼을 누른다.



2. **Net Name** 창에 네트이름을 입력하고 **OK** 버튼을 누른다.
3. 다음으로, 네트이름이 위치할 네트로 커서를 이동하고 클릭한다.
4. 녹색으로 네트이름이 표시된다면 이름이 선에 대응되지 않은 것이다.
5. 다음에 나오는 선들에 네트이름을 부여한다:
   - CLKIN
   - CLK
   - HALFCLK

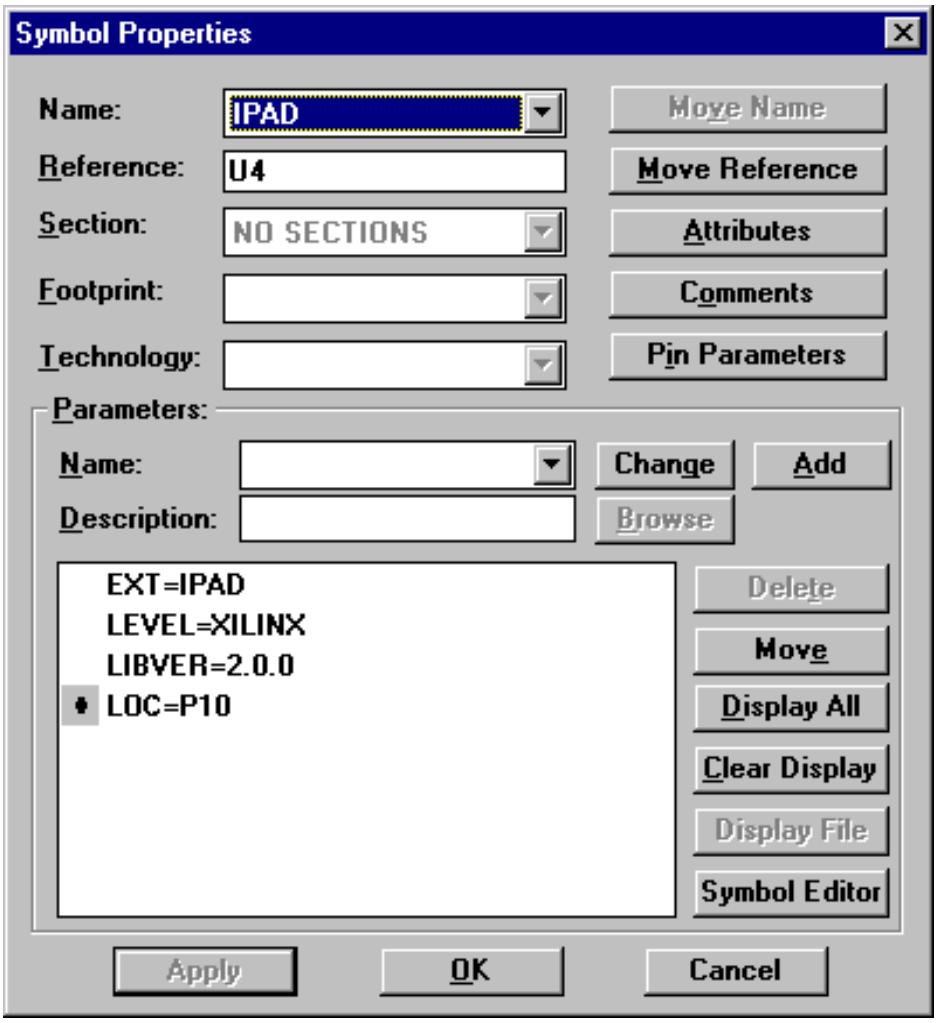


핀 위치 지정하기 (Adding Pin Locations)

Xilinx Demonstration Board에 이 회로를 적용하고 싶다면, Schematic 상에 정의할 수 있는 핀 번호 지정이 필요하다. 핀 번호를 정의하려면 다음의 순서를 따른다:

1. 선택모드로 이동하기위해서 **Esc** 키를 누른다.
2. U4로 이름이 붙여진 IPAD 심볼을 더블 클릭한다.
3. Name:란에 *LOC*를 Description:란에 *P10* 이라고 입력한다. (Parameters section).
4. 아래쪽 목록에 추가 시키기 위해 **Add** 버튼을 누른다.

NOTE: Your reference designators may be different on your screen or may be invisible. Please modify the appropriate component.



5. **OK** 버튼을 누른다.
6. IPAD 심볼 아래에 P10 parameter가 보일 것이다.
7. 같은 순서로 OPAD 심볼 U3에 LOC=P8 parameter를 추가한다.

Schematic의 변경 (Schematic Changes)

모든 회로의 변경은 select mode에서 실행된다. 이mode는 왼쪽 툴 바에 있는  버튼이 눌러져 있을 때 활성화된다. 이 select mode로 빨리 되돌아 가는 방법은 **Esc** 키를 누르면 된다.

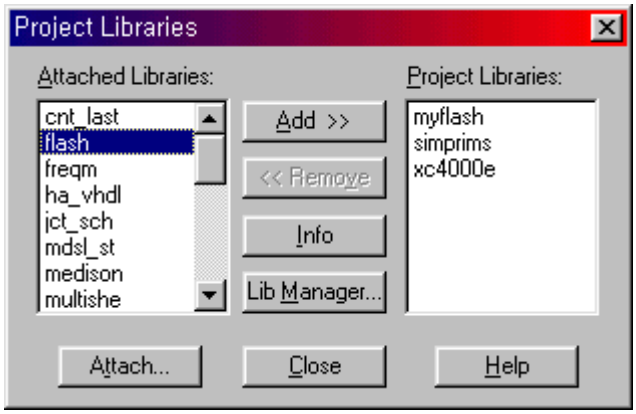
다음의 항목들은 편집 작업의 수행 시 공통적으로 쓰이는 방법들이다:

- 심볼을 이동하려면, 해당 심볼을 클릭하고 원하는 위치까지 드래그하면 된다.
- 회로의 일 부분을 이동할 때는 원하는 부분의 주위를 box로 감싸고, 선택된 심볼 중에 하나를 클릭하여 원하는 위치로 드래그하면 선택된 모든 심볼들이 모두 이동한다.
- 선택된 그룹에서 임의의 개체를 추가 삭제하려면 **Shift** 키를 누르고 그 개체를 클릭한다.
- 심볼에 연결된 선들을 그대로 유지한 체 심볼만을 지우려면, 지울 심볼을 선택하고 상단 툴 바에 있는  버튼을 누른다. 그리고 나서 **Del** 키를 누른다. 심볼이 없어진 선은 Hanging wire로 된다.
- Hanging wire에 심볼의 핀을 연결하려면, schematic 툴바에 있는  버튼을 누르거나, 핀을 선의 끝으로 드래그한 후 마우스로 버튼을 다시 눌러주면 된다.

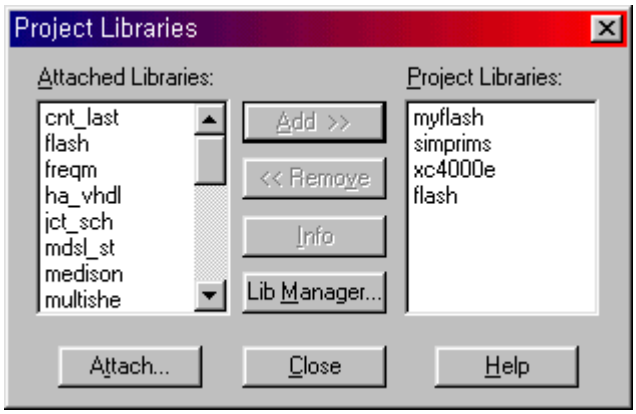
기존 Project의 Library를 불러쓰기

이번 장에서는 기존에 만들어 둔, 또는 작업이 끝난 project을 이용하여 현재 project에서 필요한 component을 선택하여 사용하는 방법을 알아 보기로 한다. 크게 두 가지 방법이 있으며 여기서는 기존의 project을 library화 하여 사용하는 attaching the project를 설명한다. 다른 한 가지는 library manager를 사용하여 필요한 component를 현재의 project에 copy하여 사용하는 방법이다..

1. Project Manager 의 상위 툴 바 버튼에서 Project libraries icon button  을 누른다.
2. 아래 대화상자가 나타나면 대화 상자의 내용을 아래와 같이 설정한다.



3. Attached Libraries는 기존에 작업이 된 project를 의미하며 Project Libraries는 현재 project가 참조하고 있는 library를 뜻하고 있다. flash를 선택한 후 Add >> button을 누른다.
4. 아래 그림과 같이 오른쪽에 flash가 첨가되면 성공적으로 flash project가 attach 된 것이다.



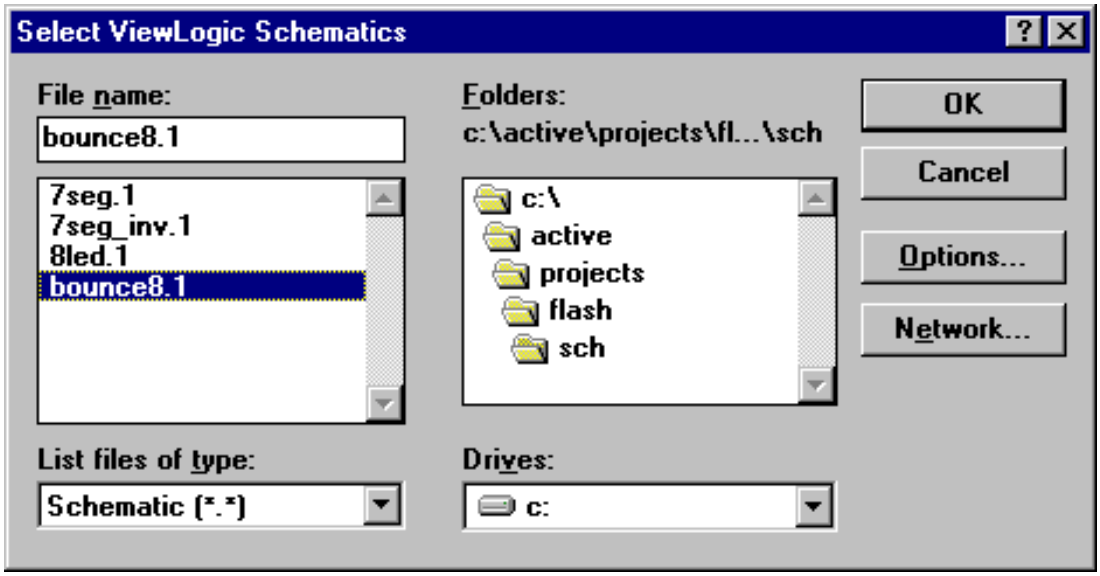
5. Close button을 눌러 Project Libraries 대화 창을 빠져 나온다.

Viewlogic 데이터 받아들이기 (Importing a Viewlogic Macro Schematic)

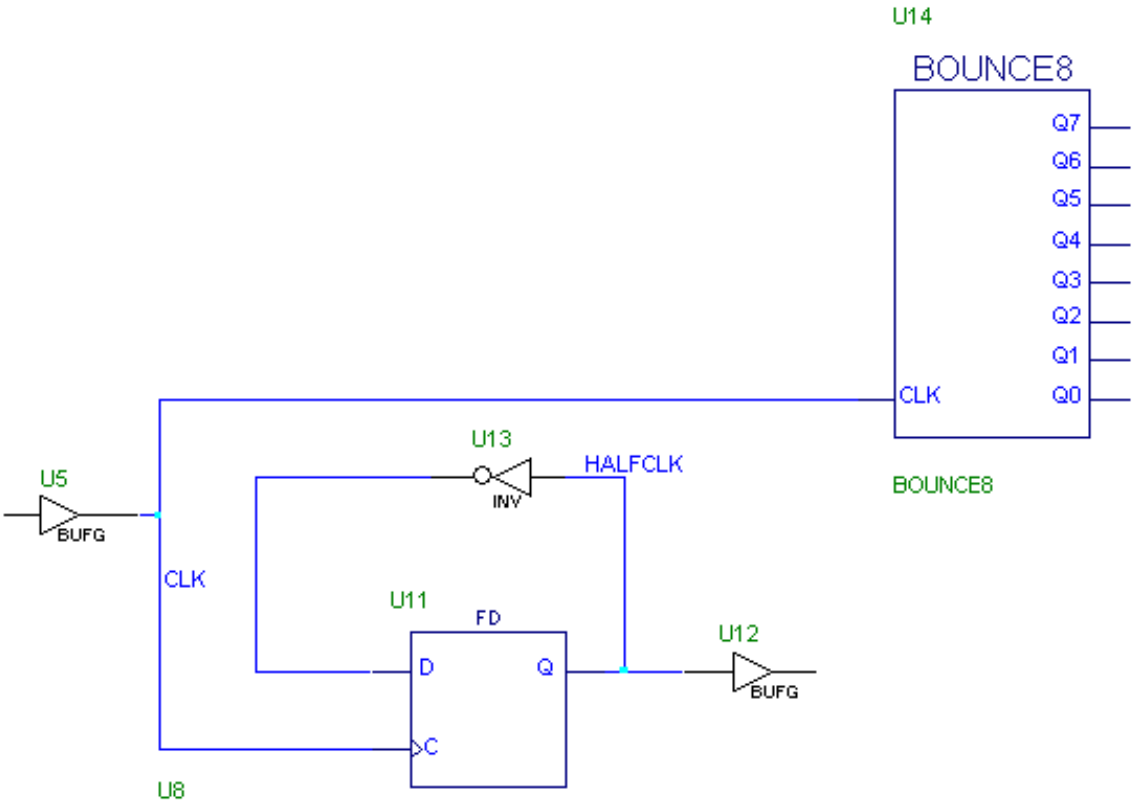
이 예제는 선택 사항이므로 Foundation Series 1.5로 실습을 할 때는 생략하기로 하며, 필요한 component를 c:\fndtn\active\projects\flash에 있는 것을 copy하여 사용한다.

Viewlogic schematic을 Foundation Schematic Capture로 Import하는 예제로, 현 회로에 Viewlogic schematic macro중 하나를 받아들여 본다.

1. **File** 메뉴에 있는 **Import Viewlogic Schematic** 명령을 선택한다.
2. **List Files of Type** 필드에서 Schematic (*.*)을 선택하고 (**Select Viewlogic Schematics**창 내에서), **File Name** 필드에서 C:\ACTIVE\PROJECTS\FLASH\SCH\BOUNCE8.1 schematic을 선택한다.



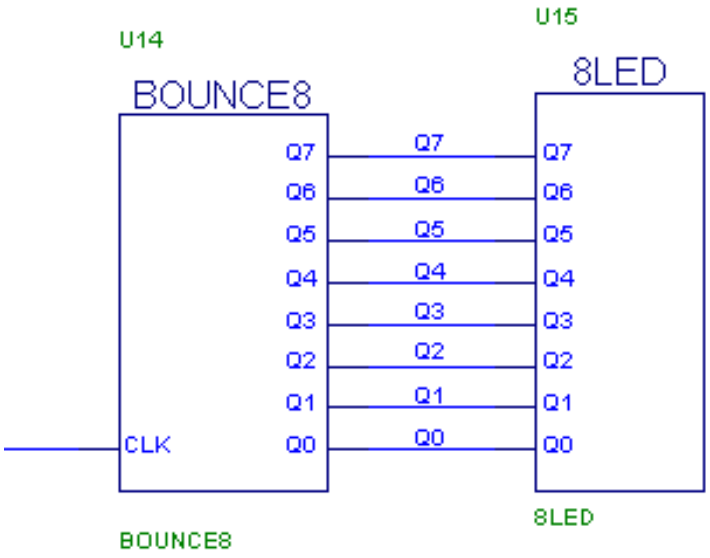
3. **OK** 버튼을 누르면 Viewlogic schematic을 받아들이기 시작한다.
4. BOUNCE8 schematic macro가 MYFLASH 프로젝트 라이브러리에 추가가 될 것이다.
5. 결과리포트를 보길 원하냐는 내용의 대화상자가 나타나면, **NO** 버튼을 선택한다.
6. **File** 메뉴의 **Update Libraries**를 실행하면 받아들여진 심볼이 라이브러리에 보태어진 것을 확인할 수 있다.
7. 받아들인 macro를 다음에 보이는 회로처럼 배치한다.



8. 동일한 방법으로 같은 디렉토리에 있는 7SEG.1와 8LED.1을 import한다. **Ctrl** 키를 사용하면 한번에 여러 개의 schematic을 선택할 수 있다.

네트이름 반복하기 (Repeating Net Names)

1. 아래의 그림처럼 8LED 심볼을 배치하고 선을 연결한다.

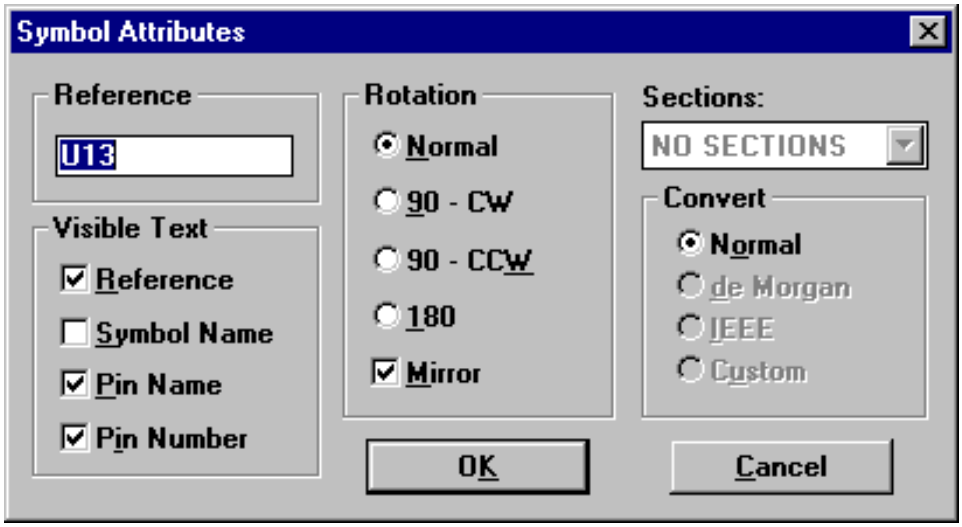


2. 네트이름을 추가하려면, 왼쪽 툴 바의  버튼을 선택하고 Q0 를 입력한다.

3. 동일한 성격의 선 이름을 빠르게 주기위해 **Repeat** 버튼을 누른다.
4. 프로그램은 반복모드로 설정이 된다. 매번 선을 클릭할 때마다 새로운 이름이 하나씩 증가하면서 배치가 된다. 이런 방법으로 8개의 선 이름을 추가한다. (이때 주의할 사항으로서 선 위를 정확히 클릭하여 네트이름과 네트의 색이 동일하도록 해주어야 한다. 만약 네트와 네트이름의 색이 다를 경우 네트이름은 라벨로 처리가 되었슴을 의미한다.)

NOTE: 심볼명을 나타내기 위해 **Esc**를 누르고, 심볼을 더블 클릭한다. 그리고 **Symbol Properties**

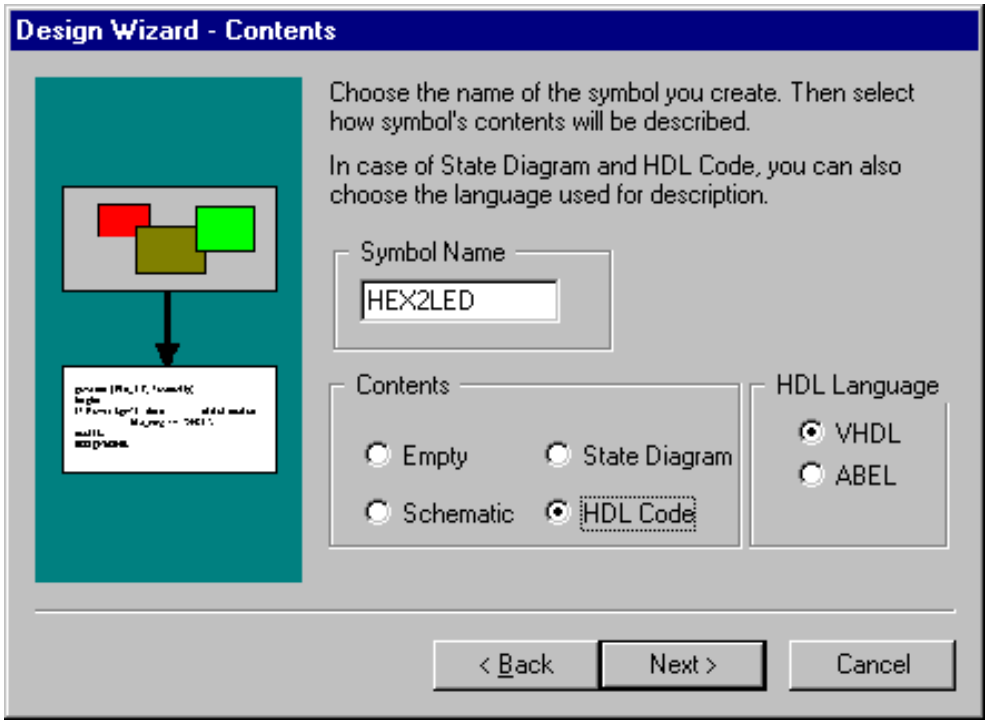
창의 **Attributes** 버튼을 선택한다. **Symbol Attributes**창 내 **Visible Text**부분의 **Symbol Name**을 선택하고 **OK**를 누른다.
다음으로 Symbol Properties 창에 있는 OK 버튼을 누르면 심볼명이 표시된다.



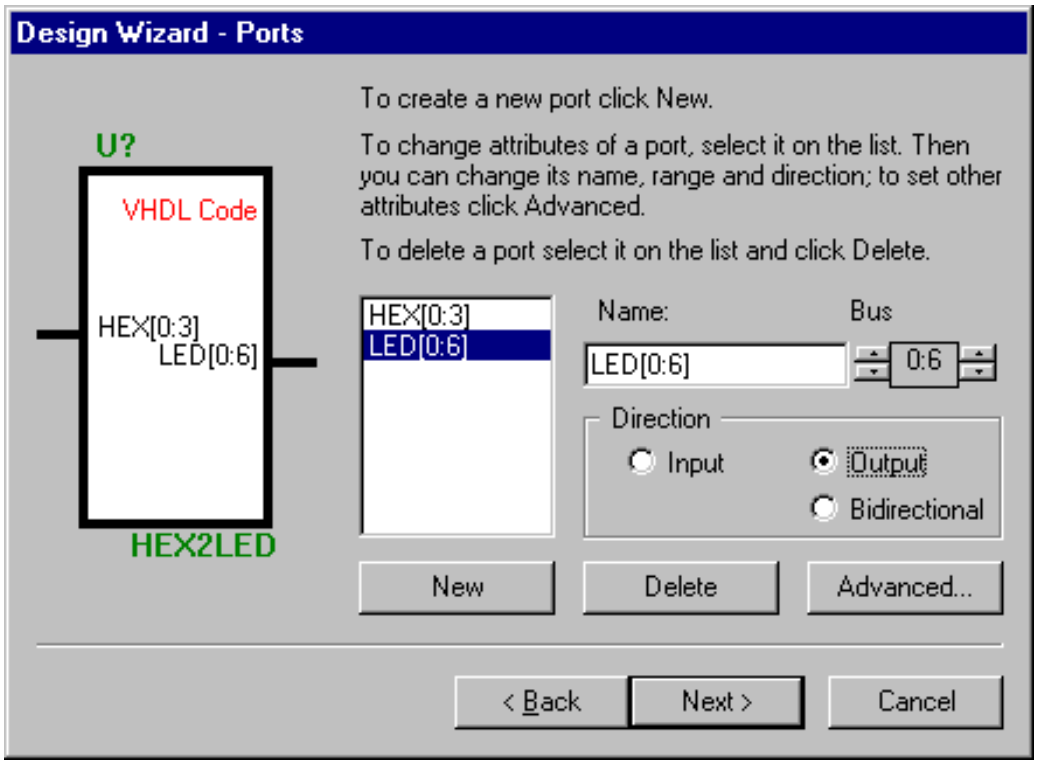
새 심볼 만들기 (Creating a New Symbol)

VHDL을 사용하여 만들 예제로서 HEX Data 값을 7Segment LED 출력 값으로 변환하는 Decoder를 만들 것이다.

