HDL 설계 방법

Foundation series F1.4 버전에서는 Project manager의 HDL editor에 있는 analyze, synthesis 기능이 동작되지 않고 단지 editor 기능만 가지고 있다. 이는 F1.5 버전에서 해결될 것이고, F1.4에서는 FPGA Express를 이용하여 HDL 설계를 할 수 있다. 지금부터는 F1.4 버전에서 FPGA Express를 이용하여 VHDL 설계를 한 후 PLD로의 구현까지를 설명하기로 한다.

1 FPGA Express

FPGA Express는 FPGA logic-synthesis와 optimization tool이다. FPGA Express를 이용하여 VHDL code나 Verilog HDL code 또는 optimize가 되지 않은 FPGA netlist로부터 optimize된 FPGA netlist를 만들어 낼 수 있다. 다음 그림은 FPGA Express가 어떻게 FPGA design flow에 적용되는가를 나타낸다.

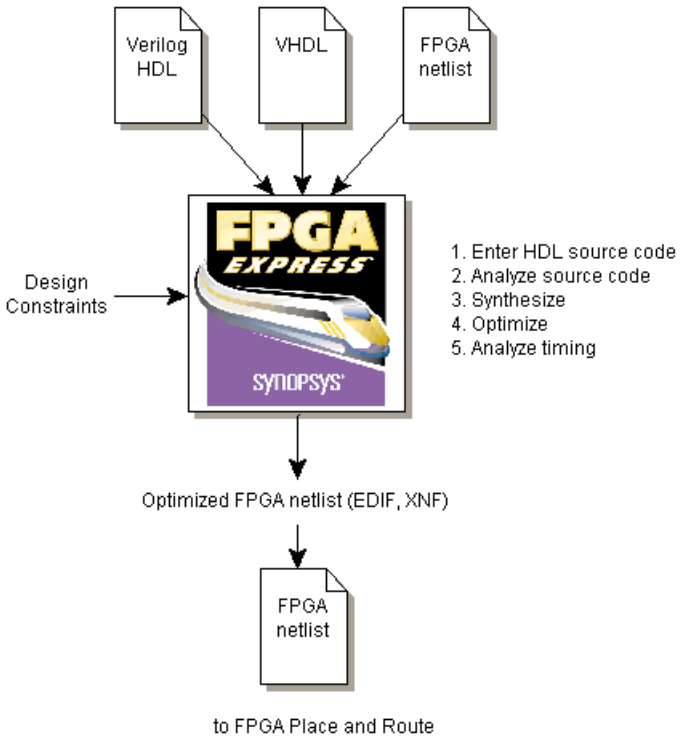


그림 1. FPGA express를 이용한 HDL 설계 흐름도

1.1 VHDL code와 Project의 생성 및 Analyze

FPGA Express를 이용하여 VHDL을 synthesis 하기 위해서는 우선 PC의 text editor를 이용하여 VHDL file을 생성시켜야 한다.