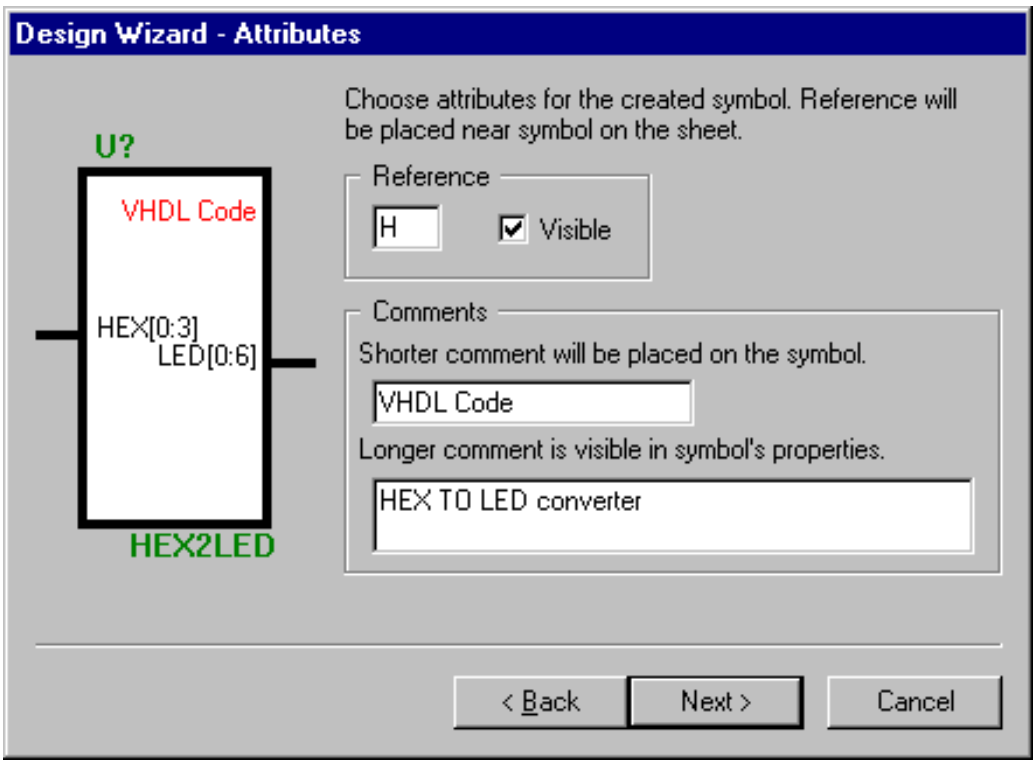
1. **File** 메뉴의 **New Sheet**명령을 선택하여 새로운 schematic을 생성한다.
2. **Hierarchy** 메뉴의 **New Symbol Wizard**을 선택한다.
3. **Design Wizard** 창의 환영 메시지가 나타나면 **Next>**를 클릭한다. 그러면 **Design Wizard - Contents** 창이 나타날 것이다.



4. **Symbol Name** box에 HEX2LED라고 입력한다.
5. **Contents** 필드에서 다음의 내용을 선택한다:
   - Contents: HDL Code
   - HDL Language: VHDL.
6. **Next>** 버튼을 누르면, **Design Wizard - Ports** 창이 나타날 것이다.



7. **Design Wizard - Ports** 창에서
   - **New** 버튼을 누른다.
   - **Name**: 필드에 HEX[0:3]을 입력한다.
   - **Direction** 필드에서 Input을 선택한다.
   - 다시 **New** 버튼을 누른다.
   - **Name**: 필드에 LED[0:6]을 입력한다.
   - **Direction** 필드에서 Output을 선택한다.
   - **Next>** 버튼을 누르면, **Design Wizard - Attributes** 창이 나타난다.

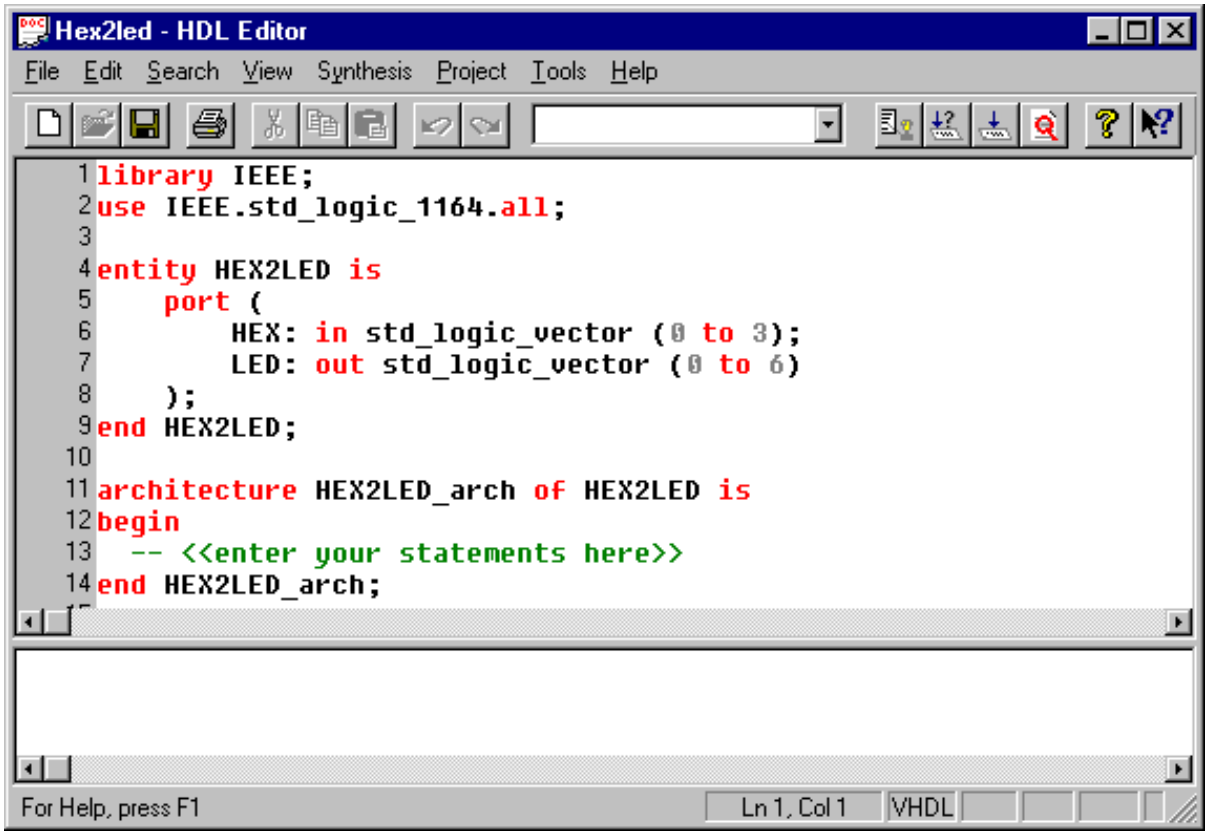


8. **Design Wizard - Attributes** 창에서 Longer comment~ 란에 "HEX to LED converter"라고 입력한다.
9. 계속하기 위해 **Next**을 누르면 심볼이 생성된다. 이 새 심볼은 프로젝트 라이브러리에 배치된다.
10. **Finish**를 누르면 심볼이 생성이 되는데, 만들어진 Macro는 프로젝트 라이브러리에 추가된다.
11. HEX2LED 심볼을 새 도면에 배치한다 (MYFLASH2.SCH).

VHDL MACRO 편집하기 (Editing a VHDL Macro)

1. 왼쪽 툴 바에서  버튼을 선택한다.
2. Schematic 상에서 HEX2LED 심볼을 두 번 클릭 한다.

3. HDL Editor가 HEX2LED 매크로의 port 정의를 보여 주는 아래 그림과 같은 윈도우를 열어 준다.



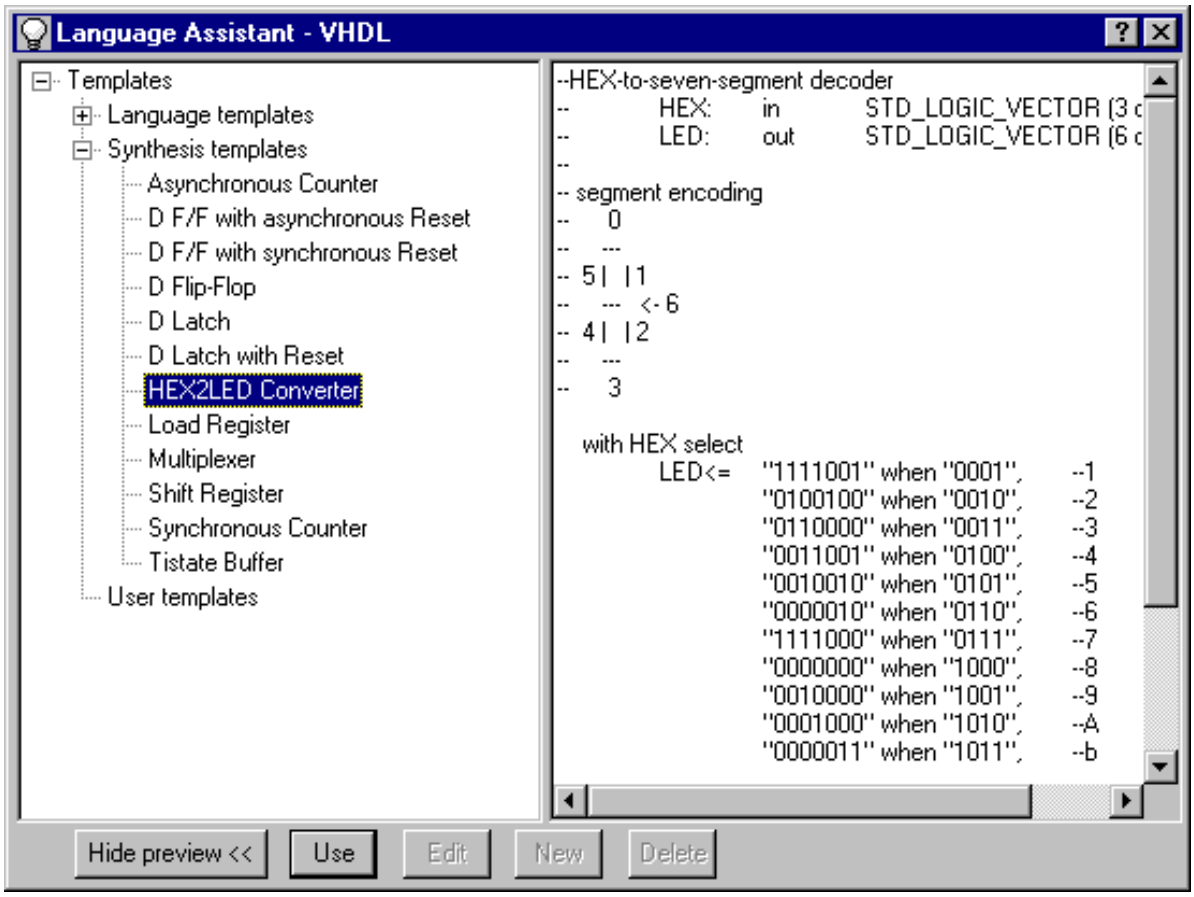
4. **entity** section위에 있는 모든 line들을 지운다.

5. **port** section 안에 있는 선언들을 아래와 같이 변경한다.
   HEX: in BIT_VECTOR (3 downto 0);
   LED: out BIT_VECTOR (6 downto 0)
   bit 0 는 LSB(Least Significant Bit)를 의미한다.

6. <<enter your statements here>> line을 지운다.

7. Architecture section의 begin 바로 아래 라인에 커서를 위치한다. 그러고 나서 **Tools** 메뉴에서 **Language Assistant**를 선택한다.

8. **Synthesis Templates**를 선택하고 **Select Template** 목록에서 HEX2LED Converter를 찾는다.
9. **Show preview>>** 버튼을 누른다. template의 문장들이 **Language Assistant** 창의 오른쪽 패널에 보여질 것이다.



10. Template을 복사하기 위해 **Use** 버튼을 누른다.

11. 아래 보여진 code를 만들기 위해 필요한 변경 작업들을 수행하라.

Final Version of HEX 2LED Macro Code:

```
entity HEX2LED is
      port (
            HEX: in BIT_VECTOR (3 downto 0);
            LED: out BIT_VECTOR (6 downto 0)
            ) ;
begin
end HEX2LED ;

architecture HEX2LED_arch of HEX2LED is
begin
--HEX-to-seven-segment decoder
--     HEX:       in    STD_LOGIC_VECTOR (3 downto 0);
--     LED:       out   STD_LOGIC_VECTOR (6 downto 0);
--
-- segment encoding
--          0
--         ---
--    5 |     | 1
--         ---     <- 6
--    4 |     | 2
--         ---
--          3

with HEX select
      LED<=    "1111001" when "0001", --1
               "0100100" when "0010", --2
               "0110000" when "0011", --3
               "0011001" when "0100", --4
               "0010010" when "0101", --5
               "0000010" when "0110", --6
               "1111000" when "0111", --7
               "0000000" when "1000", --8
               "0010000" when "1001", --9
               "0001000" when "1010", --A
               "0000011" when "1011", --b
               "1000110" when "1100", --C
               "0100001" when "1101", --d
               "0000110" when "1110", --E
               "0001110" when "1111", --F
               "1000000" when others;  --0

end HEX2LED_arch;
```

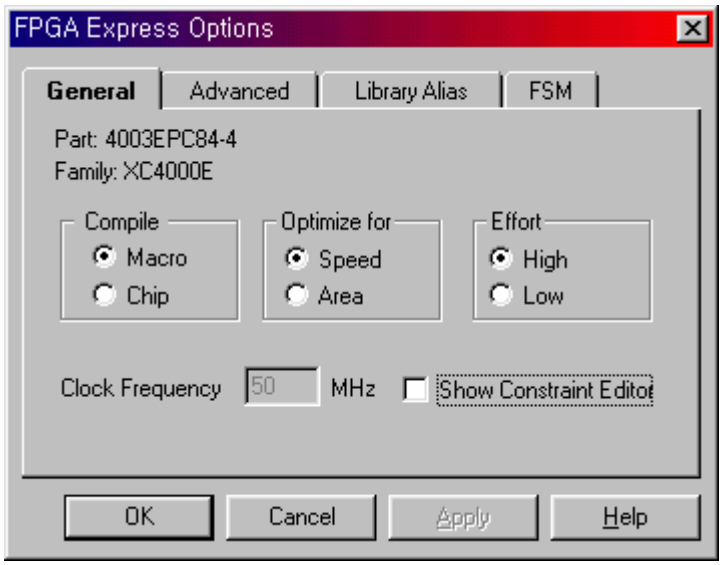
VHDL 오류 찾기 (Finding VHDL Errors)

VHDL 오류 찾기 기능을 보여주기 위해 하나의 오류를 만들어 본다.

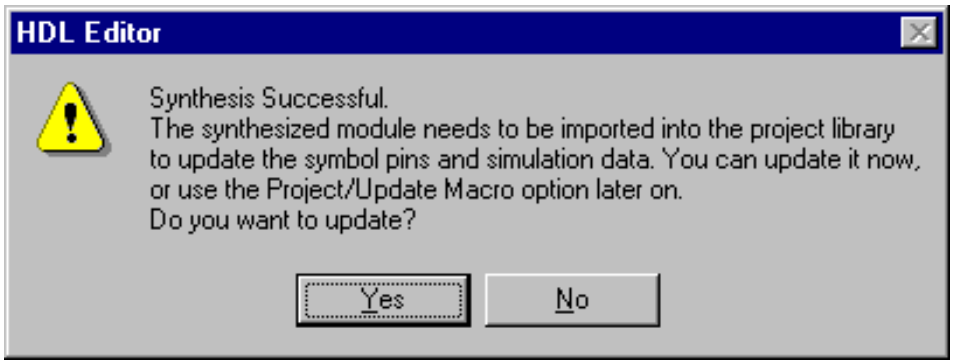
1. port declaration section 안에 있는 ';' 문자를 지운다.
2. **Synthesis** 메뉴에서 **Check Syntax** 명령을 선택한다.
3. **HDL Editor** 창 아래 안에 있는 오류 메시지를 볼 수 있을 것이다. Source code의 왼쪽 편에 붉은 화살표가 오류가 어디서 발생했는지 알 수 있는 라인을 지시한다.
4. 여러 개의 오류를 가지고 있다면 아마도 VHDL code을 편집하는 동안 실수를 했을 것이다. 앞 페이지의 VHDL Source code를 살펴 보고 이상 없이 수정한다.

회로합성 하기 (Running Synthesis)

1. **Synthesis** 메뉴에서 **Options**를 선택한다.



2. **Effort Level Low** option을 선택한다.
3. **OK** 버튼을 한번 눌러 준다.
4. **Synthesis** 메뉴에서 **Synthesize**를 선택한다. 이것은 Synopsys 사에서 제공하는 FPGA Express Synthesis compiler를 기동시키고 선택된 VHDL code로부터 XNF netlist를 생성한다.
5. Synthesis가 다 완료되면, HDL Editor는 매크로를 update 할 것인지를 물어 본다.

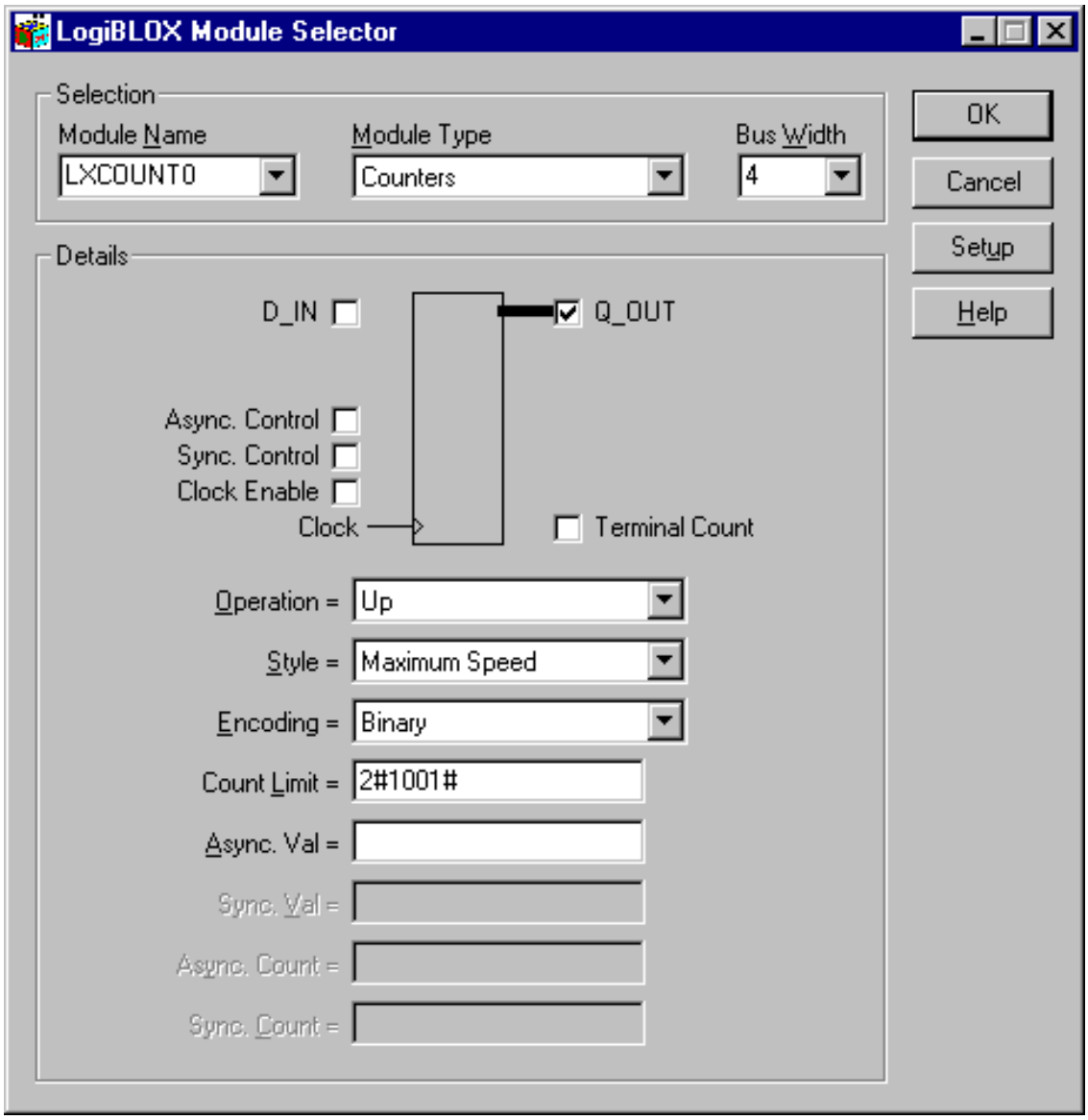


6. **Yes**를 누른다. 이것은 프로젝트 working 라이브러리 안의 매크로 netlist와 심볼을 update 시킬 것이다.
7. HDL Editor가 그 심볼이 성공적으로 Update 되었다는 메시지를 보여주면 **O.K** 버튼을 눌러 준다.
8. HDL Editor를 종료한다.

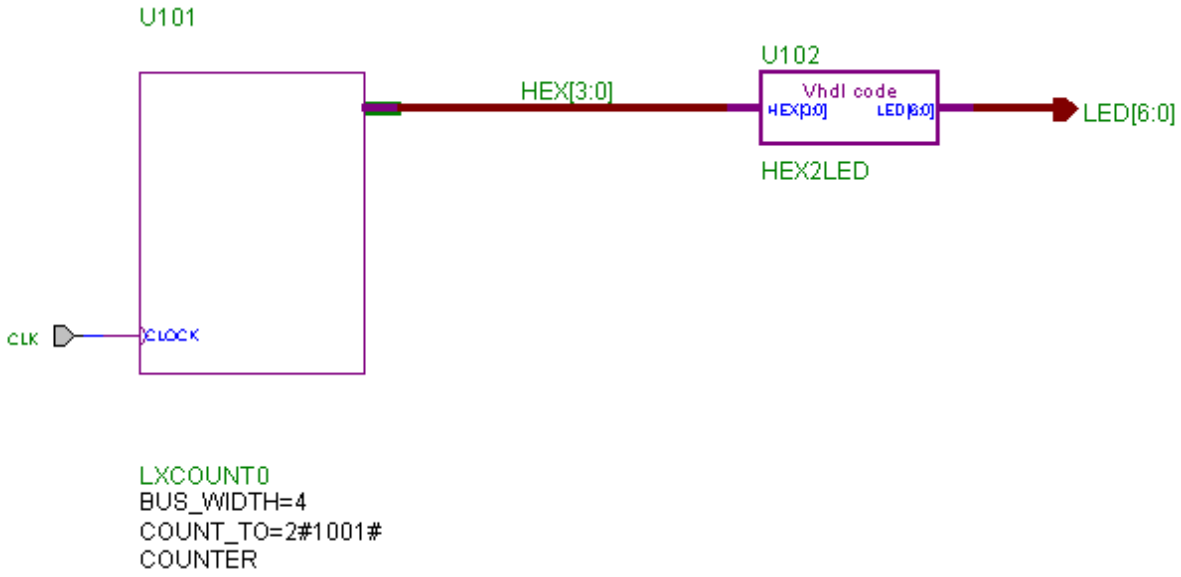
Schematic Macro 만들기 (Creating a Schematic Macro)

HEX2LED macro을 Update한 후에, 카운터의 나머지 부분도 완성이 될 것이다.

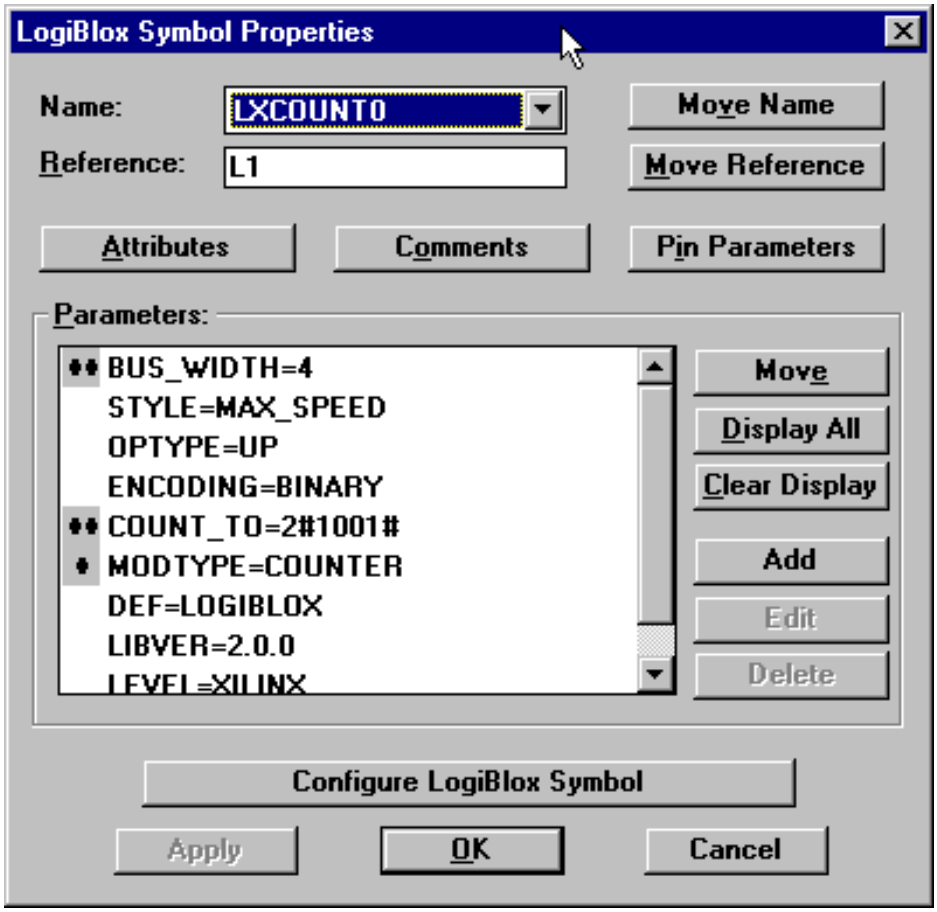
1. **Options** 메뉴에서 **LogiBLOX….**를 선택한다. 이것은 **LogiBLOX Module Selector** dialog를 보여 주는 LBGUI program를 호출 한다.



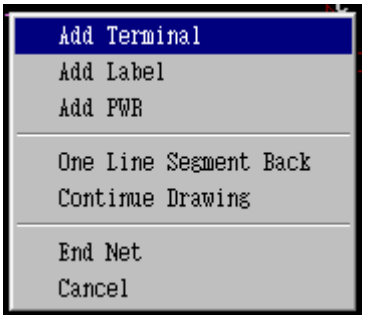
2. Select Counter from the **Module Type** box에서 Counter를 선택 후 **Bus Width** box에 4를 넣는다.
3. **Module Name** box에 LXCOUNT0 라고 넣는다.
4. **Details** group 안에서 **D_IN** 과 **Clock Enable** boxes를 선택하지 않는다. **Operation** 안에 Up을 선택, **Encoding** 안에는 Binary, 그리고 **Count Limit** field 에는 2#1001#를 넣는다.
5. 프로젝트 라이브러리 안에 LXCOUNT0 logiblox 심볼을 넣기 위하여 **OK**를 누른다.
6. 성공적으로 심볼이 만들어 진 것을 보고하는 메시지를 확인한다.
7. Schematic상에 LXCOUNT0 component을 위치 시킨다.



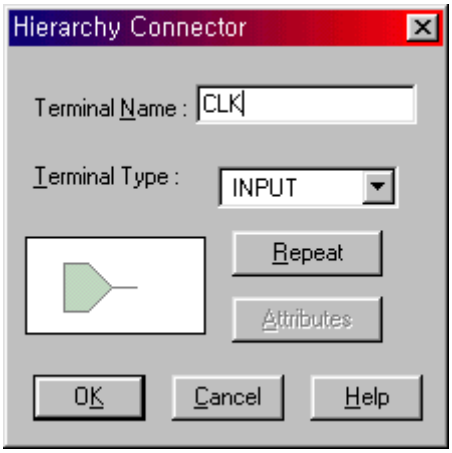
8. **LogiBlox Symbol Properties** 대화 상자를 열기 위해 LXCOUNT0 logiblox 심볼을 두 번 연속 누른다. Configure LogiBlox Symbol 버튼은 대화 상자 안에서 logiblox symbol parameters를 직접 변경할 수 있도록 하기 위해 제공되고 있음을 유의한다. parameter 변경은 component의 이름을 변경하지 않는 한 schematic component 와 라이브러리 module 둘 다 영향을 미친다. 아무 것도 변경하지 않고 대화 상자를 닫는다.



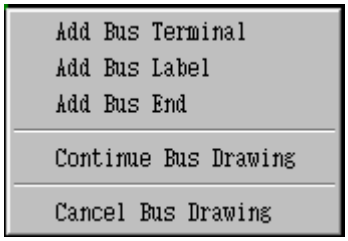
9. 왼쪽편의 매크로의 Clock 핀으로부터 wire 그리기를 시작한다.
10. Wire 그리기를 끝내기 위해 오른쪽 마우스 버튼을 누른다.



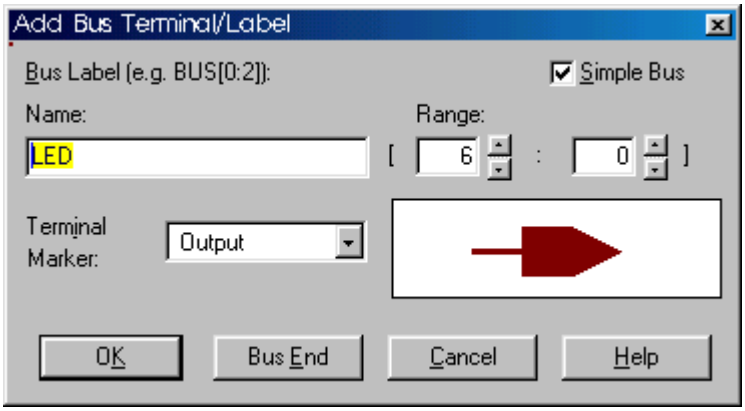
11. 위의 대화 상자에서 Add Terminal을 선택한다.
12. 아래 그림에서 보는 것과 같이 CLK 이름을 넣는다.



13. 왼쪽 툴 바상에서  버튼을 사용하여, HEX2LED macro의 LED[6:0] 핀으로 부터 버스를 오른쪽으로 그린다.
14. 버스 그리기를 멈추기 위해 마우스 오른쪽 버튼을 누른다.



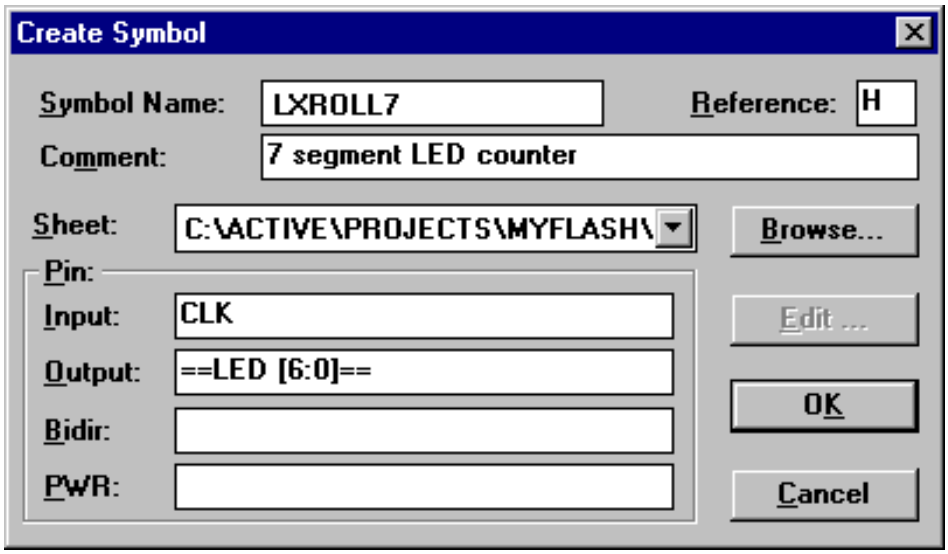
15. 아래 그림과 같이 Bus I/O terminal name을 LED[6:0]로 넣는다.



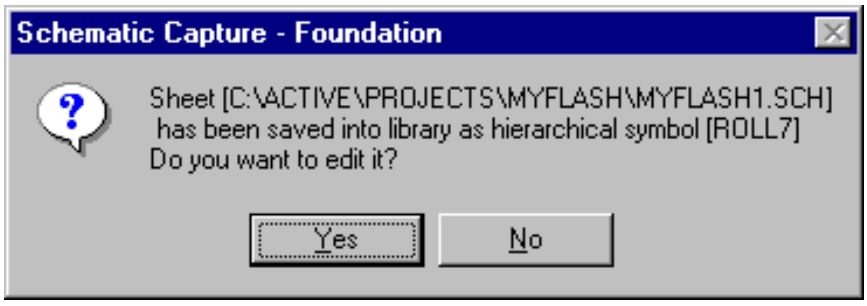
16. LXCOUNT0의 출력 핀으로부터 버스를 HEX2LED macro의 HEX[3:0] 입력으로 버스를 그린다.
17.  버튼을 누른다.
18. Bus name 을 HEX[3:0] 라 넣는다.

Macro schematic 저장하기 (Saving the Macro Schematic)

15. Macro schematic이 다 완성되고 나서 **Hierarchy** 메뉴에서 **Create Macro Symbol From Current Sheet** 명령을 선택한다.
16. **Create Macro** 대화창에서 Symbol Name 필드에 LXROLL7 라 이름한다.
17. **Comment** 필드에 "7 segment LED counter"라 주석을 달아 준다.



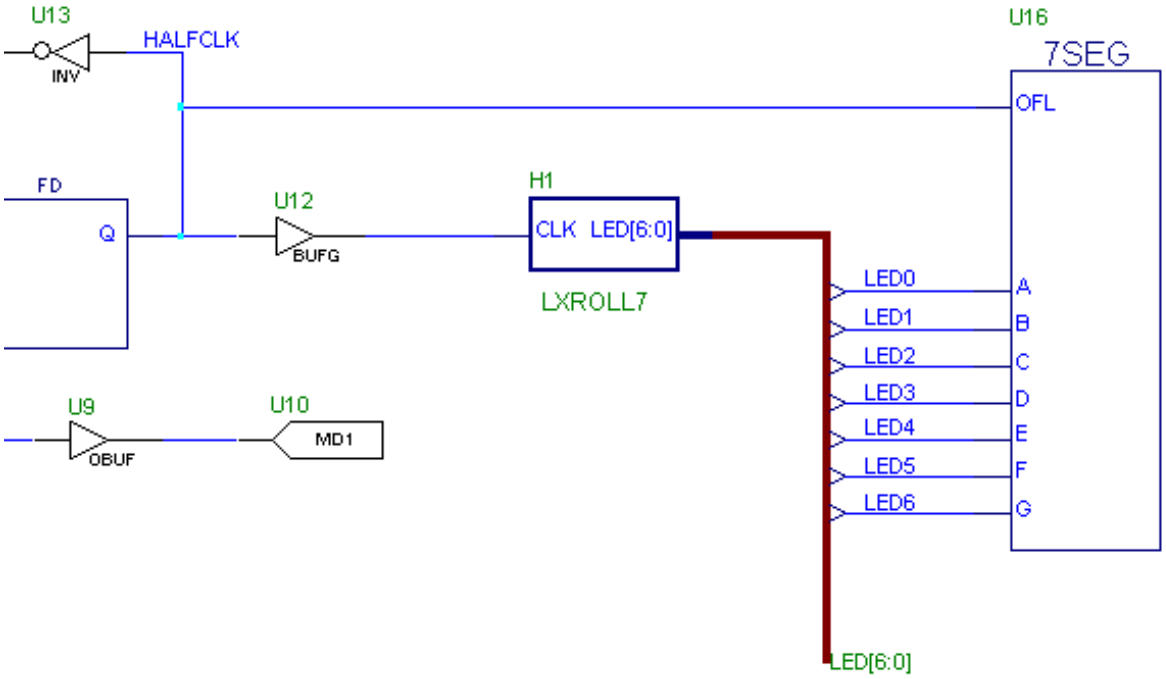
18. Macro을 만들기 위해 **OK** 버튼을 누른다.



4. Macro가 성공적으로 만들어졌다는 정보 창이 보여질 것이다. 이 정보 창이 깜빡일 때 만일 이 매크로를 좀 더 편집하고자 하면 **Yes**를 누르고 **No**를 선택하면 macro schematic이 닫힐 것이다.

상위 schematic 끝내기 (Finishing the Top Level Schematic)

1. Top level schematic로 되 돌아 가서 LXROLL7 심볼을 위치시킨다. 이 심볼은 프로젝트 라이브러리에 macro가 저장되는 동안에 소프트웨어에 의해 자동으로 만들어 졌다.
2. 7SEG macro를 위치시키고 아래 그림과 같이 버스와 Wire를 연결한다.



3.  버튼을 누른다.
4. Bus name (LED[6:0])을 클릭한다.
5. Status line 이 배치를 위해 선택되어진 처음 버스 탭 LED6 을 보여 줄 것이다..

6. Counter의 G 핀을 누른다. 버스 탭이 버스에 그려지고 LED6 로 이름이 붙여질 것이다.
7. 다음 핀들을 F, E, D, C, B, A 순서로 차례차례 누른다.
8. 버스 탭 모드를 빠져 나가기 위해 **Esc** 키를 두 번 누른다.

NOTES:

- Macro schematic 안으로 들어가기 위해,  버튼을 선택하고 원하는 macro 심볼을 두 번 누른다.
- 상위 레벨로 되돌아 가기 위해, 하위 레벨 schematic의 background를 두 번 누른다.

기능적 시뮬레이션 (Functional Simulation)

1. Functional simulation을 수행하기 위해, schematic 툴 바에서 

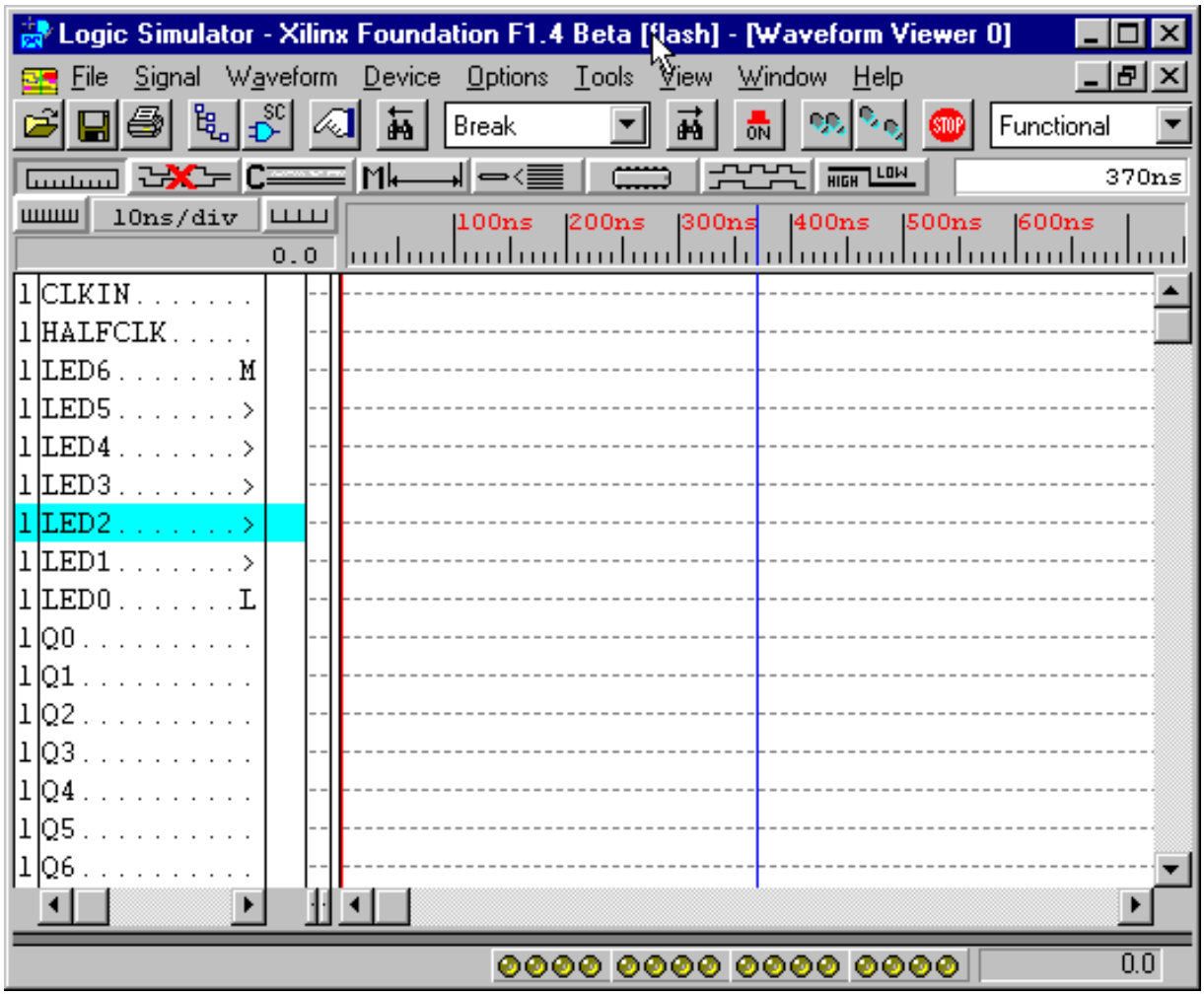
버튼을 누른다. 디자인의 현재 version이 simulator안으로 들어 올 것이다.
2. Simulation 용 test points을 선택하기 위해, logic simulator 안에 있는 

버튼을 누름으로써 schematic 창으로 되돌아 간다. 그런 다음 Schematic Editor 안에 있는  버튼을 누름으로 **Test Points** mode로 들어 간다.
3. Schematic Editor 상에서 다음 네트들을 누른다.
   CLKIN
   HALFCLK
   LED[6:0]
   Q0, Q1...Q7

NOTE: **Waveform Viewer** 창은 버스를 낱개로 또는 묶음 형태 중 하나로 보여 줄 수 있다. Display mode를 바꾸기 위해  버튼을 누른다.

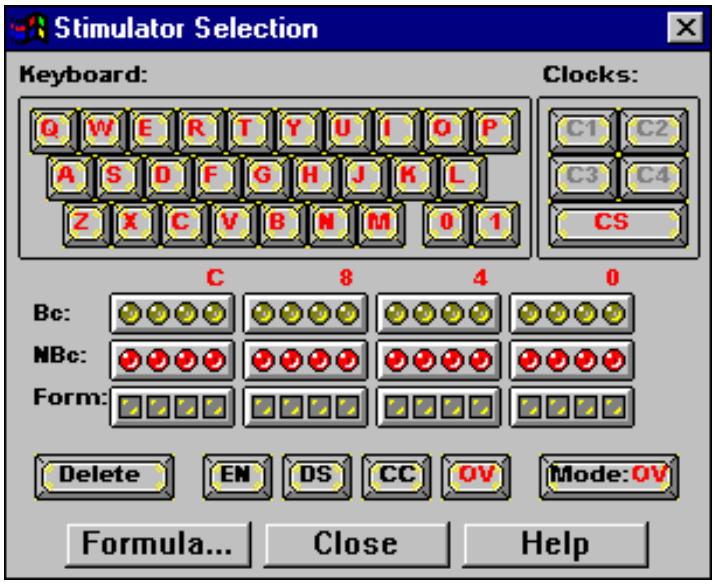
4. Simulator로 되돌아가서 다음 아래에 보이는 것과 같은지 확인 한다.

NOTE: Text Points mode을 벗어 나려면 **Esc** 을 누른다.

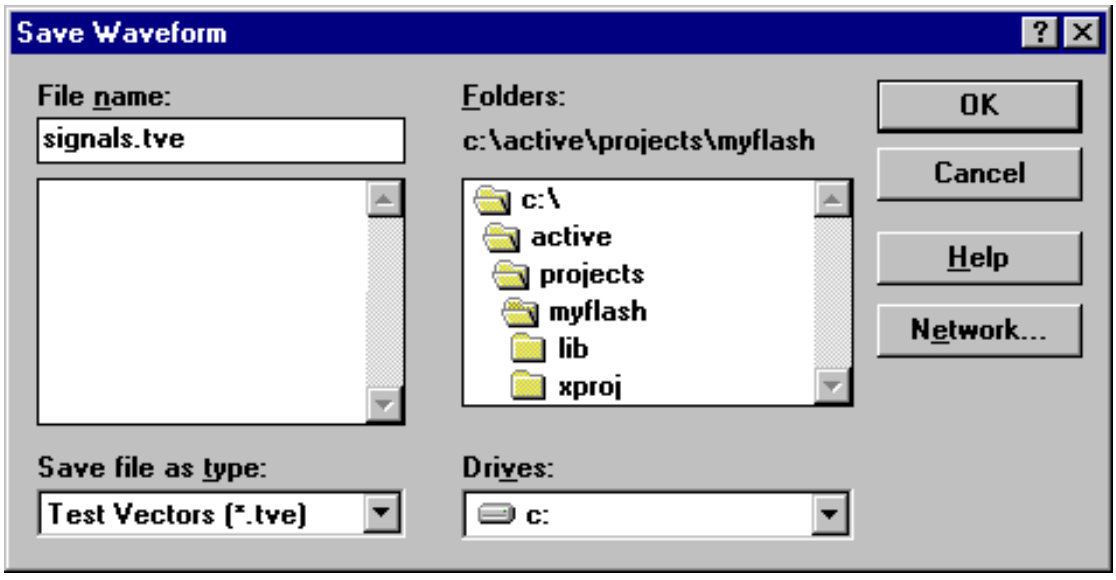


시뮬레이션 데이터 만들기 (Creating Test Vectors)

1. Clock Test Vector를 정의하기 위해, **Waveform Viewer**에서 CLKIN signal을 누르면 CLKIN이 highlight 된다.
2. **Signal** 메뉴에서 **Add Stimulators** 을 선택한다.