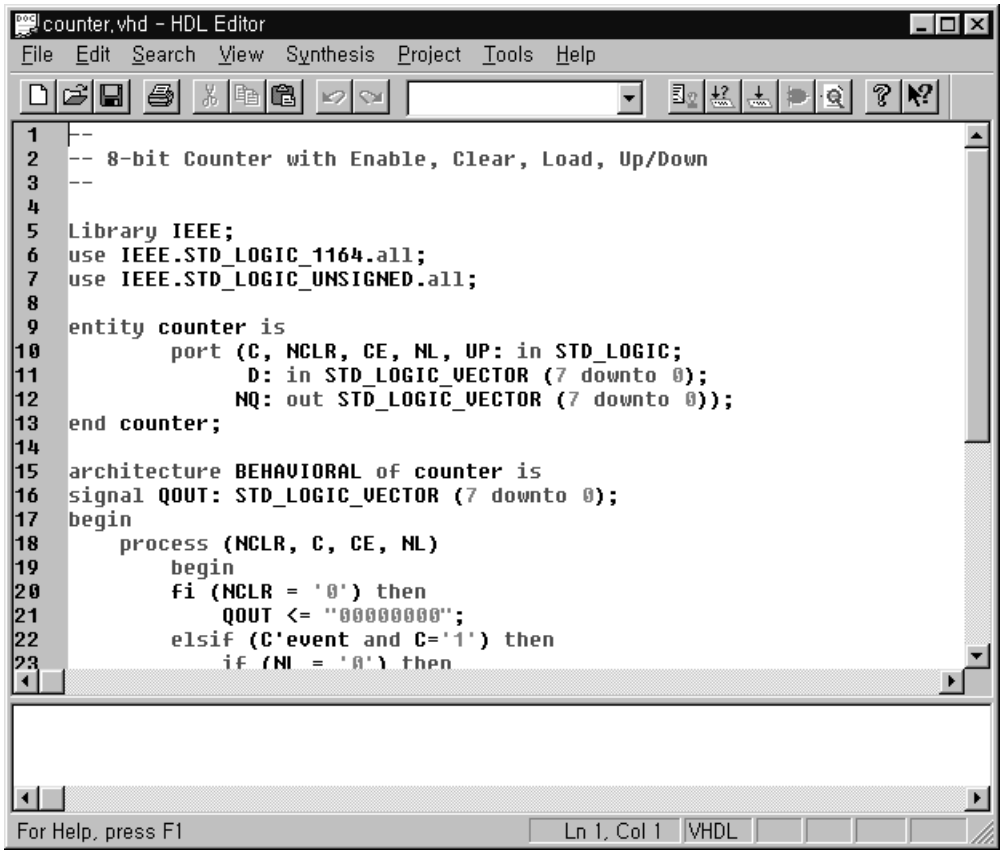


그림 2. Text editor를 이용한 VHDL 설계

Windows의 시작메뉴 중에서 Xilinx Foundation series에 있는 Foundation Express를 실행시키면 그림 3-3과 같이 FPGA Express가 실행된다. FPGA Express에서 최초로 할 일은 Project를 생성시키는 것이며, 이것은 File 메뉴 아래 New나 New project icon을 클릭하면 된다.