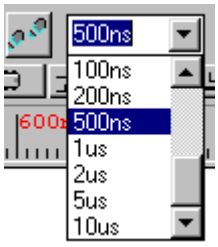
3. Yellow and red *light emitting diode* (LED) icons 은 이진 클럭 발생기의 그림적인 표현들이다. 오른쪽에서 처음 노란 아이콘은 10ns 주기를 가지는 B0 클럭이고, 다음 아이콘은 20ns 주기를 가지는 아이콘이 된다. (Default 설정값을 가지고 있다고 가정하였을 때)
4. 오른쪽에서 두 번째 노란 아이콘을 누르면 **Waveform Viewer** 안에 CLKIN signal에 B1 clock이 보여질 것이다..
5. 선택된 신호를 저장하기 위해서, **File** 메뉴에서 **Save Waveform** 명령을 사용한다.



6. 이름을 “signals”이라 치고 **OK** 버튼을 누른다. 이것은 SIGNALS.TVE 파일을 저장한다.

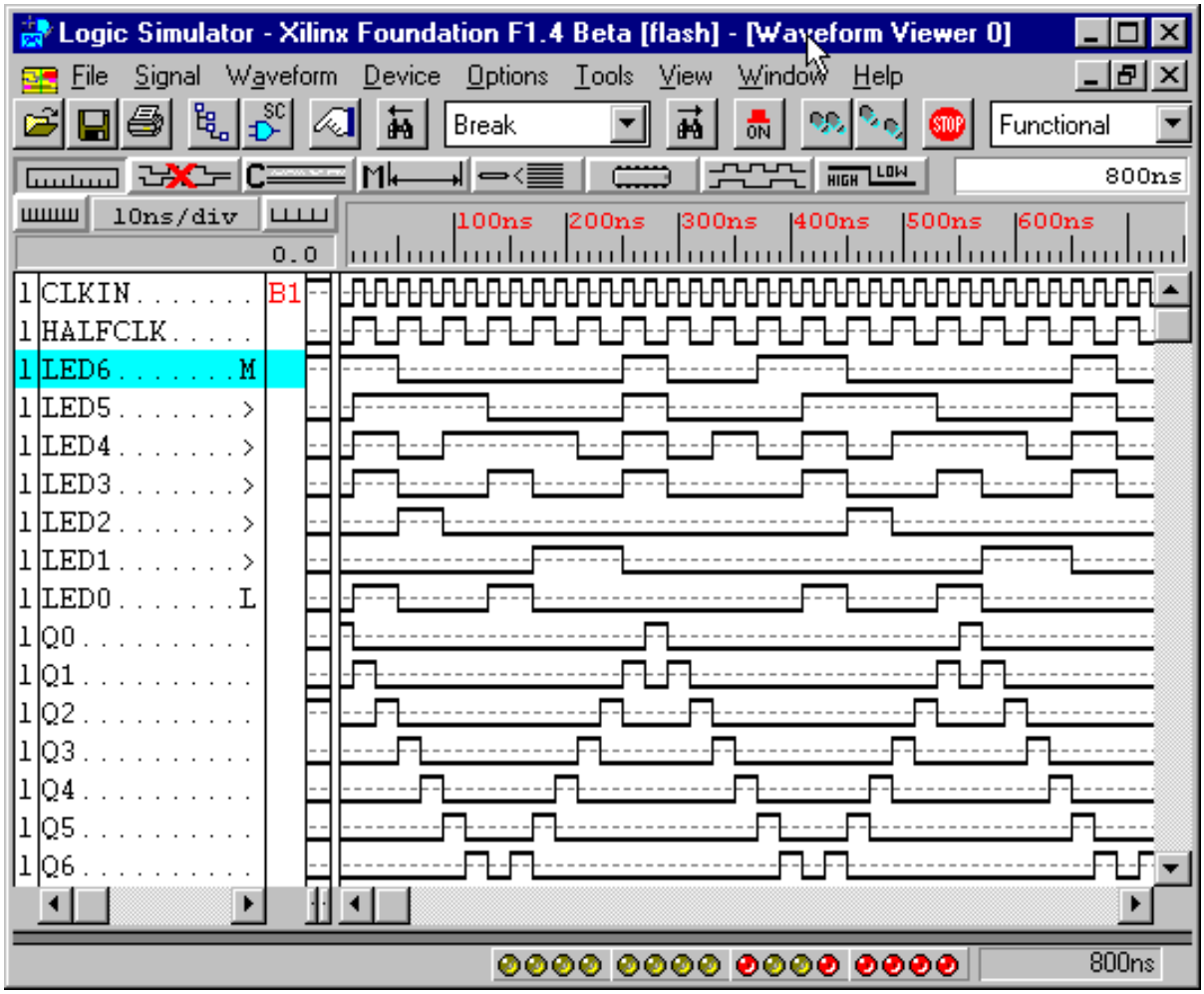
시뮬레이션 실행하기 (Running Simulation)

1. Logic Simulator 에서 아래 그림과 같이 simulation period을 정의한 후 발자국 버튼을 누른다.



2. 아래 그림과 같이 waveform이 발생되어 질것이다.

3.  버튼을 여러 번 누른다.



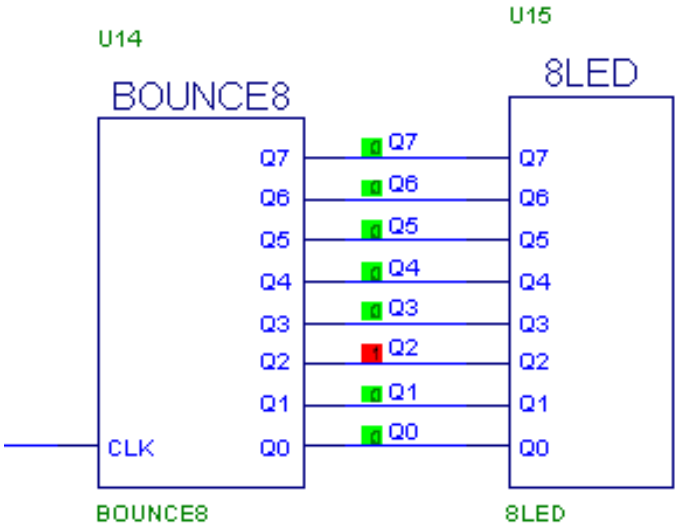
4. **File** 메뉴의 **Save Wafeform** 명령어를 이용하여, 파일명을 FUNCTNL로 하여 simulation 결과를 저장한다. 이 파일은 place and route 후에 나오는 timing simulation 결과와 비교하는

데 사용 되어 질 수 있다.

5. 위의 그림 상의 menu는 software의 version에 따라 다를 수 있다.

Schematic에서 결과 보기 (Viewing Results on the Schematic)

1. Schematic 창으로 전환한다.
2. 선택된 신호들이 색깔 점검단자로 표시되는 것을 주목하라. 로직의 상태에 따라 다음과 같이 색깔이 대응 된다.
   - Green = “0” logic 0
   - Red = “1” logic 1
   - Blue = “X” (Unknown)
   - Yellow = High Impedance/Not driven
3. schematic에서부터 simulation을 계속 진행하려면 **SC Probes** toolbox 안에 있는 **Step** 버튼을 누른다.
4. Q0...Q7 signal lines 상에서 red icon이 위 아래로 이동하는 것이 확인하라.



TIMESPEC Symbol 첨가하기 (Adding TIMESPEC Symbol)

1. 라이브러리에서 TIMESPEC 심볼을 선택하여 그 것을 schematic상에 위치 시킨다.
2. **Esc** 키를 누르고 TIMESPEC 심볼을 두 번 누른다. **Symbol Properties** 대화상자가 나타날 것이다.
3. New parameters 첨가하기:
   - Name filed에 TS01을 넣는다.

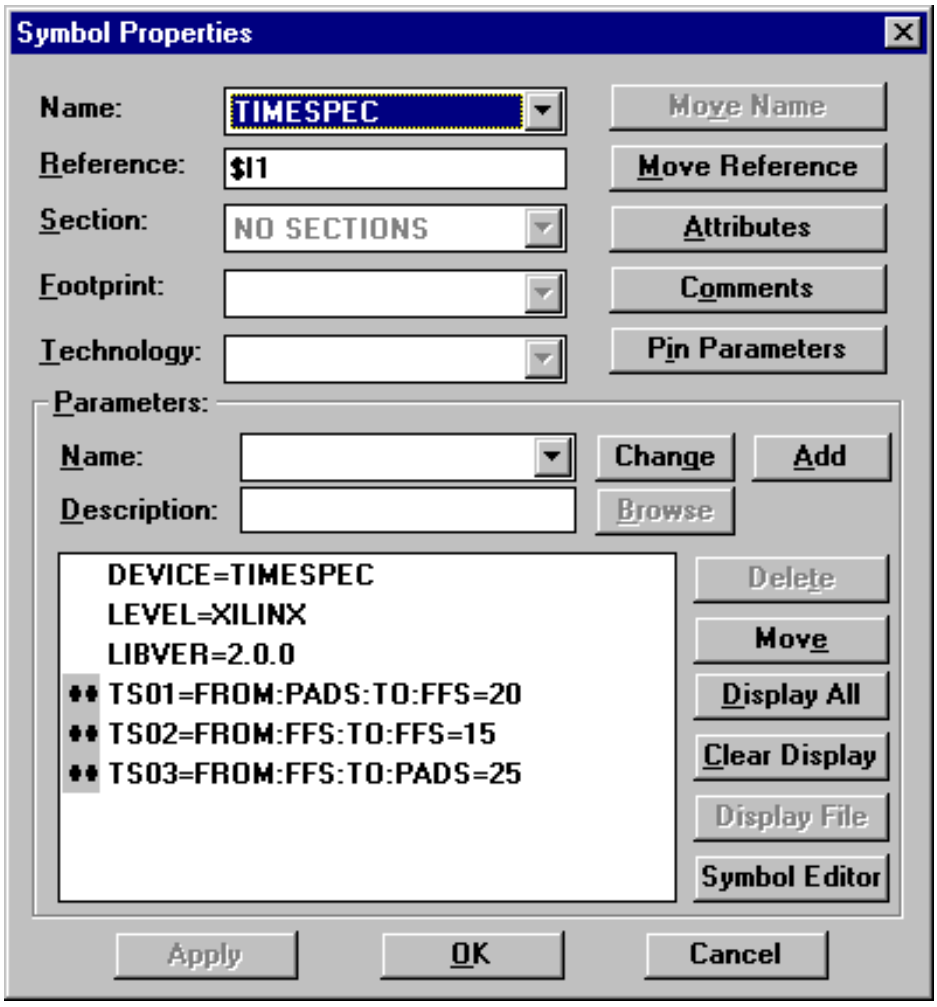
- description field에 FROM:PADS:TO:FFS=20 이라 넣는다.
- **Add**를 누른다.

다음 파라메터를 첨가하기위해 같은 과정을 반복하라.:

- TS02=FROM:FFS:TO:FFS=15
- TS03=FROM:FFS:TO:PADS=25

4. name과 description mark(diamonds)가 보이게 하기 위하여 첨가된 파라메터들을 누른다.
5. **OK** 버튼을 누른다.

NOTE: The diamonds are responsible for displaying the parameters on the schematic



6. 보이는 심볼이 TIMESPEC이 첨가된 결과이다.

TIMESPEC

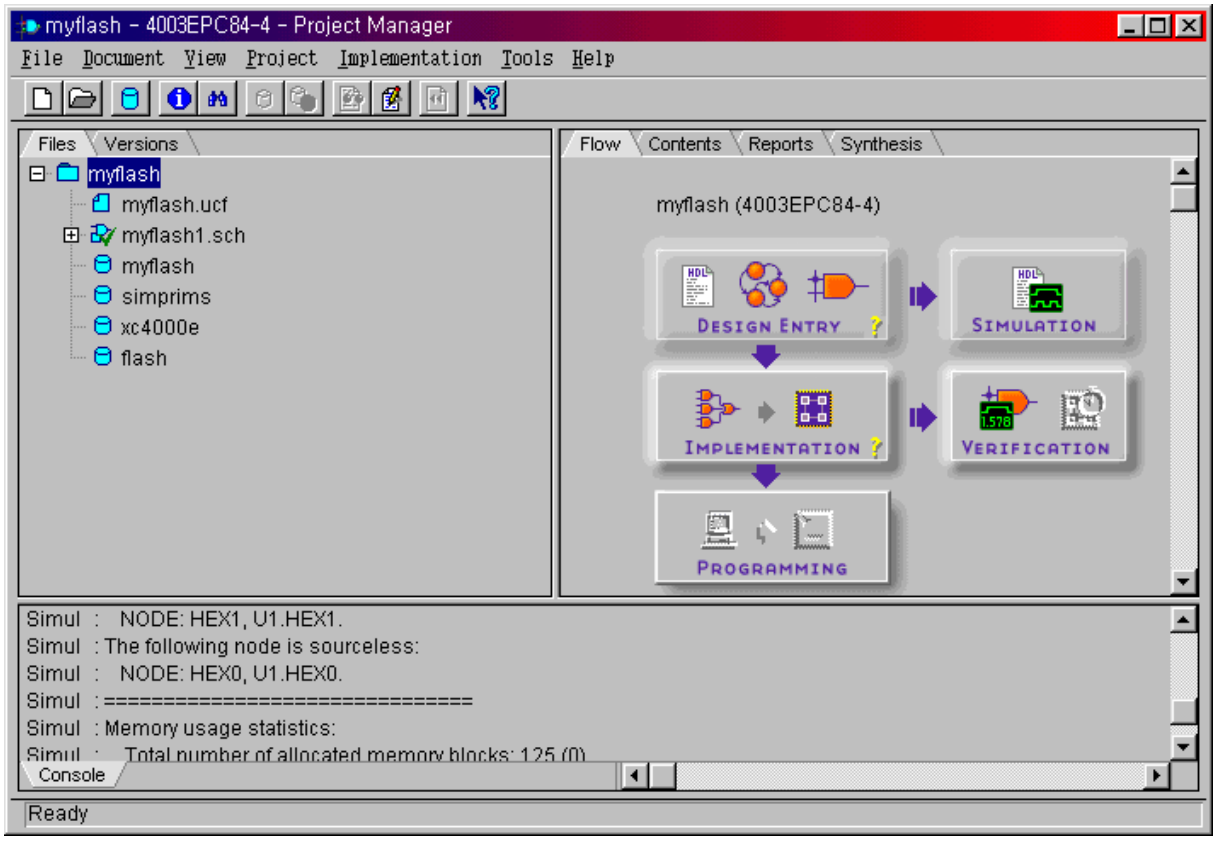
TS01=FROM:PADS:TO:FFS=20
TS02=FROM:FFS:TO:FFS=15
TS03=FROM:FFS:TO:PADS=25

7. Schematic의 변경을 저장하기 위해 **Save** 버튼을 누른다.

XACT 실행하기 (Running XACT)

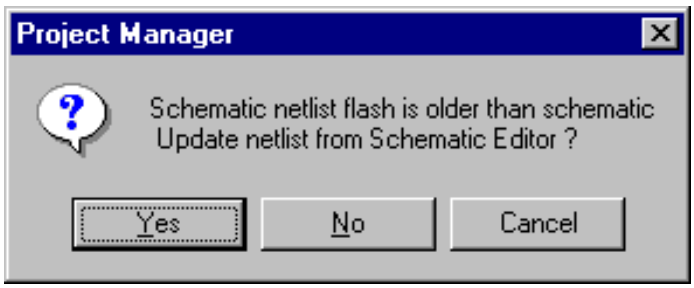
XACT는 Xilinx Automatic CAE Tools의 첫글자를 따서 만들어진 글자로서, 지금까지 작업한 회로를 가지고 실제 Xilinx Device에 들어갈 Data로 변환하는 Place & Route를 수행하는 프로그램을 일컫기도 한다.

1. 먼저 기존에 열어 논 **Schematic** 과 **Simulator** 창을 닫는다. (프로그램을 끝낸다는 의미로서, 굳이 실행을 끝마치지 않아도 XACT를 수행하는데 별이상은 없으나, XACT수행시 좀더 많은 메모리를 확보하기 위해서 이런방법이 권장된다.)
2. **Project Manager** 창에서 **Implementation** 버튼을 누른다.

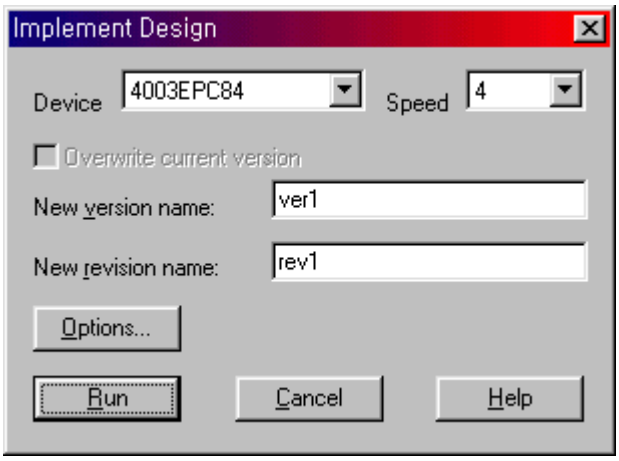


3. Project Manager가 schematic netlist가 up-to-date가 되지 않았다는 정보를 줄 것이다. Netlist를

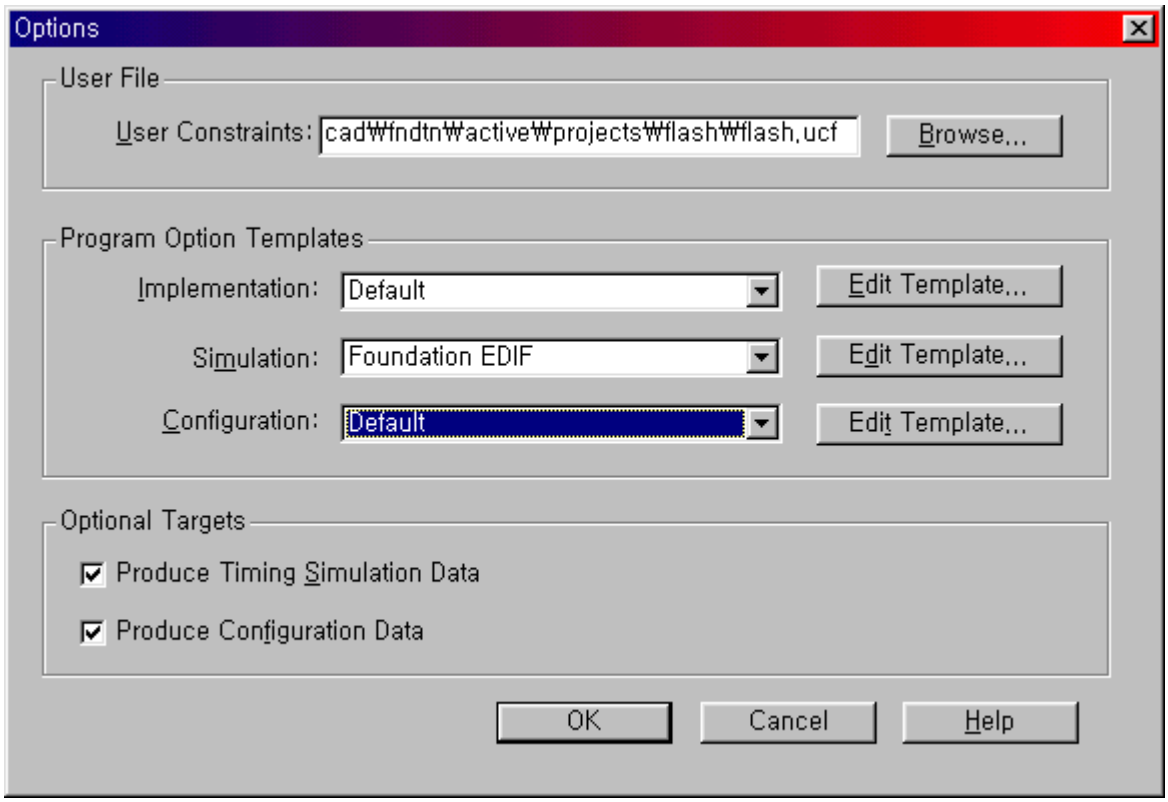
Update하기위해 **Yes**를 누른다.



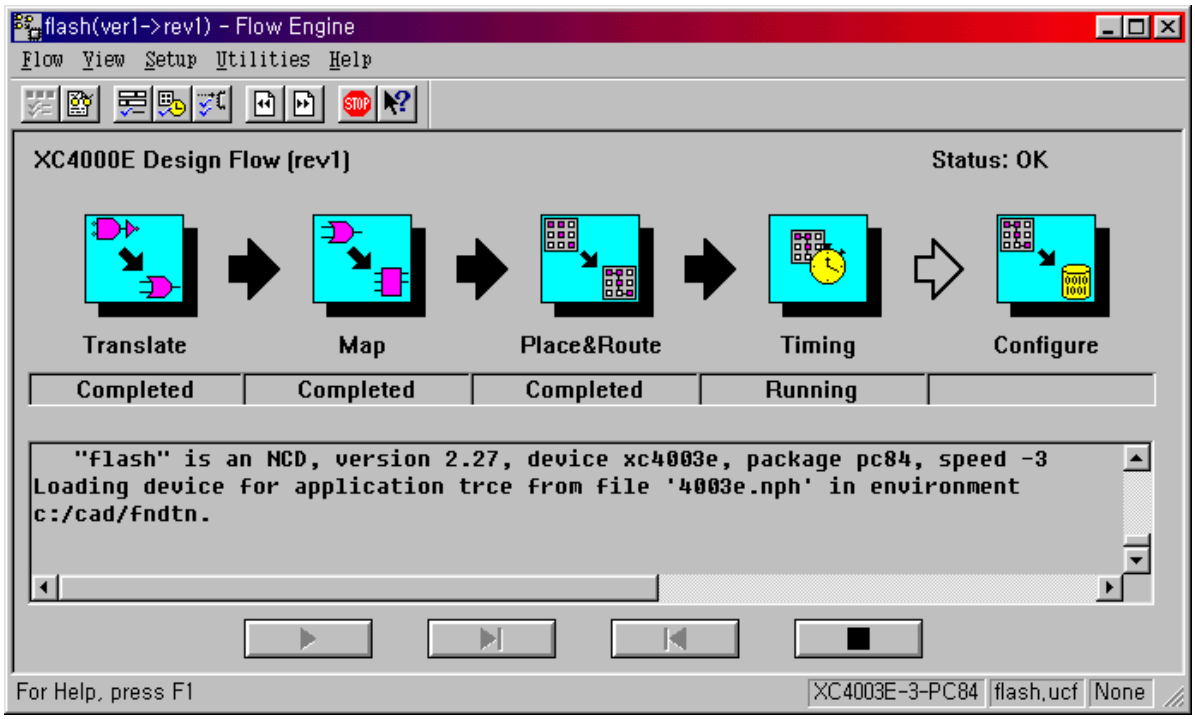
4. 아래 그림과 같은 대화 상자가 나타나면 사용자가 편의에 맞게 Version 및 Revision name을 정의한다.



5. **Options** 버튼을 누른다. 아래의 대화 상자에서 **Produce Timing Simulation Data** check box을 선택하고, **OK**을 누른다.