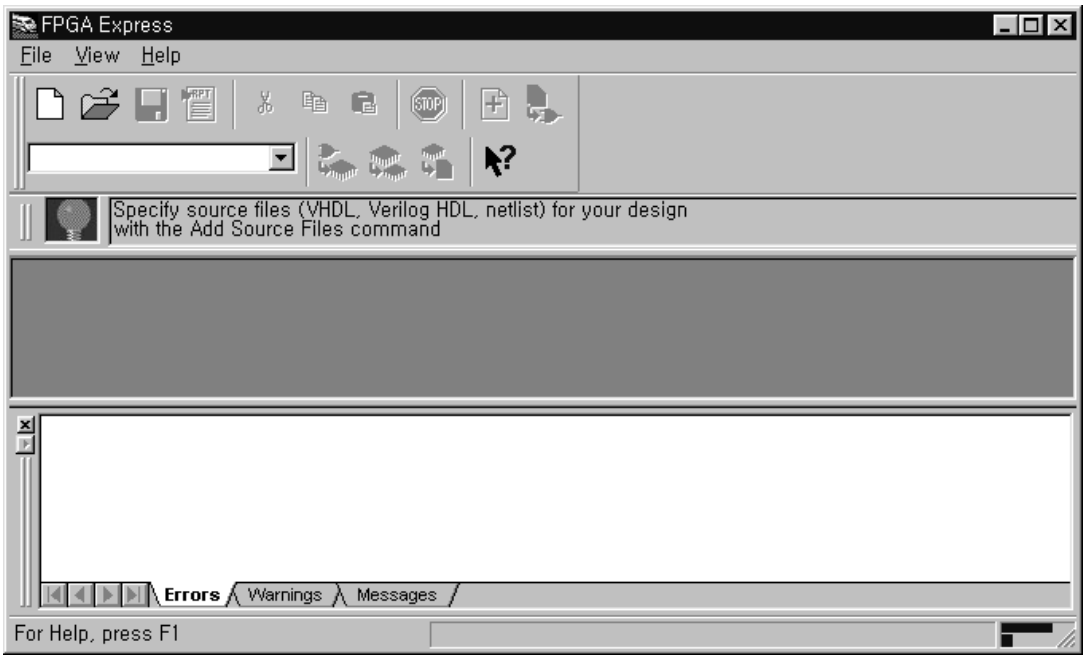


그림 3. FPGA Express의 초기화면

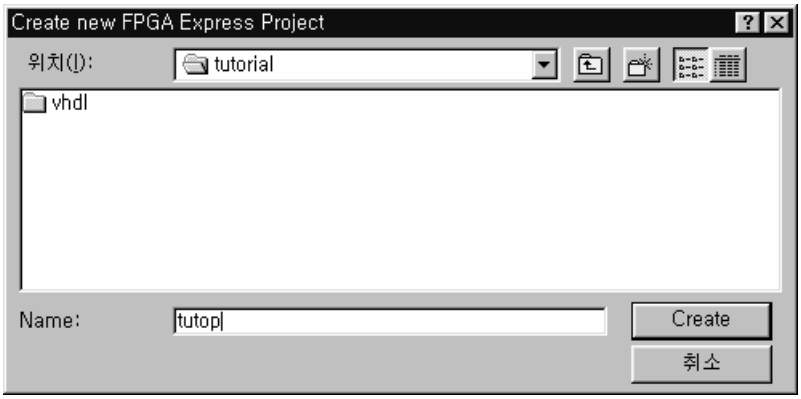


그림 4. Project 생성

그림 4와 같이 디렉토리를 선택한 후 Project 이름을 기입하고 Create를 클릭하면 바로 그림 5와 같이 Identify source창이 뜬다.

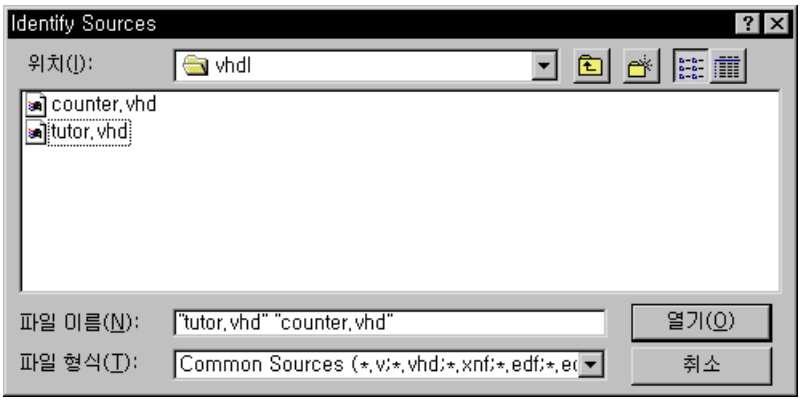


그림 5. Identify Source

Identify source는 synthesis를 하기 위해 미리 작성해 놓은 VHDL source file이나 optimization을 하기 위한 netlist file을 선택하기 위한 것이다. Identify source 창에서 VHDL file들을 선택하고 열기를 누르면, FPGA Express창은 Design sources 부분과 Chips부분으로 나눠지며, 선택한 VHDL source file을 자동으로 analysis한다. Design source 부분은 VHDL source file의 이름과 위치, 상태를 나타내며, Chips 부분은 device type과 같은 구현 정보를 나타낸다. Analysis는 VHDL에 대한 syntax error나 warning을 검사한다. 이렇게 project를 만들고, source file을 identify 하면 그림 6과 같이 work 디렉토리가 만들어지고, VHDL analysis를 실행한 결과가 나타난다.

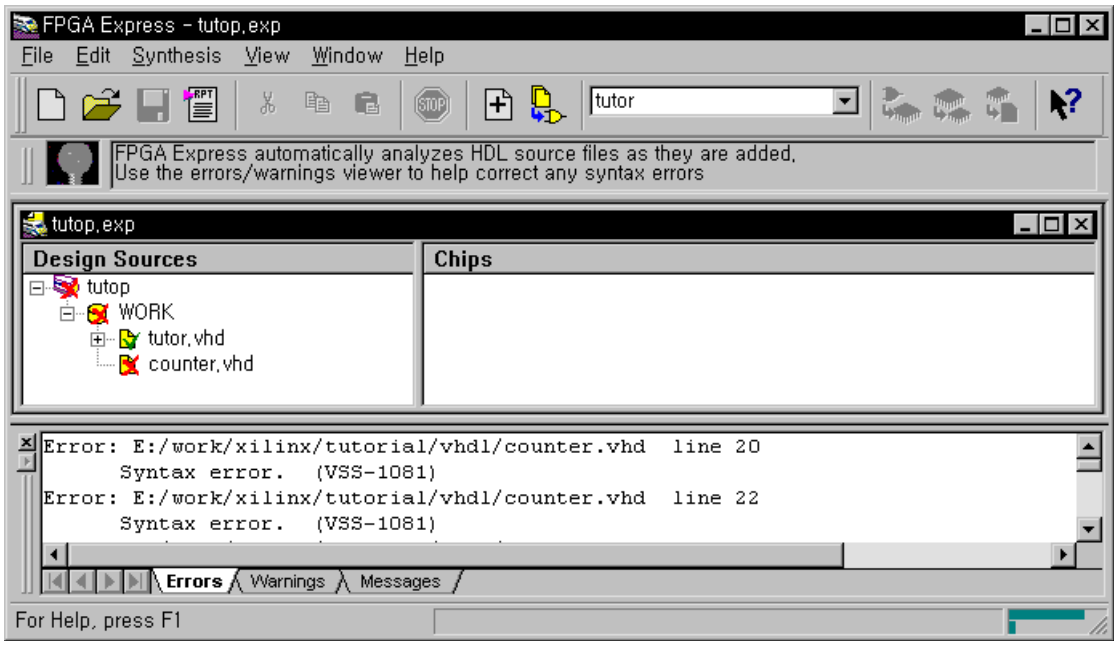


그림 6. Identify 후 analysis 결과

Design sources창에서 빨간색으로 가위표시()가 된 것은 Error가 발생한 것이고, 녹색으로 체크()된 것은 에러없이 analyze된 file 임을 가리킨다. Error라고 출력된 메시지가 있는 곳을 두 번 마우스로 클릭하거나, 마우스 오른쪽 버튼을 누른 후 나타나는 Edit File 메뉴를 선택하면 그림 7과 같이 error가 발생한 file이 편집 가능하게 된다.

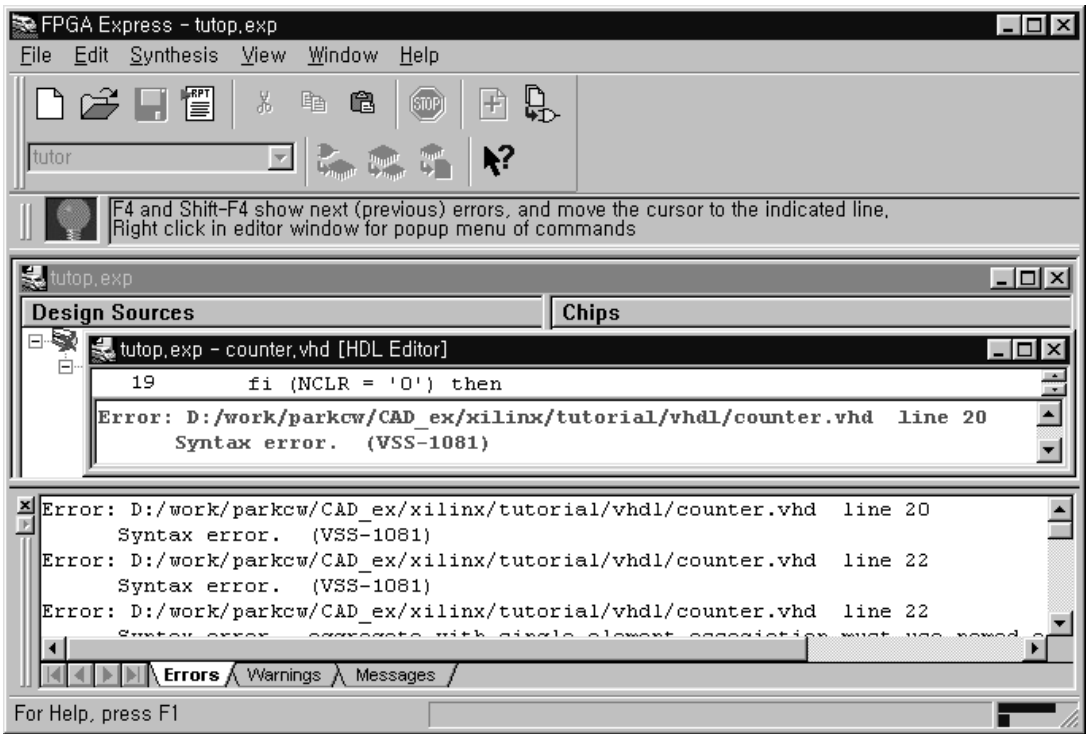


그림 7. Error 수정을 위한 File 편집

Error를 수정하고 save를 한 후 Design Sources 부분에서 오른쪽 마우스 버튼을 눌러 Update File 메뉴를 선택하던지 Update icon( )을 선택하면 다시 analysis가 수행된다. 이렇게 해서 error가 제거된 결과가 그림 8처럼 나타난다. 이렇게 project를 만들고, identify를 하여 analysis를 수행하면, design을 synthesis할 준비가 된 것이다.