6. Place and Route 과정을 시작하기 위해 Implement Design 대화 상자에서 **Run** 버튼을 누른다.



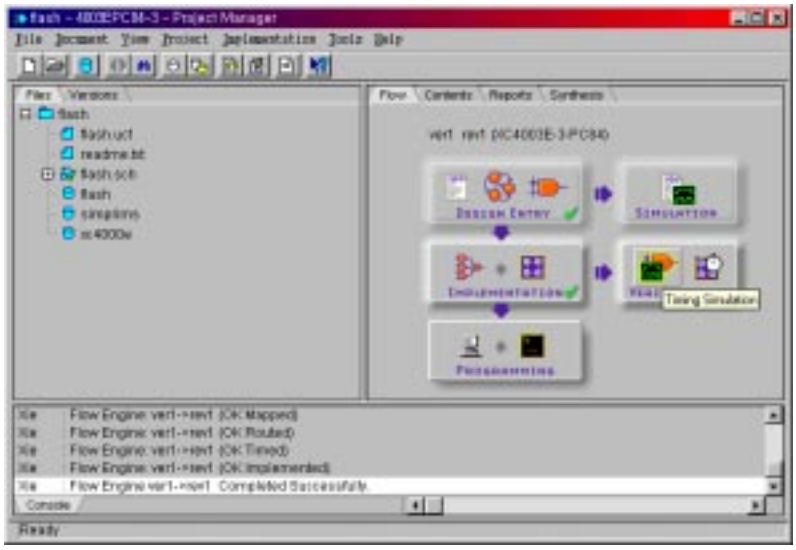
7. PAR(Place and Route)과정이 다 완료되면, Design Manager가 메시지 box를 보여 줄 것이다. Design Manager를 닫기 위해 **OK**를 누른다. 각 단계에 대한 간략한 설명은 아래와 같다.
   - TRANSLATE : XNF, EDIF Format의 Netlist 파일을 읽어 들여 AND, OR gate, Decoder,

RAM등과 같은 Logic Element로 표현되는 Logic Design인 NGD(Native Generic Design)파일로 변환하는 과정이다. 이 NGD파일은 Xilinx Internal Database file format이다.

- MAP : 디자인된 Logic Element들을 CLB(Configurable Logic Block)나 IOB(Input Output Block)같은 Physical Element들로 할당하는 과정이다.
- PLACE & ROUTE : Place는 CLB, IOB등 Logic Block들을 Xilinx Device의 특정 위치에 할당하는 과정이고, Route는 Xilinx Device에 위치한 Logic Block들을 Interconnection Element들로 서로 연결하는 과정이다.
- TIMING : 이 과정은 Place & Route된 Design의 Physical Delay을 산출하여 Timing Simulation 수행 시 필요한 Data을 만들며 개략적인 System Performance Report을 만든다.
- CONFIGURE : Physical Implementation를 Binary Stream으로 변환하여 design.bit 파일을 만든다. 여기서 만들어진 bit 파일을 Xilinx Device에 Download할 때 사용한다.

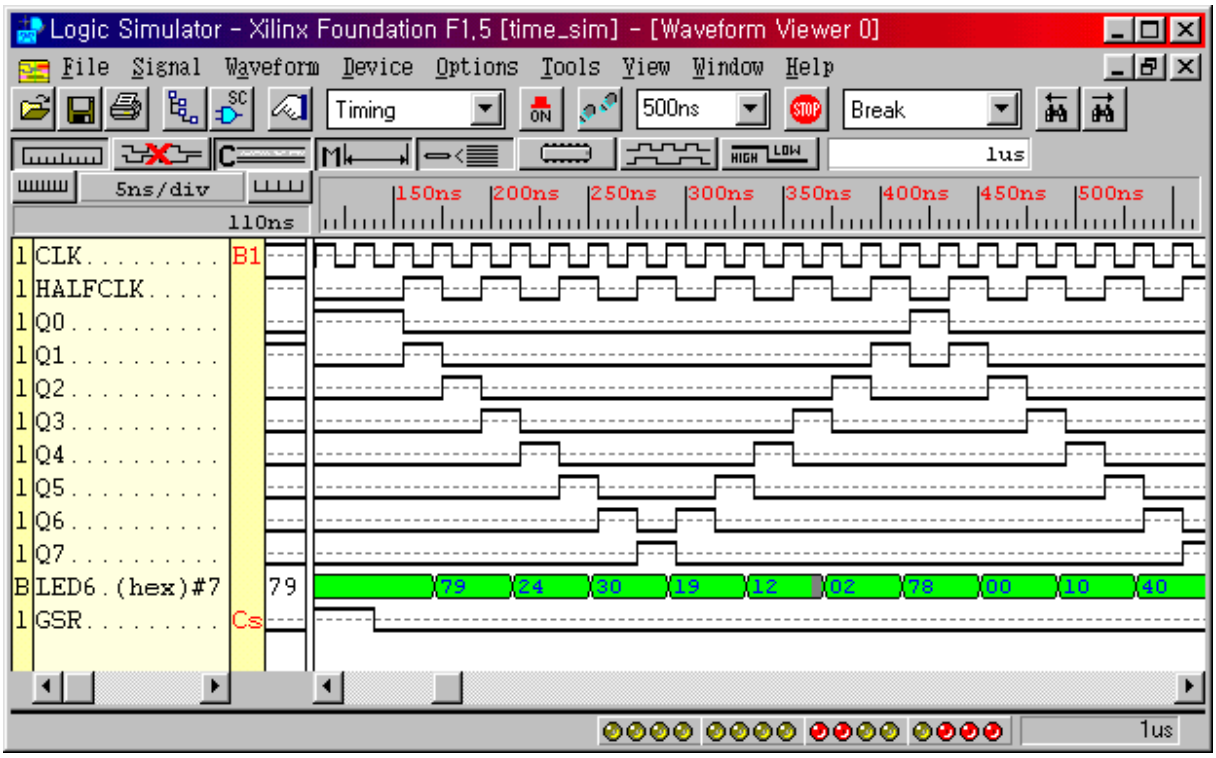
Timing Simulation 하기 (Timing Simulation)

1. 프로젝트 Flowchart에서 **Verification** 버튼을 누름으로 Timing Simulation을 시작한다.

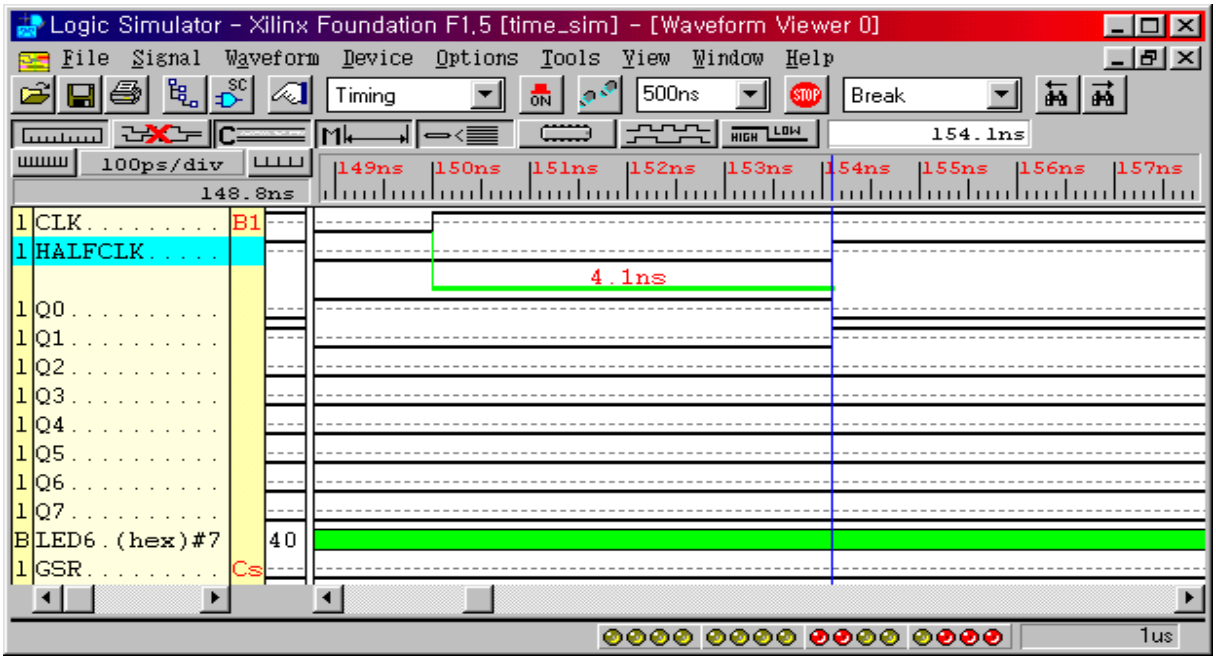


2 Timing simulation Result는 아래 그림과 같을 것이다. 다만 주의해야 할 사항은 FPGA를 디자인 할 때에 Timing Simulation시 Global Reset을 사용하여 Power-On 효과를 내어야 한다는 것이다. 이는 모든 Flip-Flop의 초기값을 "0" 로 만드는 것이다. 여기서는 GSR(Global Set Reset) Signal을 초기에 high로 하여 모든 Flip-Flop을 Clear하고 일정 시간이 경과하면 low로 하여 동작시의 값을 Flip-Flop이 가지도록 한다. (Foundation Series 1.4까지는 위의 작업을 사용자가 일일이 손수해 주었어야 했지만 Foundation Series 1.5에서 부터는 이 작업을

Simulator가 알아서 스스로 해주므로 크게 신경을 쓰지 않아도 된다. 그러므로 사용자는 Timing Simulation시 GSR이란 신호가 Waveform Viewer에 왜 나타나는지만 이해하면 된다.)



3 위의 그림에서 보는 것과 같이 Simulation Mode가 Timing으로 되어 있는 것을 알 수 있을 것이다.



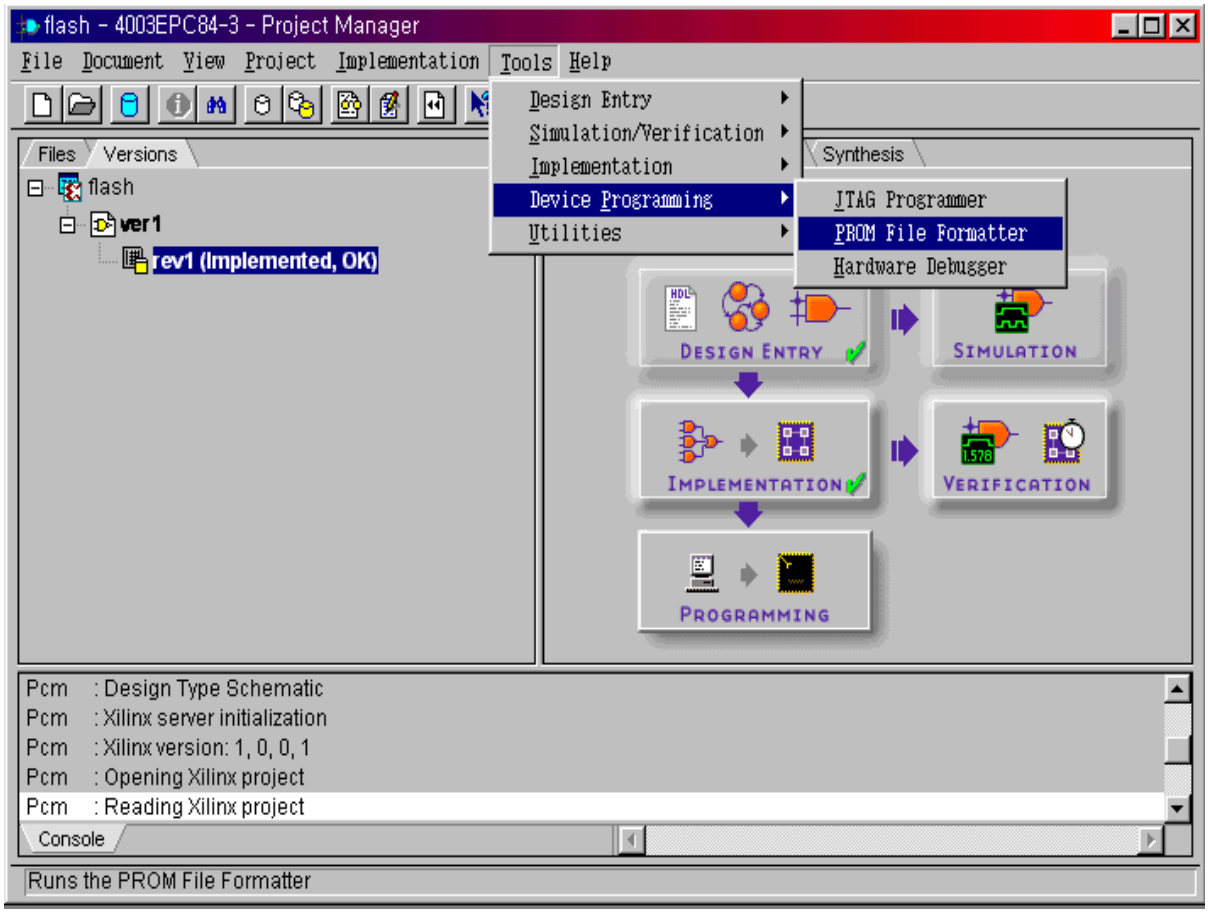
4 위의 그림을 보면 CLK이 분주하여 HALFCLK으로 가는 데 걸리는 실제 delay가 얼마나

생기는 지를 알 수 있을 것이다.

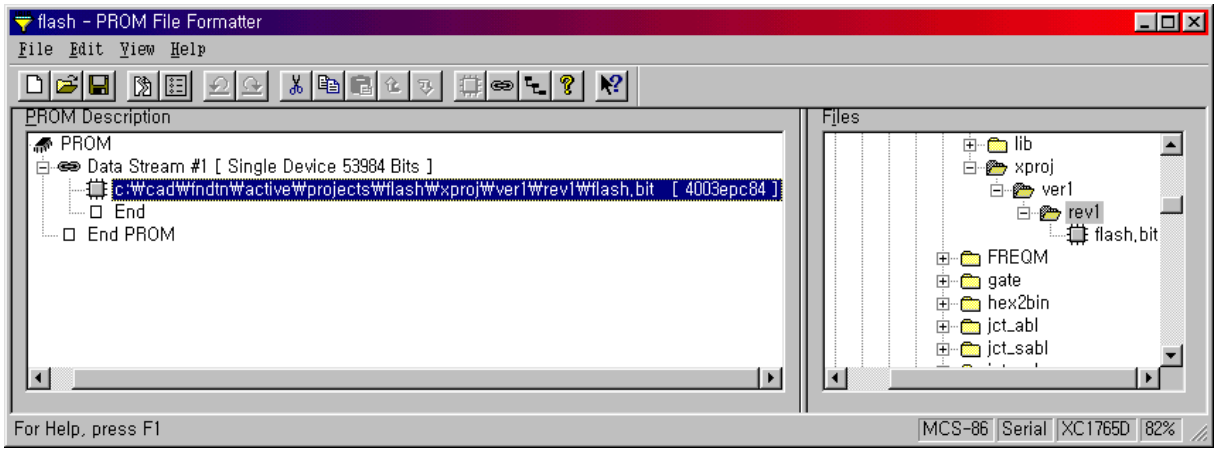
ROM File 만들기 (Creating a MCS date for PROM)

Xilinx FPGA는 SRAM Based Architecture이기 때문에 별도의 Configuration Data을 저장할 ROM (Read Only Memory)이 필요하며 이를 만족시키기 위해서는 Configuration Data을 ROM file 로 저장하여 범용ROM또는 Xilinx Serial PROM에 저장하여야 한다.

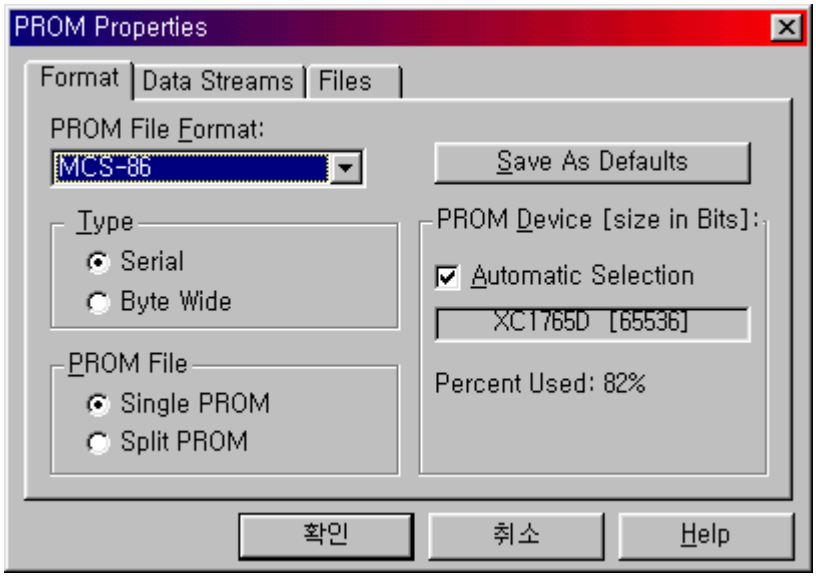
사용자는 ROM에 저장할 Data를 만들어야 한다. 아래 그림에 표현 한 대로 정리하여 만들기 바란다.



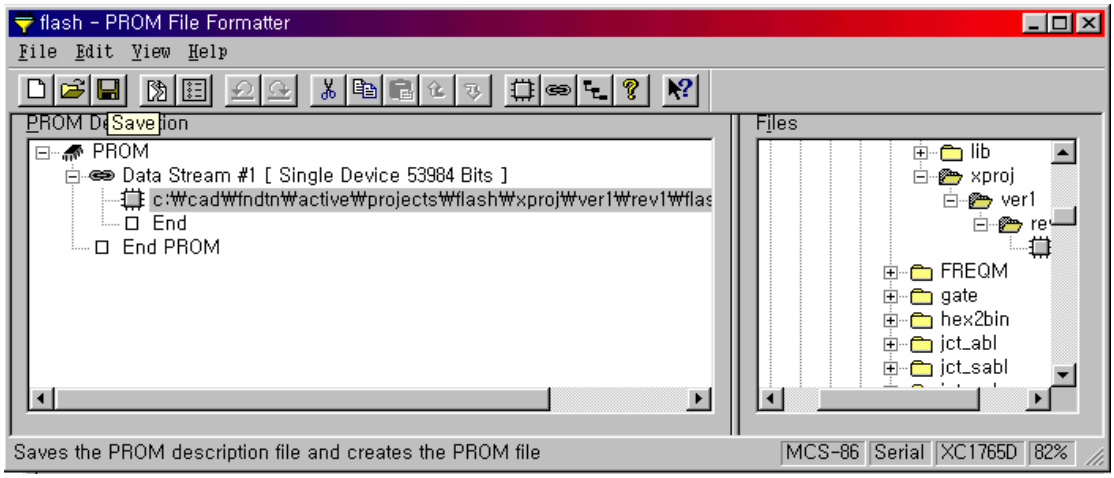
1. 위의 그림에서 보는 바와 같이 Tools -> Device Programming -> PROM File Formatter을 선택한다.
2. 아래 그림처럼 PROM File Formatter라고 하는 별도의 프로그램이 실행되면서 현재 열려있는 Project에서 ROM File을 만들기 위해 선택한 Version의 revison을 자동으로 인식하여 화면 왼쪽 PROM Description란에 자동적으로 등록이 된다.
3. 아래 그림에서 File -> PROM Properties을 선택하여 만들고자 하는 ROM File을 선택한다.



4. 여기서는 Xilinx 의 PROM을 선택한다.



5. 위의 그림에서 사용하는 ROM Writer을 선정하여 거기에 맞는 Data Format을 선택한다. 여기서는 Intel Hex Format인 MCS-86을 선택한다.
6. 이 디자인은 Xilinx PROM인 XC1765D을 자동적으로 선택한다. 만일 Bitstream이 선택한 PROM이 담을 수 있는 용량보다 클 때에는 Split PROM을 선택하여 구분된 Data을 만든다.
7. 원하는 선택 사항을 만들면 확인 버튼을 누른다.



8. 위의 그림에서 Floppy Icon을 눌러 MCS 파일을 저장한다. (실제로 Floppy Icon을 누르면 저장할 파일명을 물어보는 윈도우가 뜨는데 그곳에서는 확장자가 PDR로 지정이 되어있다. 이는 PROM Property에서 선택한 옵션사항들이 저장되는 파일로서 이 파일을 저장하면 자동적으로 MCS File도 생성이 된다.)
9. 오른쪽 Directory Structure에서 저장된 MCS 파일의 Directory를 찾아서 필요한 mcs 파일을 ROM-Writer Program이 있는 directory로 복사하여 필요한 ROM을 굽는다.

후기

이상으로 회로 디자인에서 최종 동작까지 필수적인 과정들을 수행하였습니다.

좀 더 자세한 교육자료가 필요하시면 http://support.xilinx.com/support/techsup/tutorials/index.htm로 가셔서 Foundation F1.5 Watch Tutorial 9/98 (1580KB)과 Foundation Watch Tutorial Design Files (ZIP format), 9/98 (1646KB)을 받아서 업무에 적용하시길 바랍니다.

회로 설계시 의문사항이나 좀 더 자세한 내용을 알고자 하는 분은 현명전자 응용기술부의 F.A.E (Field Application Engineer)에게 도움을 받으시길 바랍니다.