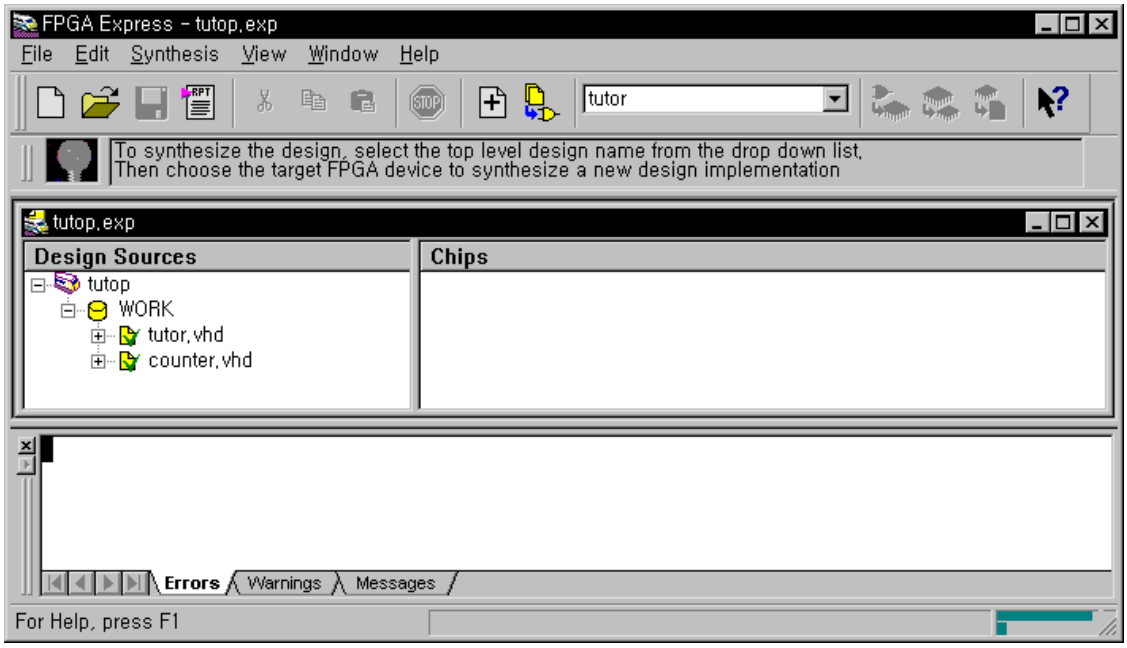


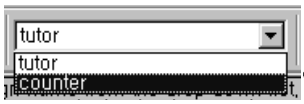
그림 8. Error를 수정 후 Update File을 수행한 결과

1.2 Synthesis

Design을 최적화하기 이전에 우선 synthesis를 해야하며, 이를 위한 방법은 다음과 같다.

가. Top-level design의 지정

Synthesis를 하기 위해서는 우선 top-level entity(VHDL), module(Verilog), 또는 schematic design을 지정해야 하며, FPGA Express는 이 정보를 사용하여 design hierarchy와 interconnections을 구축한다. Top-level design을 지정하는 방법에는 세 가지가 있다.

○ Tool bar에 있는 

에서 top-level design을 선택한다.

○ Design sources창에서 source file을 double-click하면 design entity들이 확장되어 나타나는데, 여기서 top-level design에 해당하는 icon을 선택하고, 오른쪽 마우스 버튼을 누른 후 Create Implementation을 선택한다.

○ Icon bar에 있는  을 선택한다.

위와 같이 하면 그림 9와 같이 Create Implementation dialog box가 나타난다.

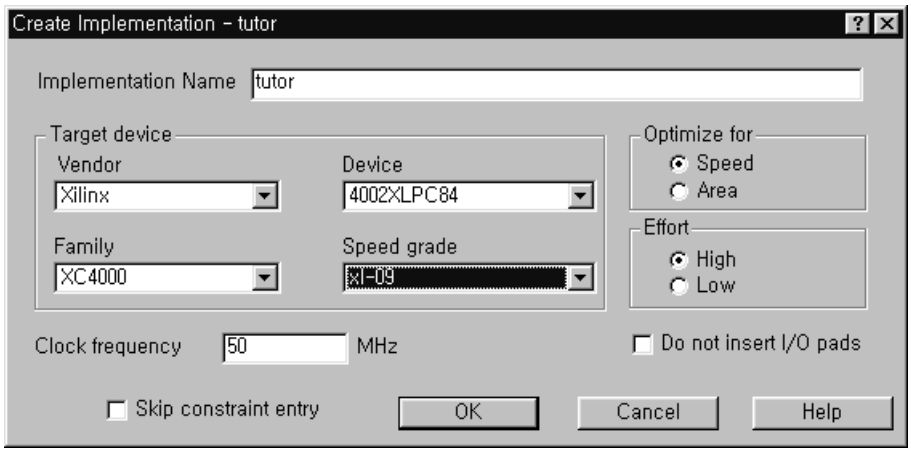


그림 9. Create Implementation dialog box

나. Target architecture와 clock frequency의 지정

Create Implementation dialog box에서 target architecture와 clock frequency를 지정해야만 design implementation과 synthesize logic을 만들 수 있다. Target architecture를 지정하기 위해서는 design을 위한 vendor와 family를 선택한 후 옵션으로 device type과 speed를 선택하면 된다. Target clock frequency는 MHz 단위로 수치를 기입하면 되며, 이는 design에 있는 모든 clock들의 default 값으로 사용된다. 이 값은 synthesize 후에 design constraint table에서 변경할 수 있다.

다. Design implementation의 생성

Create Implementation dialog box에서 target architecture와 clock frequency를 지정하면, FPGA Express는 지정된 architecture에 맞는 logic을 synthesis할 준비가 된 것이다. 여기서 옵션으로 어떤 CPU effort를 가지고 optimize를 할 것인가를 지정해 줄 수 있다. 이는 설계자가 자신이 목표로 하는 상황과 결과에 따라 판단하여 결정해야 한다. Effort가 high로 선택되면, Synthesis시 속도가 오래 걸리지만 가장 좋은 결과를 기대할 수 있으며, low가 선택되면, 그 반대가 된다. 그리고 Optimize for를 area로 선택하면, 차지하는 면적이 작아져서 더 많은 CLB를 채울 수 있지만 delay가 길어지며, speed로 선택하면 delay가 적어지고 면적은 더 차지하게 된다. 하지만 이것들은 모두 회로에 따라 최적화할 수 있는 한계가 있기 때문에 설계자는 optimize 시킨 후 report file을 참고하여 자신의 목적에 맞는 가장 효율적인 방법을 찾아야 한다. 그리고 현재 design이 앞으로 sub block으로 사용될 예정이면, Do not insert I/O pads를 마크한다. Create Implementation dialog box에서 위와 같이 모든 설정을 하고 OK를 click하면 design이 elaborate되면서 그림 10과 같이 Chips 창에 target device가 나타나며, 최적화 되지 않은 중간 design implementation을 생성하게 된다.