하태욱 twha@hmelec.co.kr

차영근 ykcha@hmelec.co.kr

양창우 cwyang@hmelec.co.kr

이경득 kdlee@hmelec.co.kr

1998년 3월 21일 Revision 2…

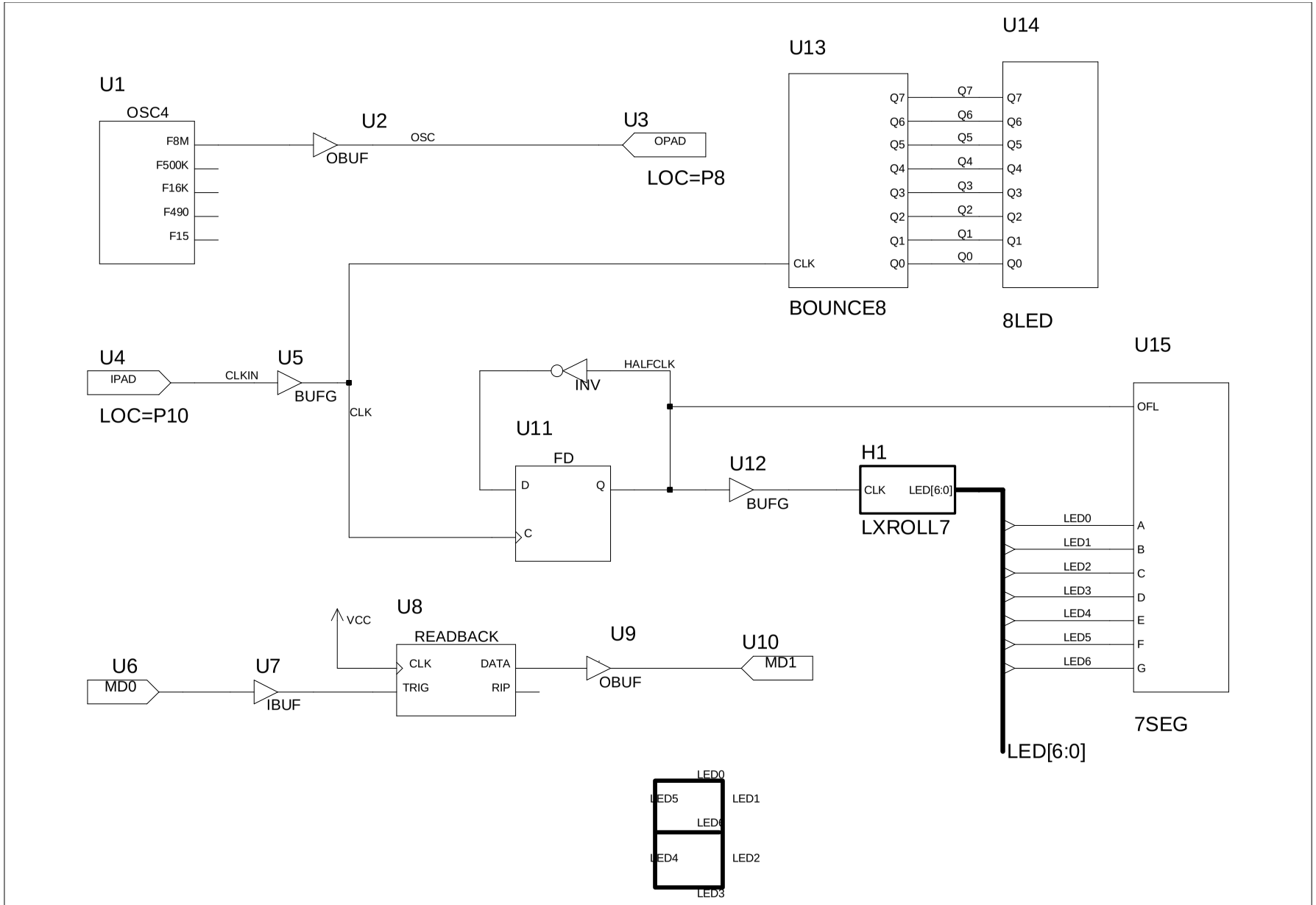
XILINX®

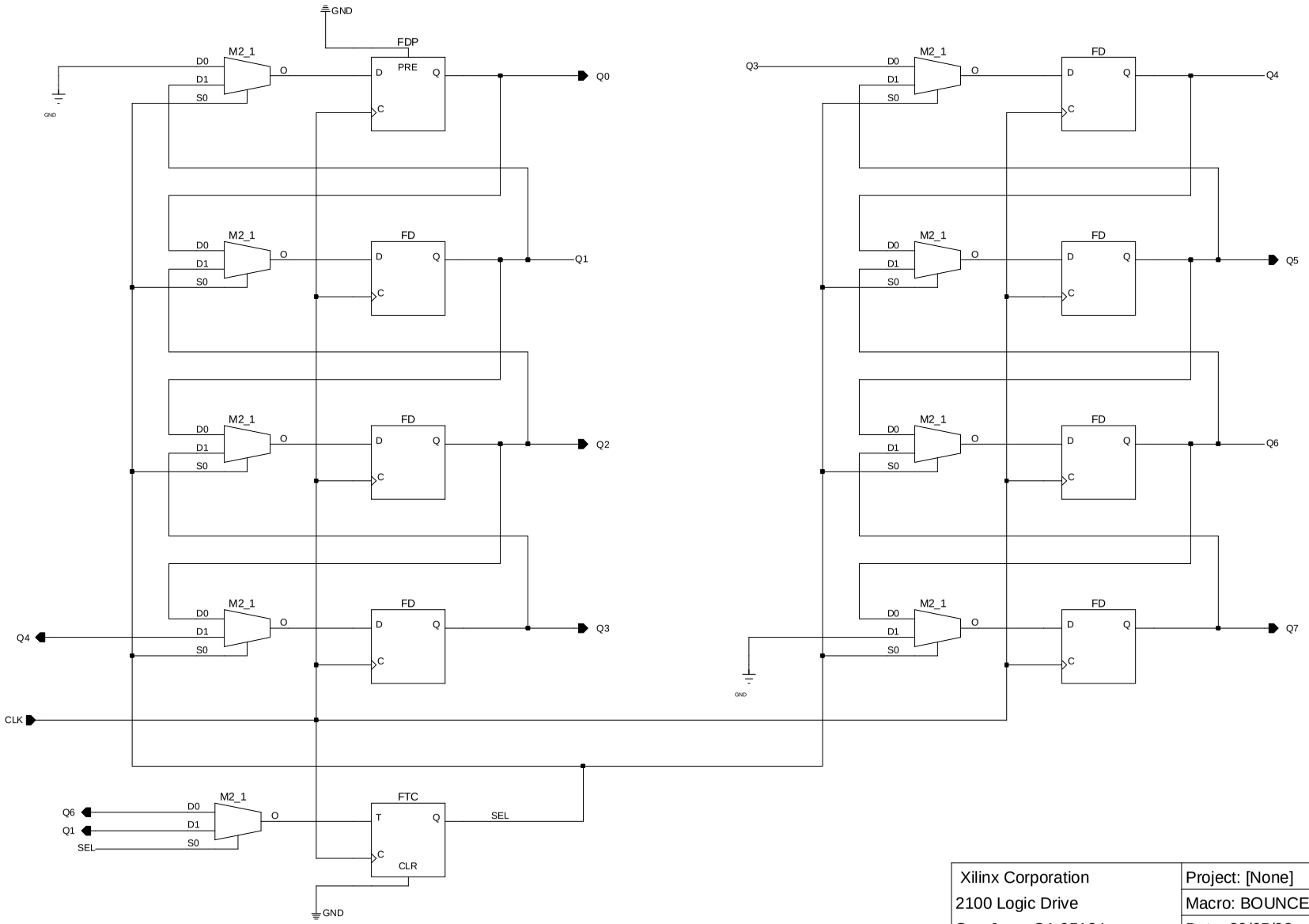
BASIC TRAINING

Lab Circuits

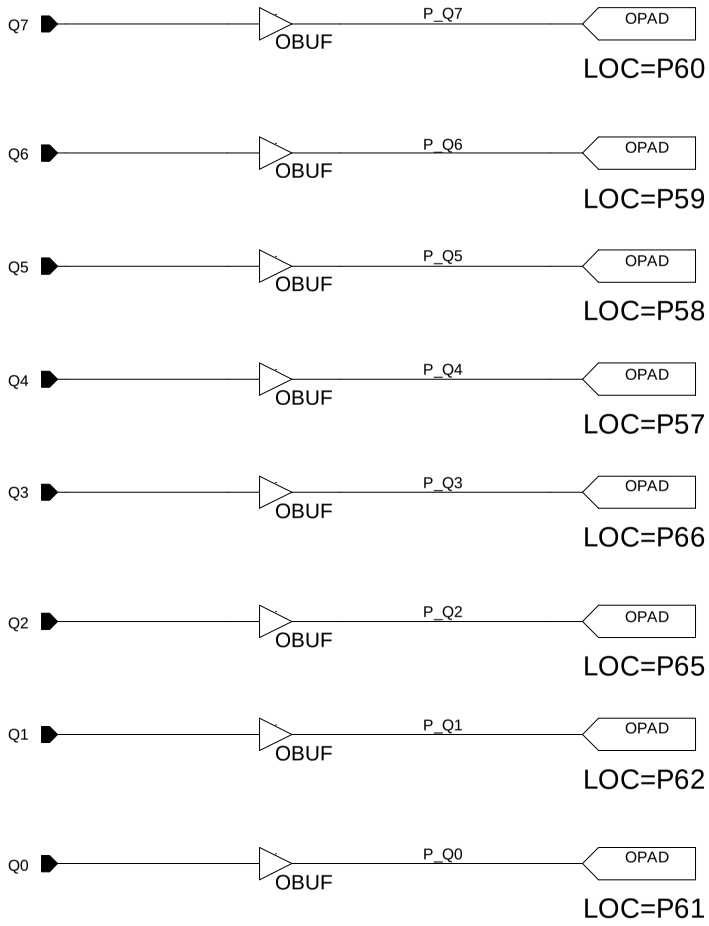
Hyun Myung Electronics Co., Ltd.
Tel 02-3141-0147 / Fax 02-3141-0149
http://www.hmelec.co.kr

XS AT CE TP

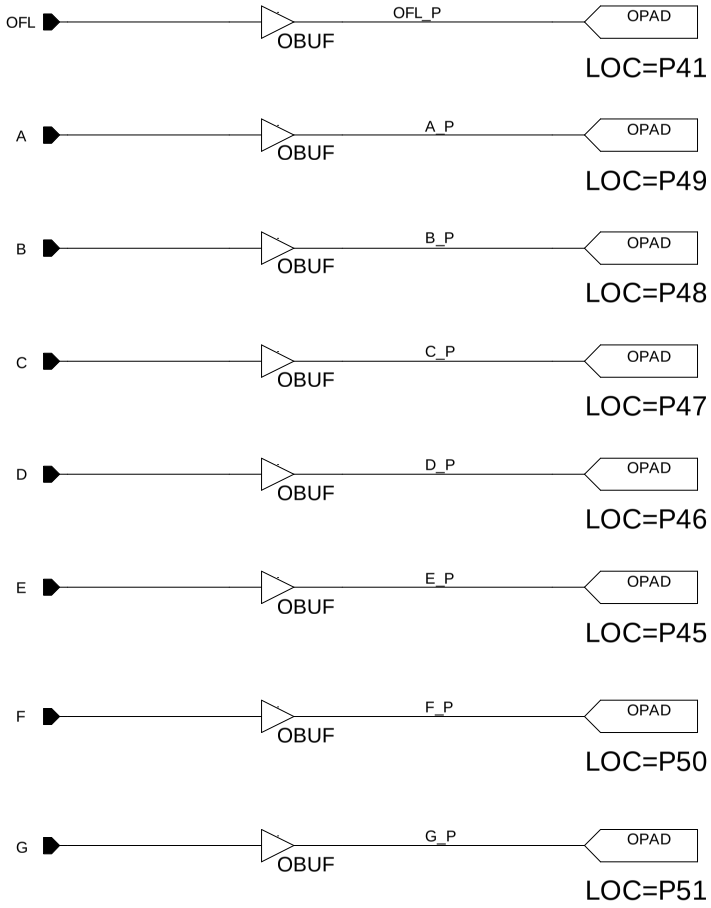

U1
OSC4
F8M
F500K
F16K
F490
F15
U2
OBUF
OSC
U3
OPAD
LOC=P8
U13
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Q0
CLK
BOUNCE8
U14
8LED
U4
IPAD
LOC=P10
U5
BUFG
CLKIN
CLK
HALFCLK
INV
U11
FD
D
Q
C
U12
BUFG
H1
CLK
LED[6:0]
LXROLL7
U15
OFL
A
B
C
D
E
F
G
LED0
LED1
LED2
LED3
LED4
LED5
LED6
7SEG
LED[6:0]
U8
READBACK
VCC
CLK
DATA
TRIG
RIP
U6
MD0
U7
IBUF
U9
OBUF
U10
MD1
LED0
LED1
LED2
LED3
LED4
LED5
Xilinx Corporation
2100 Logic Drive
San Jose, CA 95124
Project: FLASH
Sheet: FLASH
Date: 20/05/98



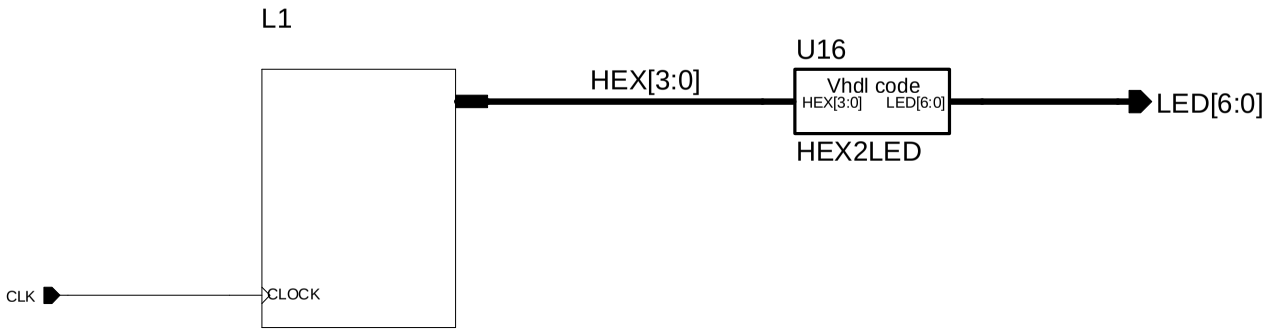
Xilinx Corporation	Project: [None]
2100 Logic Drive	Macro: BOUNCE8
San Jose, CA 95124	Date: 20/05/98



Xilinx Corporation	Project: [None]
2100 Logic Drive	Macro: 8LED
San Jose, CA 95124	Date: 20/05/98

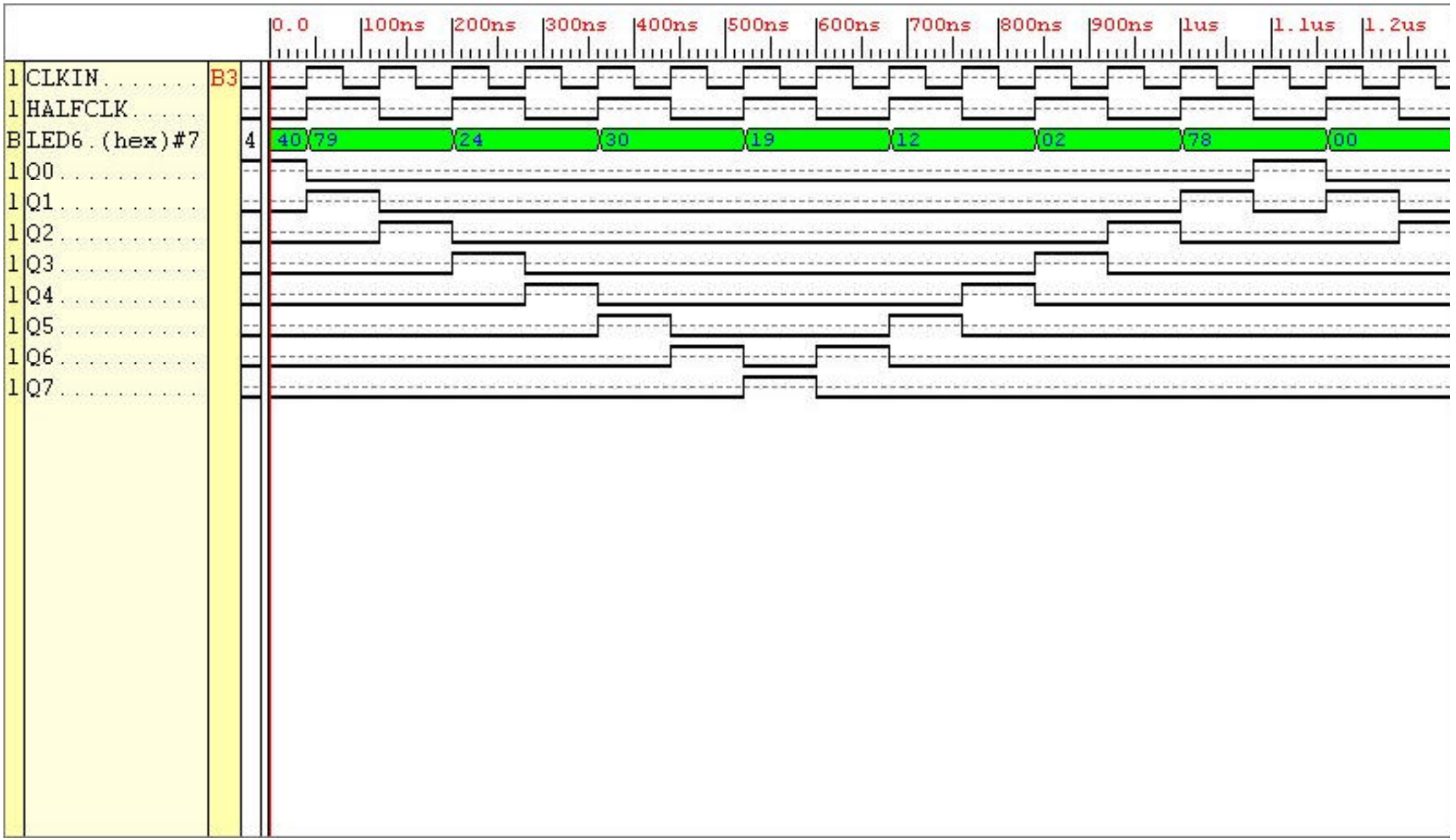


Xilinx Corporation	Project: [None]
2100 Logic Drive	Macro: 7SEG
San Jose, CA 95124	Date: 20/05/98

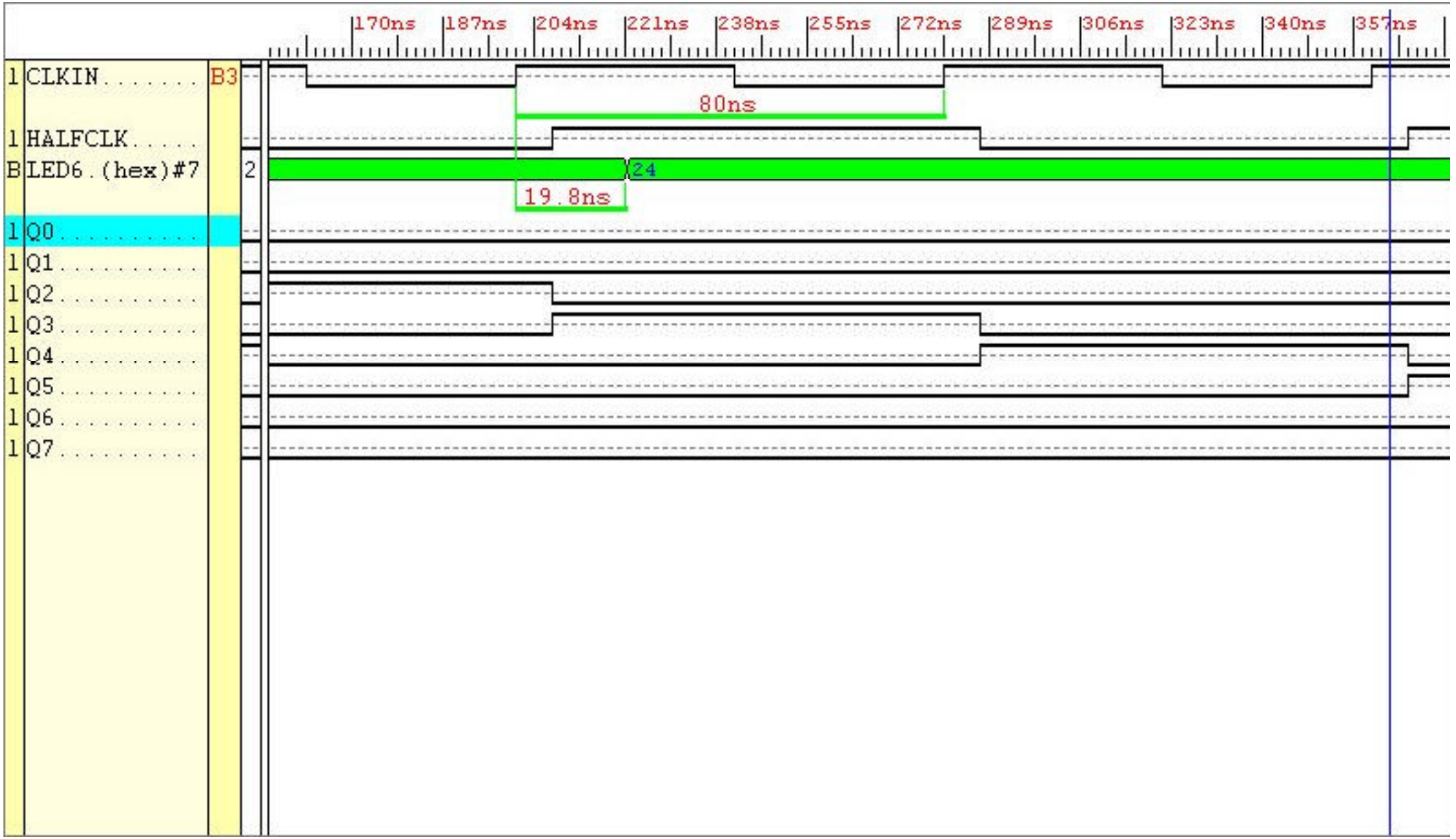


Xilinx Corporation	Project: [None]
2100 Logic Drive	Macro: LXROLL7
San Jose, CA 95124	Date: 20/05/98

```
1:
2:  entity HEX2LED is
3:     port (
4:        HEX:              in                BIT_VECTOR (3 downto 0) ;
5:        LED:              out               BIT_VECTOR (6 downto 0)
6:        ) ;
7:  begin
8:  end HEX2LED ;
9:
10: architecture ARCH_NAME of HEX2LED is
11: begin
12: --HEX-to-seven-segment decoder
13: --          HEX:      in        STD_LOGIC_VECTOR (3 downto 0);
14: --          LED:      out       STD_LOGIC_VECTOR (6 downto 0);
15: --
16: -- segment encoding
17: --     0
18: --    ---
19: --  5 |   | 1
20: --    ---   <- 6
21: --  4 |   | 2
22: --    ---
23: --     3
24:    with HEX select
25:          LED<=     "1111001" when "0001",   --1
26:                    "0100100" when "0010",            --2
27:                    "0110000" when "0011",            --3
28:                    "0011001" when "0100",            --4
29:                    "0010010" when "0101",            --5
30:                    "0000010" when "0110",            --6
31:                    "1111000" when "0111",            --7
32:                    "0000000" when "1000",            --8
33:                    "0010000" when "1001",            --9
34:                    "0001000" when "1010",            --A
35:                    "0000011" when "1011",            --b
36:              "1000110" when "1100",          --C
37:                    "0100001" when "1101",            --d
38:                    "0000110" when "1110",            --E
39:                    "0001110" when "1111",            --F
40:                    "1000000" when others;            --0
41:
42: end ARCH_NAME ;
```



flash-function-Image1



flash-timing-Image1

XILINX®

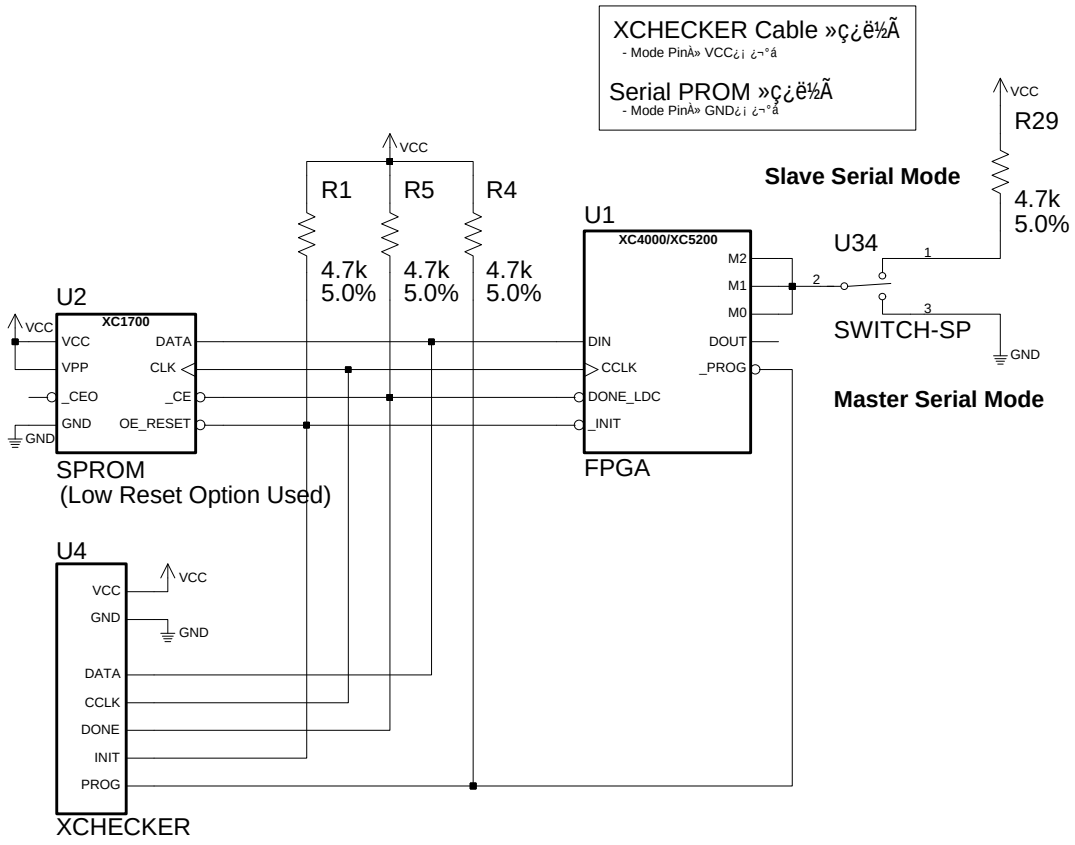
BASIC TRAINING

Configuration

Hyun Myung Electronics Co., Ltd.
Tel 02-3141-0147 / Fax 02-3141-0149
http://www.hmelec.co.kr

XS AT CE TP

FPGA¿Í Serial PROM / X-Checker Cable°úÀç ¿¬°á¹æ¹ý



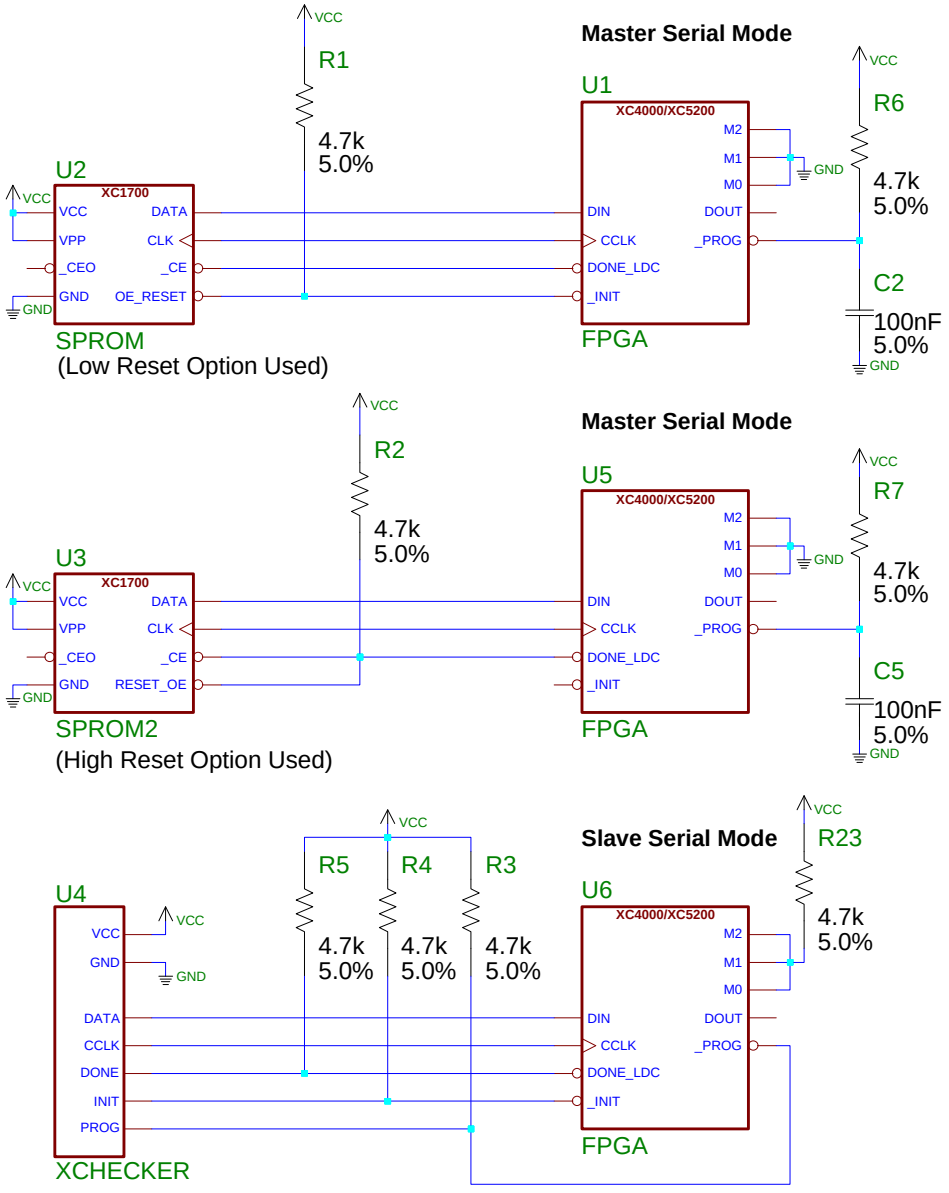
M0	M1	M2	
0	0	0	MASTER
1	1	1	SLAVE
1	1	1	X-Checker

===== Âü°í»çÇ× =====

* SPARTAN Device´Â Mode pinÀÌ ÇÏ³ªÀÌ¹Ç·Î M0, M1À° No Connection
* °¢DeviceÀÇ Pin nameÁß Underscore´Â Active LOW·Î µ¿ÀÛÇÔÀ» ³ªÅ¸³½´Ù
* DONE_LDC´Â DONE pin¶ç´Á /LDC pinÁß ÇÏ³ª¸¸ ÀÇ¹ÌÇÔ
* XC1700 Serial PROMÀÇ RESET, OEÀÇ ±Ø¼ºÀ° ROM Writer¿¡¼ ¼³Á¤ÇÑ´Ù
* ÀúÇ×Àº 2K ~ 10K»çÀÌÀÇ °ªÀ» »ç¿ëÇÒ ¼öÀÖ´Ù

Hyun Myung Electronics Co., Ltd.	Project: CONFIG
+82-2-3141-0147	Sheet: CONFIG6
hmkor@hmelec.co.kr	Date: 10/03/98

FPGA¿Í Serial PROM°úÀÇ ¿¬°á¹æ¹ý-1-



X-Checker Cable°ú FPGA°£ÀÇ °á¼±¹æ¹ý

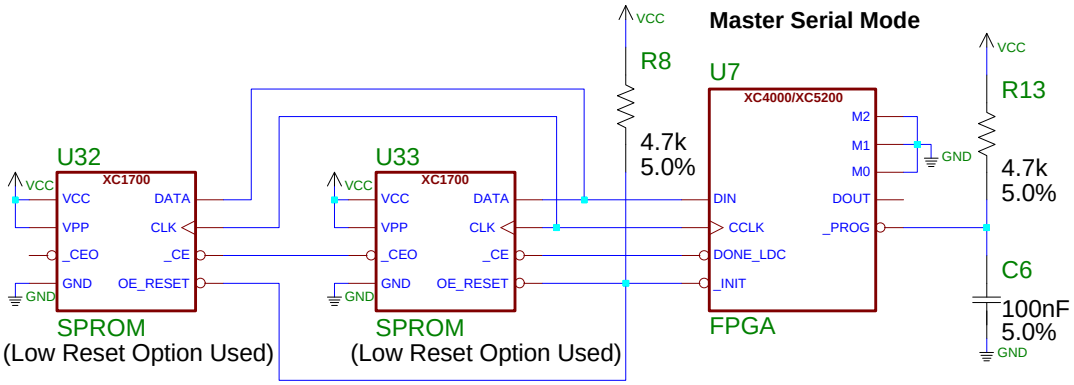
M0	M1	M2	
0	0	0	MASTER
1	1	1	SLAVE
1	1	1	X-Checker

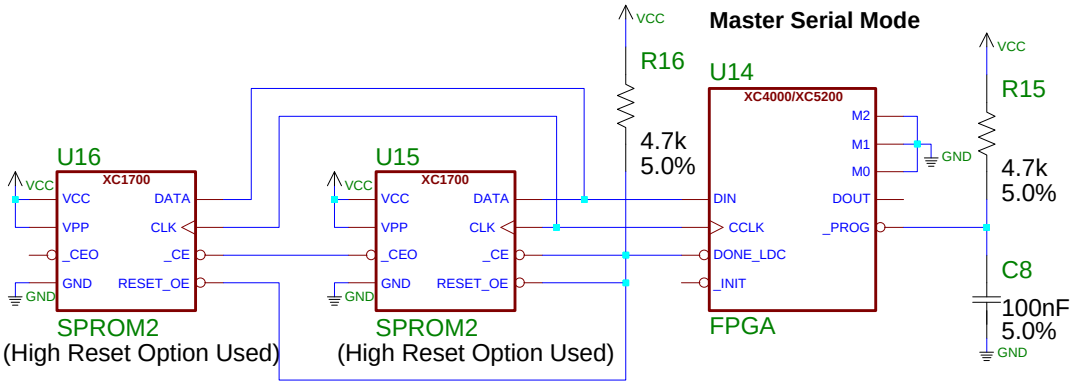
===== Âü°í»çÇ× =====

* SPARTAN Device´Â Mode pinÀÌ ÇÏ³ªÀÌ¹Ç·Î M0, M1À° No Connection
* °¢DeviceÀÇ Pin nameÀß Underscore´Â Active LOW·Î µ¿ÀÛÇÔÀ» ³ªÅ¸³½´Ù
* DONE_LDC´Â DONE pin¶ç´Á /LDC pinÀß ÇÏ³ª¸¦ »ç¿ëÇÔÀ» ÀÇ¹ÌÇÔ
* XC1700 Serial PROMÀÇ RESET, OEÀÇ ±Ø¼°À° ROM Writer¿¡¼ ¼³Á¤ÇÑ´Ù
* ÀúÇ×À° 2K ~ 10K»çÀÌÀÇ °ªÀ» »ç¿ëÇÒ ¼öÀÖ´Ù

Hyun Myung Electronics Co., Ltd.	Project: CONFIG
+82-2-3141-0147	Sheet: config1
hmkor@hmelec.co.kr	Date: 10/03/98

FPGA¿Í Serial PROM°úÀÇ ¿¬°á¹æ¹ý-2-





Cascading Serial Configuration PROMs

(µÎ°³ÀÌ»óÀÇ Serial PROM°ú FPGAÀÇ °á¼±)

===== Âü°í»çÇ× =====

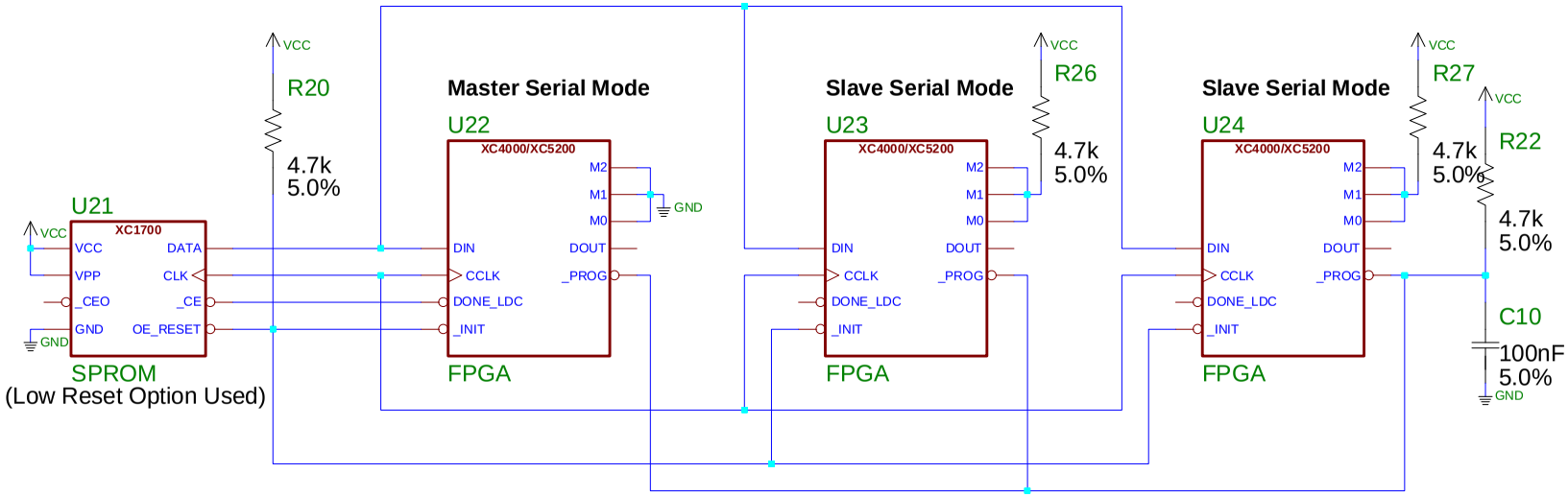
* SPARTAN Device´Â Mode pinÀÌ ÇÏ³ªÀÌ¹Ç·Î M0, M1À° No Connection
* °¢DeviceÀÇ Pin nameÁß Underscore´Â Active LOW·Î µ¿ÀÛÇÔÀ» ³ªÅ¸³½´Ù
* DONE_LDC´Â DONE pin¶ç´Á /LDC pinÁß ÇÏ³ª¸¸ »ç¿ëÇÔÀ» ÀÇ¹ÌÇÔ
* XC1700 Serial PROMÀÇ RESET, OEÀÇ ±Ø¼º°À° ROM Writer¿¡¼ ¼³Á¤ÇÑ´Ù
* ÀúÇ×À° 2K ~ 10K»çÀÌÀÇ °ªÀ» »ç¿ëÇÒ ¼öÀÖ´Ù

M0	M1	M2	
0	0	0	MASTER
1	1	1	SLAVE
1	1	1	X-Checker

Hyun Myung Electronics Co., Ltd.
+82-2-3141-0147
hmkor@hmelec.co.kr

Project: CONFIG
Sheet: Config2
Date: 10/03/98

FPGA¿Í Serial PROM°úÀÇ ¿¬°á¹æ¹ý-3-



ÇÏ³ªÀÇ PROMÀ¸·Î ´Ù¼ö°³ÀÇ FPGA¿¡ µ¿ÀÏÇÑ ³»¿ëÀ» ConfigurationÇÒ¶§ÀÇ °á¼±µµ

M0	M1	M2	
0	0	0	MASTER
1	1	1	SLAVE
1	1	1	X-Checker

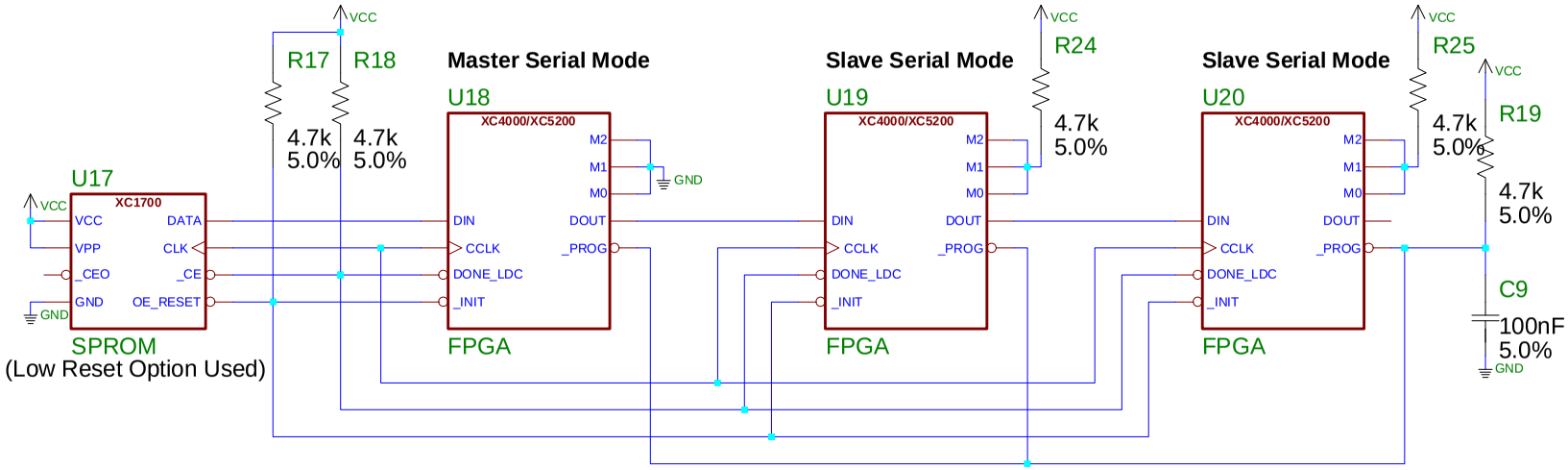
===== Âü°í»çÇ× =====

* SPARTAN Device´Â Mode pinÀÌ ÇÏ³ªÀÌ¹Ç·Î M0, M1À° No Connection
* °¢DeviceÀÇ Pin nameÁß Underscore´Â Active LOW·Î µ¿ÀÛÇÔÀ» ³ªÅ¸³½´Ù
* DONE_LDC´Â DONE pin¶ç´Â /LDC pinÁß ÇÏ³ª, ¦ »ç¿ëÇÔÀ» ÀÇ¹ÌÇÔ
* XC1700 Serial PROMÀÇ RESET, OEÀÇ ±Ø¼ºÀ° ROM Writer¿¡¼ ¼³Á¤ÇÑ´Ù
* ÀúÇ×À° 2K ~ 10K»çÀÌÀÇ °ªÀ» »ç¿ëÇÔ ¹ö ÀÖ´Ù

Hyun Myung Electronics Co., Ltd.
+82-2-3141-0147
hmkor@hmelec.co.kr

Project: CONFIG
Sheet: Config3
Date: 10/03/98

FPGA¿Í Serial PROM°úÀÇ ¿¬°á¹æ¹ý-4-



[Daisy Chain °á¼±µµ]

ÇÏ³ªÀÇ PROMÀ¸·Î ´Ù¼ö°³ÀÇ FPGA¸¦ ConfigurationÇÒ¶§ÀÇ °á¼±µµ

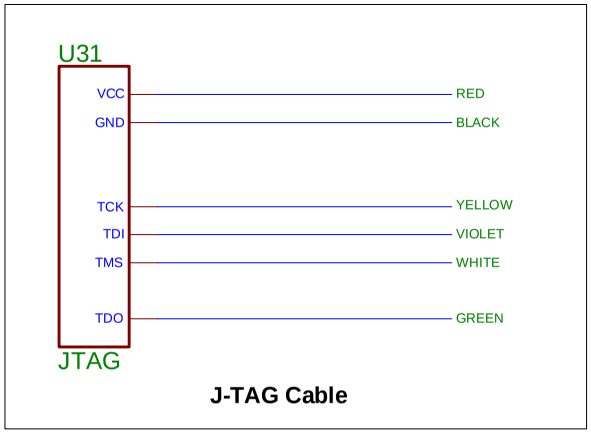
M0	M1	M2	
0	0	0	MASTER
1	1	1	SLAVE
1	1	1	X-Checker

===== Âü°í»çÇ× =====

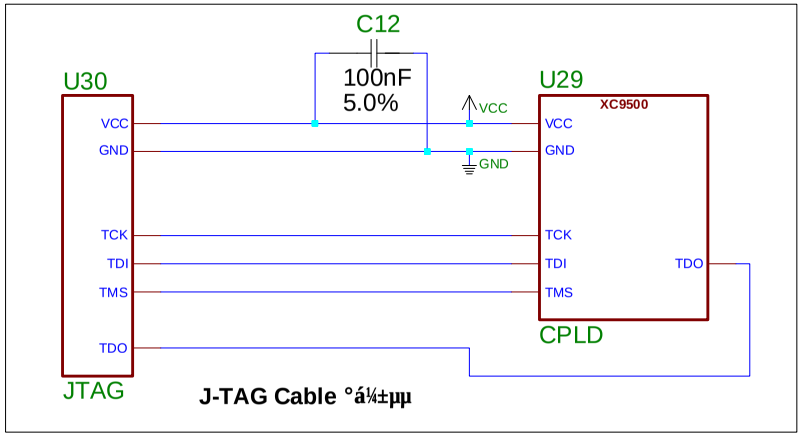
* SPARTAN Device´Â Mode pinÀÌ ÇÏ³ªÀÌ¹Ç·Î M0, M1À° No Connection
* °¢DeviceÀÇ Pin nameÁß Underscore´Â Active LOW·Î µ¿ÀÛÇÔÀ» ³ªÅ¸³½´Ù
* DONE_LDC´Â DONE pin¶ç´Â /LDC pinÁß ÇÏ³ª¸¸ »ç¿ëÇÔÀ» ÀÇ¹ÌÇÔ
* XC1700 Serial PROMÀÇ RESET, OEÀÇ ±Ø¼ºÀ° ROM Writer¿¡¼ ¼³Á¤ÇÑ´Ù
* ÀúÇ×À° 2K ~ 10K»çÀÌÀÇ °ªÀ» »ç¿ëÇÔ ¹ö ÀÖ´Ù

Hyun Myung Electronics Co., Ltd.	Project: CONFIG
+82-2-3141-0147	Sheet: Config4
hmkor@hmelec.co.kr	Date: 10/03/98

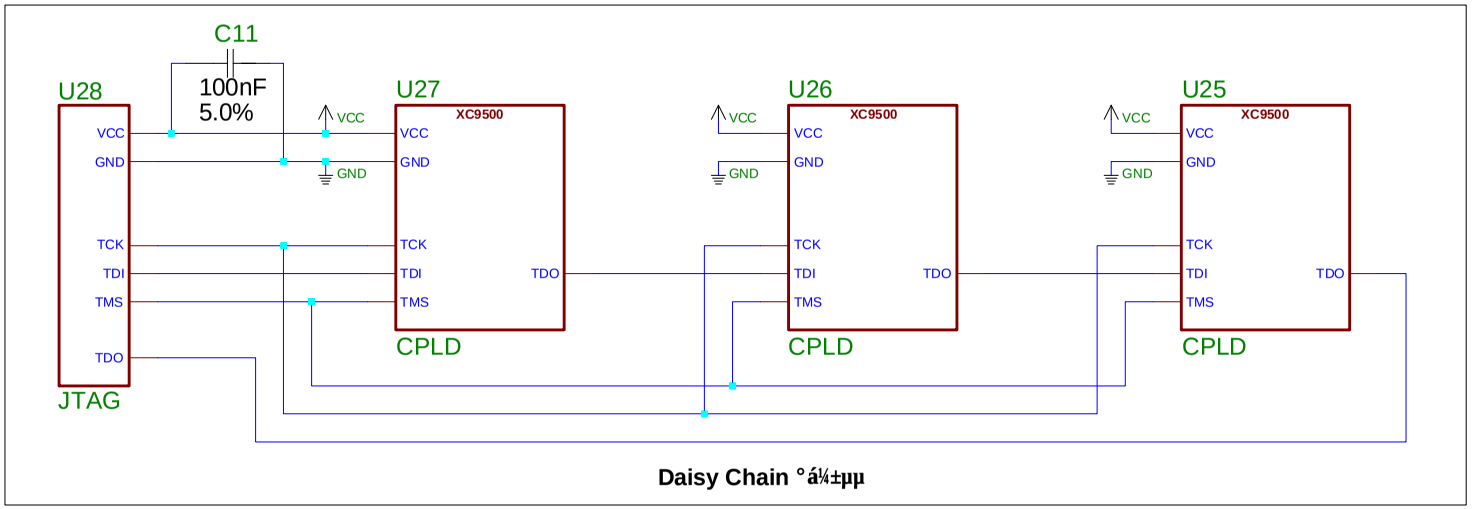
CPLD¿Í J-TAG Cable°úÀÇ ¿¬°á¹æ¹ý (ISP: In-System Programming)



J-TAG Cable



J-TAG Cable °á¼±µµ



Daisy Chain °á¼±µµ

===== Âü°í»çÇ× =====

* Device, ¦ Target PCB³ª Á¦±×(JIG)¿¡ ÀåÂøµÈ »óÅÂ¿¡¼ Power ONµÈÈÄ Data, ¦ ±Á´Â´Ù

* °¢DeviceÀÇ VCC pin¿¡ Bypass Capacitor, ¦ ²À »ðÀÔÇÑ´Ù

Hyun Myung Electronics Co., Ltd.
+82-2-3141-0147
hmkor@hmelec.co.kr

Project: CONFIG
Sheet: Config5
Date: 10/03/98

XILINX®

BASIC TRAINING

Appendix

Hyun Myung Electronics Co., Ltd.
Tel 02-3141-0147 / Fax 02-3141-0149
http://www.hmelec.co.kr

XS AT CE TP

Appendix A

Glossary

This appendix contains definitions and explanations for terms used in the *Foundation Series Quick Start Guide 1.5*.

ABEL

ABEL is a high-level language (HDL) and compilation system produced by Data I/O Corporation.

actions

In state machines, actions are HDL statements that are used to make assignments to output ports or internal signals. Actions can be executed at several points in a state diagram. The most commonly used actions are state actions and transition actions. State actions are executed when the machine is in the associated state. Transition actions are executed when the machine goes through the associated transition.

Aldec

An Electronic Design Automation (EDA) vendor. Aldec provides the Foundation Project Manager, Schematic Editor, Logic Simulator, and HDL Editor.

aliases

Aliases, or signal groups, are useful for probing specific groups of nodes.

analyze

The Foundation Express process in which design source files are examined for correct syntax.

architecture

Architecture is the common logic structure of a family of programmable integrated circuits. The same architecture can be realized in different manufacturing processes. Examples of Xilinx architectures are the XC4000, XC5200, and XC9500 devices.

attributes

Attributes are instructions placed on symbols or nets in an FPGA schematic to indicate their placement, implementation, naming, direction, or other properties.

back-annotation

Back-annotation is the translation of a routed or fitted design to a timing simulation netlist.

BitGen

A program that produces a bitstream for Xilinx device configuration. BitGen takes a fully routed NCD (Circuit Description) file as its input and produces a configuration bitstream, a binary file with a .bit extension.

Black Box Instantiation

Instantiation where the synthesizer is not given the architecture or modules. In Foundation, black boxes are translated with the implementation tools.

block

1. A block is a group of one or more logic functions.
2. A block is a schematic or symbol sheet. There are four types of blocks.
   - A Composite block indicates that the design is hierarchical.
   - A Module block is a symbol with no underlying schematic.
   - A Pin block represents a schematic pin.
   - An Annotate block is a symbol without electrical connectivity that is used only for documentation and graphics.

breakpoint

A breakpoint is a condition for which a simulator must stop to perform simulation commands.

buffer

A buffer is an element used to increase the current or drive of a weak signal and, consequently, increase the fanout of the signal. A storage element.

bus

A bus is a group of nets carrying common information. In LogiBLOX, bus sizes are declared so that they can be expanded accordingly during design implementation.

CLB

The Configurable Logic Block (CLB). Constitutes the basic FPGA cell. It includes two 16-bit function generators (F or G), one 8-bit function generator (H), two registers (flip-flops or latches), and reprogrammable routing controls (multiplexers).

CLBs are used to implement macros and other designed functions. They provide the physical support for an implemented and downloaded design. CLBs have inputs on each side, and this versatility makes them flexible for the mapping and partitioning of logic.

component

A component is an instantiation or symbol reference from a library of logic elements that can be placed on a schematic.

condition

If there is more than one transition leaving a state in a state machine, you must associate a condition with each transition. A condition is a Boolean expression.

constraint

Constraints are specifications for the implementation process. There are several categories of constraints: routing, timing, area,

mapping, and placement constraints.

Using attributes, you can force the placement of logic (macros) in CLBs, the location of CLBs on the chip, and the maximum delay between flip-flops. CLBs are arranged in columns and rows on the FPGA device. The goal is to place logic in columns on the device to attain the best possible placement from the standpoint of both performance and space.

constraints editor

A GUI tool that you can use to enter design constraints. In Foundation 1.5, there are two constraint editors. The Express editor is available only in the Foundation Express product configuration. The Xilinx Constraints Editor is integrated with the Design Implementation tools and available in all product configurations.

constraint file

A constraint file specifies constraints (location and path delay) information in a textual form. An alternate method is to place constraints on a schematic.

CORE Generator

A software tool for generating and delivering parameterizable cores optimized for FPGAs. The library includes cores as complex as DSP filters and multipliers, and as simple as delay elements. You can use these cores as building blocks in order to complete your designs more quickly.

CPLD

Complex Programmable Logic Device (CPLD) is an erasable programmable logic device that can be programmed with a schematic or a behavioral design. CPLDs constitute a type of complex PLD based on EPROM or EEPROM technology. They are characterized by an architecture offering high speed, predictable timing, and simple software.

The basic CPLD cell is called a macrocell, which is the CPLD implementation of a CLB. It is composed of AND gate arrays and is surrounded by the interconnect area.

CPLDs consume more power than FPGA devices, are based on a different architecture, and are primarily used to support behavioral designs and to implement complex counters, complex state machines, arithmetic operations, wide inputs, and PAL crunchers.

CPLD fitter

The CPLD Fitter implements designs for the XC9500 devices.

design entry tools

A set of tools accessible from the Project Manager. These tools include the Schematic Editor, State Editor, and HDL Editor.

Foundation Express, an embedded portion of the Foundation software package, contains the VHDL and Verilog design languages.

daisy chain

A daisy chain is a series of bitstream files concatenated in one file. It can be used to program several FPGAs connected in a daisy chain board configuration.

design implementation tools

A set of tools that comprise the mainstream programs offered in the Xilinx design implementation tools. The tools are NGDBuild, MAP, PAR, NGDAnno, TRCE, all the NGD2 translator tools, BitGen, PROMGen, and EPIC. The GUI-based tools are Flow Engine, Constraint Editor, and Template Manager.

Design Manager

Xilinx Foundation 1.4 graphical user interface for managing and implementing designs. In Foundation 1.4, the Design Manager is accessed by selecting the Implement M1 button from the Project Manager. (For Xilinx Foundation 1.5 series, the Project Manager replaces the Design Manager.)

effort level

Effort level refers to how hard the Xilinx Design System (XDS) tries to place a design. The effort level settings are.

- High, which provides the highest quality placement but requires the longest execution time. Use high effort on designs that do not route or do not meet your performance requirements.
- Medium, which is the default effort level. It provides the best trade-off between execution time and high quality placement for most designs.
- Low, which provides the fastest execution time and adequate placement results for prototyping of simple, easy-to-route designs. Low effort is useful if you are exploring a large design space and only need estimates of final performance.

elaborate

The HDL process that combines the individual parts of a into a single design and then synthesizes the design.

EXORmacs (Motorola)

EXORmacs is a PROM format supported by the Xilinx tools. Its maximum address is 16 777 216. This format supports PROM files of up to (8 x 16 777 216) = 134 217 728 bits.

fanout

Fanout is the maximum number of specified unit loads that a specified output can drive.

fitter

The fitter is the software that maps a PLD logic description into the target CPLD.

floorplanning

Floorplanning is the process of choosing the best grouping and connectivity of logic in a design.

It is also the process of manually placing blocks of logic in an FPGA where the goal is to increase density, routability, or performance.

FPGA

Field Programmable Gate Array (FPGA), is a class of integrated circuits pioneered by Xilinx in which the logic function is defined by the customer using Xilinx development system software after the IC has been manufactured and delivered to the end user. Gate arrays are another type of IC whose logic is defined during the manufacturing process. Xilinx supplies RAM-based

FPGA devices.

FPGA applications include fast counters, fast pipelined designs, register intensive designs, and battery powered multi-level logic.

functional simulation

Functional simulation is the process of identifying logic errors in your design before it is implemented in a Xilinx device. Because timing information for the design is not available, the simulator tests the logic in the design using unit delays. Functional simulation is usually performed at the early stages of the design process.

gate

A gate is an integrated circuit composed of several transistors and capable of representing any primitive logic state, such as AND, OR, XOR, or NOT inversion conditions. Gates are also called digital, switching, or logic circuits.

guided design

Guided design is the use of a previously implemented version of a file for design mapping, placement, and routing. Guided design allows logic to be modified or added to a design while preserving the layout and performance that have been previously achieved.

guided mapping

An existing NCD file is used to "guide" the current MAP run. The guide file may be used at any stage of implementation: unplaced or placed, unrouted or routed. In Release 1.5, guided mapping is supported through the standalone Design Manager.

HDL

Hardware Description Language. A language that describes circuits in textual code. The two most widely accepted HDLs are VHDL and Verilog.

An HDL, or hardware description language, describes designs in a technology-independent manner using a high level of abstraction.

HDL Editor

Foundation’s editor for XABEL, Verilog, and VHDL. The HDL Editor also provides a syntax checker, language templates, and access to the synthesis tools.

HDL Flow

Within the Foundation Project Manager, the Synthesis button displays in the Project Flow area if a project is defined as an HDL Flow. In Release 1.5, an HDL Flow supports VHDL and Verilog top-level designs.

hierarchical design

A hierarchical design is a design composed of multiple sheets at different levels of your schematic.

Hierarchy Browser

The left-hand portion of the Foundation Project Manager that displays the current design project. The browser also displays two tabs, Files and Versions.

implementation

Implementation is the mapping, placement and routing of a design. A phase in the design process during which the design is placed and routed. (For CPLDs, the design is fitted.)

instantiation

Incorporating a macro or module into a top-level design. The instantiated module can be a LogiBLOX module, VHDL module, Verilog module, schematic module, state machine, or netlist.

Language Assistant

The Language Assistant in the HDL Editor provides templates to aid you in common VHDL and Verilog constructs, common logic functions, and architecture-specific features.

Library Manager

The Library Manager is the tool used to perform a variety of operations on the design entry tools libraries and their contents. These libraries contain the primitives and macros that you use to build your design.

locking

Lock placement applies a constraint to all placed components in your design. This option specifies that placed components cannot be unplaced, moved, or deleted.

LogiBLOX

Xilinx design tool for creating high-level modules such as counters, shift registers, and multiplexers.

logic

Logic is one of the three major classes of ICs in most digital electronic systems: microprocessors, memory, and logic. Logic is used for data manipulation and control functions that require higher speed than a microprocessor can provide.

Logic Simulator

The Logic Simulator, a real-time interactive design tool, can be used for both functional and timing simulation of designs. The Logic Simulator creates an electronic breadboard of your design directly from your design’s netlist. The Logic Simulator can be accessed by clicking the Functional Simulation icon on the Simulation phase button or the Timing Simulation icon on the Verification phase button in the Project Manager.

macro

A macro is a component made of nets and primitives, flip-flops or latches, that implements high-level functions, such as adders, subtractors, and dividers. Soft macros and RPMs are types of macros.

A macro can be unplaced, partially placed, or fully placed; it can also be unrouted, partially routed, or fully routed. See also "physical macro."

mapping

Mapping is the process of assigning a design’s logic elements to the specific physical elements that actually implement logic

functions in a device.

MCS file

An MCS file is an output from the PROMGen program in Intel's MCS-86 format.

MRP file

An MRP (mapping report) file is an output of the MAP run. It is an ASCII file containing information about the MAP run. The information in this file contains DRC warnings and messages, mapper warnings and messages, design information, schematic attributes, removed logic, expanded logic, signal cross references, symbol cross references, physical design errors and warnings, and a design summary.

NCD file

An NCD (netlist circuit description) file is the output design file from the MAP program, LCA2NCD, PAR, or EPIC. It is a flat physical design database correlated to the physical side of the NGD in order to provide coupling back to the user's original design. The NCD file is an input file to MAP, PAR, TRCE, BitGen, and NGDAnno.

net

A net is a logical connection between two or more symbol instance pins. After routing, the abstract concept of a net is transformed to a physical connection called a wire.

A net is an electrical connection between components or nets. It can also be a connection from a single component. It is the same as a wire or a signal.

netlist

A netlist is a text description of the circuit connectivity. It is basically a list of connectors, a list of instances, and, for each instance, a list of the signals connected to the instance terminals. In addition, the netlist contains attribute information.

NGA file

An NGA (native generic annotated) file is an output from the NGDAnno run. An NGA file is subsequently input to the appropriate NGD2 translation program.

NGDAnno

The NGDAnno program distributes delays, setup and hold time, and pulse widths found in the physical NCD design file back to the logical NGD file. NGDAnno merges mapping information from the NGM file, and timing information from the NCD file and puts all this data in the NGA file.

NGDBuild

The NGDBuild program performs all the steps necessary to read a netlist file in XNF or EDIF format and create an NGD file describing the logical design. The GUI equivalent is called Translate.

NGD file

An NGD (native generic database) file is an output from the NGDBuild run. An NGD file contains a logical description of the

design expressed both in terms of the hierarchy used when the design was first created and in terms of lower-level Xilinx primitives to which the hierarchy resolves.

NGM file

An NGM (native generic mapping) file is an output from the MAP run and contains mapping information for the design. The NGM file is an input file to the NGDAnno program.

one-hot encoding

For state machines, in one-hot encoding, an individual state register is dedicated to one state. Only one flip-flop is active, or hot, at any one time.

optimization

Optimization is the process that decreases the area or increases the speed of a design. Foundation allows you to control optimization of a design on a module-by-module basis. This means that you have the ability to, for instance, optimize certain modules of your design for speed, some for area, and some for a balance of both.

optimize

The third step in the FPGA Express synthesis flow. In this stage, the implemented design is re-synthesized with constraints the user specifies. This is the final step before writing out the XNF file from FPGA Express.

PAR (Place and Route)

PAR is a program that takes an NCD file, places and routes the design, and outputs another NCD file. The NCD file produced by PAR can be used as a guide file for reiterative placement and routing. The NCD file can also be used by the bitstream generator, BitGen.

path delay

A path delay is the time it takes for a signal to propagate through a path.

PCF file

The PCF file is the "Physical Constraints File" created by the MAP program. It is an ASCII file containing physical constraints created by the MAP program as well as physical constraints you enter. You can edit the PCF file from within EPIC.

PDF file

Project Description File. The PDF file contains library and other project-specific information. Not to be confused with an Adobe Acrobat document with the same extension.

physical Design Rule Check (DRC)

Physical Design Rule Check (DRC) is a series of tests to discover logical and physical errors in the design. Physical DRC is applied from EPIC, BitGen, PAR, and Hardware Debugger. By default, results of the DRC are written into the current working directory.

pin

A pin can be a symbol pin or a package pin. A package pin is a physical connector on an integrated circuit package that carries signals into and out of an integrated circuit. A symbol pin, also referred to as an instance pin, is the connection point of an instance to a net.

pinwires

Pinwires are wires which are directly tied to the pin of a site (that is, CLB, IOB).

PLD

A PLD, or programmable logic device, is composed of two types of gate arrays: the AND array and the OR array, thus providing for sum of products algorithmic representations. PLDs include three distinct types of chips: PROMs, PALs, and PLAs. The most flexible device is the PLA (programmable logic array) in which both the AND and OR gate arrays are programmable. In the PROM device, only the OR gate array is programmable. In the PAL device, only the AND gate array is programmable. PLDs are programmed by blowing the fuses along the paths that must be disconnected.

FPGAs and CPLDs are classes of PLDs.

Project Manager

The primary GUI for managing a Foundation Project. Design entry, synthesis, simulation, implementation, and downloading can be launched from the Project Manager.

Project Flowchart

The right-hand portion of the Foundation Project Manager that provides access to the synthesis and implementation tools, and the current design project. The Project Flowchart can display up to four tabs: Flow, Contents, Reports, and Synthesis.

PROM File Formatter

The PROM File Formatter is the program used to format one or more bitstreams into an MC86, TEKHEX, EXORmacs or HEX PROM file format.

radix

A radix is the base—usually binary, octal, decimal, or hexadecimal— in which waveforms are displayed in a waveform viewer.

readback

- A readback of logic usually accompanied by a comparison check to verify that the design was downloaded in its entirety.
- A readback of the states stored in the device memory elements to ensure that the device is behaving as expected.

route

The process of assigning logical nets to physical wire segments in the FPGA that interconnect logic cells.

route-through

A route that can pass through an occupied or an unoccupied CLB site is called a route-through. You can manually do a route-through in EPIC. Route-throughs provide you with routing resources that would otherwise be unavailable.

Schematic Flow

If a project is defined as a Schematic Flow, no Synthesis button displays in the Project Flow area of the Project Manager. A Schematic Flow may only have schematic designs as top-level designs. However, these top-level designs can contain HDL modules. If the designs contain HDL modules, the Synthesis tab displays in the upper portion of the Project Flow area.

states

The values stored in the memory elements of a device (flip-flops, RAMs, CLB outputs, and IOBs) that represent the state of that device for a particular readback (time). To each state, there corresponds a specific set of logical values.

state diagram

A state diagram is a pictorial description of the outputs and required inputs for each state transition as well as the sequencing between states. Each circle in a state diagram contains the name of a state. Arrows to and from the circles show the transitions between states and the input conditions that cause state transitions. These conditions are written next to each arrow.

state machine

A state machine is a set of combinatorial and sequential logic elements arranged to operate in a predefined sequence in response to specified inputs. The hardware implementation of a state machine design is a set of storage registers (flip-flops) and combinatorial logic, or gates. The storage registers store the current state, and the logic network performs the operations to determine the next state. See also "symbolic state machine" and "encoded state machine."

state table

A state table shows the value of the outputs for all combinations of current states and inputs. It also defines the next state for each set of inputs.

static timing analysis

A static timing analysis is a point-to-point delay analysis of a design network.

static timing analyzer

A static timing analyzer is a tool that analyzes the timing of the design on the basis of its paths.

status bar

The status bar is an area located at the bottom of a tool window that provides information about the commands that you are about to select or that are being processed.

stimulus information

Stimulus information is the information defined at the schematic level and representing a list of nodes and vectors to be simulated in functional and timing simulation.

Synopsys

Synopsys supports HDL, a behavioral language for entering equations. HDL also enables you to include LogiBLOX schematic components in a design.

synthesis

The HDL design process in which each design module is elaborated and the design hierarchy is created and linked to form a unique design implementation. Synthesis starts from a high level of logic abstraction (typically Verilog or VHDL) and automatically creates a lower level of logic abstraction using a library containing primitives.

TEKHEX (Tektronix)

TEKHEX (Tektronix) is a PROM format supported by Xilinx. Its maximum address is 65 536. This format supports PROM files of up to (8 x 65 536) = 524 288 bits.

TRCE

TRCE (Timing Reporter and Circuit Evaluator) "trace" is a program that will automatically perform a static timing analysis on a design using the specified (either timing constraints. The input to TRCE is an NCD file and, optionally, a PCF file. The output from TRCE is an ASCII timing report which indicates how well the timing constraints for your design have been met.

TWR file

A TWR (Timing Wizard Report) file is an output from the TRCE program. A TWR file contains a logical description of the design expressed both in terms of the hierarchy used when the design was first created and in terms of lower-level Xilinx primitives to which the hierarchy resolves.

UCF file

A UCF (User Constraints File) contains user-specified logical constraints.

verification

Verification is the process of reading back the configuration data of a device and comparing it to the original design to ensure that all of the design was correctly received by the device.

Verilog

An industry-standard HDL (IEEE Std 1364) originally developed by Cadence Design Systems, now maintained by OVI. Recognizable as a file with a .v extension.

Verilog is a commonly used Hardware Description Language (HDL) that can be used to model a digital system at many levels of abstraction ranging from the algorithmic level to the gate level. It is IEEE standard 1364-1995.

VHDL

VHSIC (VHSIC an acronym for Very High-Speed Integrated Circuits) Hardware Description Language. An industry-standard (IEEE 1076.1) HDL. Recognizable as a file with a .vhd or .vhdl extension.

VHDL is an acronym for VHSIC Hardware Description Language, which can be used to model a digital system at many levels of abstraction ranging from the algorithmic level to the gate level. It is IEEE standard 1076-1993.

A language that is capable of describing the concurrent and sequential behavior of a digital system with or without timing.

wire

A wire is either 1) a net or 2) a signal.

XABEL

Xilinx ABEL is a design entry package consisting of a Xilinx-specific version of the ABEL design entry software and a series of translation programs. It uses Boolean equations, truth tables, and state machines to create modules and full designs for CPLDs and modules for FPGAs.

XVHDL Compiler

In Foundation 1.4, this compiler synthesizes and generates EDIF 2 0 0 from Metamor VHDL code or state machine designs. It has been replaced by the Synopsys FPGA Express compiler for Foundation Series F1.5.

Foundation Design Entry

HOT KEY LISTS

Key	Project Manager	Schematic Editor	Simulator	HDL Editor	State Editor	Symbol Editor
+	Expand 1 level					
*	Expand Branch					
-	Collapse Branch					
Alt+Bksp						Undo
Alt+F3				Find...	Find String	
Alt+X		Exit				
Ctrl++		Zoom In			Zoom In	Zoom In
Ctrl+*	Expand All					
Ctrl+-		Zoom Out			Zoom Out	Zoom Out
Ctrl+A		Undo		Redo	Redo	
Ctrl+B		Table Setup				
Ctrl+C	Copy Project	Copy		Copy	Copy	Copy
Ctrl+D	Delete Project			Delete End		
Ctrl+E		Symbol Editor				
Ctrl+F2		Integrity Test		Toggle Bookmark		
Ctrl+G		Snap to Grid		Go to...		
Ctrl+H		Hierarchy Push			HDL Code Gen.	
Ctrl+I	Document Info.	Hierarchy Pop				
Ctrl+K		Annotate				
Ctrl+L	List of Libraries	Rotate Symbol				
Ctrl+M		Mirror Symbol				
Ctrl+N	New Project	New Sheet		New	New	New
Ctrl+O	Open Project	Open		Open	Open	Open...
Ctrl+P		Print		Print...	Print...	Print
Ctrl+S		Scratchpad		Save	Save	Save
Ctrl+T	Project Type	Printer Setup				Test Symbol
Ctrl+U		Deselect All				
Ctrl+V		Paste		Paste	Paste	Paste
Ctrl+W		Redraw Wires				
Ctrl+X		Cut		Cut	Cut	Cut
Ctrl+Y				Cut Line		
Ctrl+Z		Redo		Undo	Undo	
Del	Remove Doc.	Delete		Delete	Delete	Delete
Enter	Open Document					
F1	Help	Help	Help	Help	Help	Help
F2		Select & Drag	Cascade	Next Bookmark		
F3		Symbols	Tile	Find Next	Find Next	
F4		Draw Wires		Next Error	Next Error	
F5		Draw Buses				
F6		Draw Bus Taps				
F7		Query				
F8		Test Points	Short Step			
F9		Center	Long Step			
F10		Redraw			Redraw	Redraw
Pgdn		Full Page			Page	
PgUp		Previous Zoom			Previous	

Hyun Myung Electronics Co., Ltd.
Tel 02-3141-0147 / Fax 02-3141-0149
hmkor@hmelec.co.kr