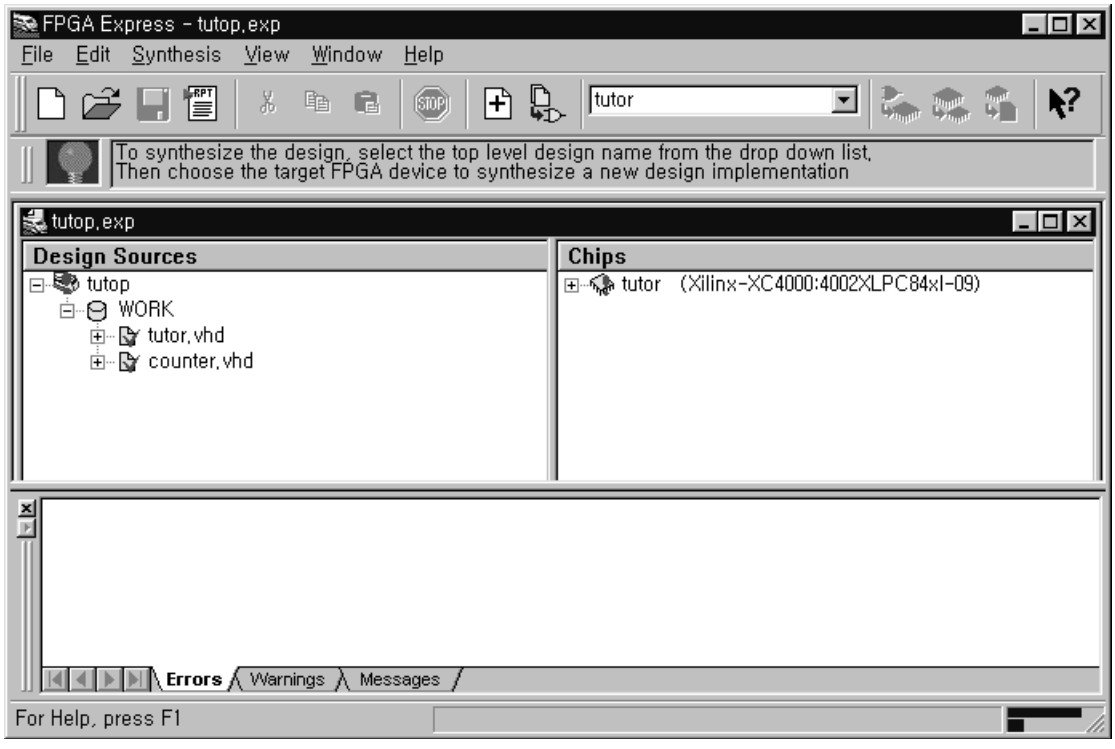


그림 10 Create Implementation의 실행 결과

1.3 Design constraints의 입력

Design을 target device로 optimize하기 전에, 여러분이 constraint table의 값을 어떻게 입력해 주는가에 따라 place & route의 결과를 향상시킬 수 있다. FPGA Express의 Chips 창에서 design implementation을 선택한 후 마우스 오른쪽 버튼을 click 한 후 Edit Constraint를 실행한다. 그러면 Clock, Path, Port, Module, Xilinx option에 대해서 constraint를 기입할 수 있게 된다.

가. Clock

Edit Constraint에서 Clocks를 선택하면 그림 11과 같이 design에서 default clock과 주기를 가진 signal의 이름이 나타나며 값을 설정할 수 있다. Default는 Create Implementation에서 지정한 clock frequency 값이 사용된다.

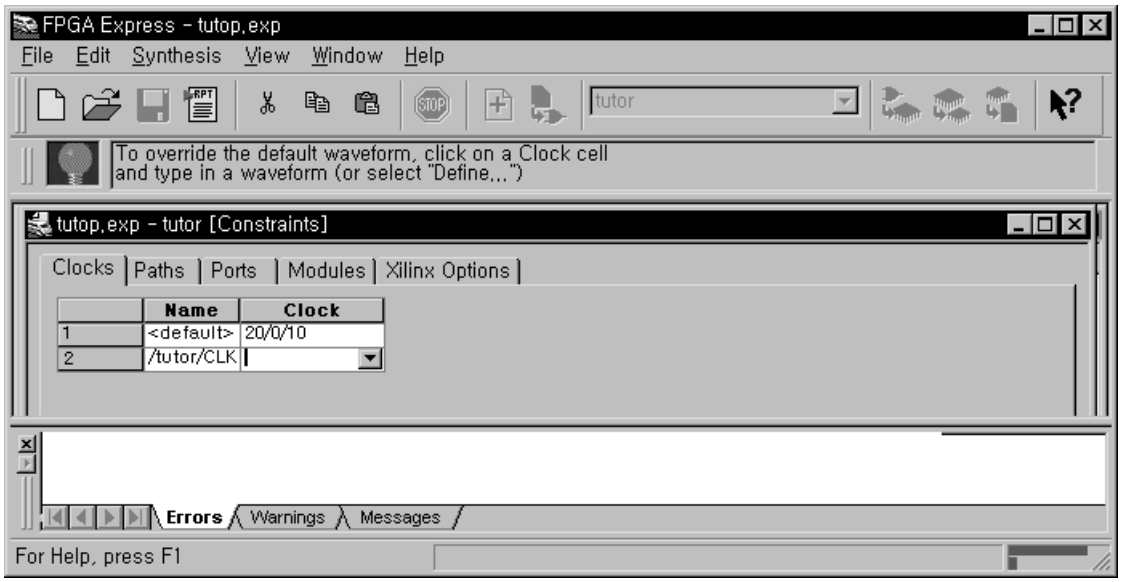
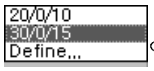


그림 11. Edit constraint - Clocks

그림 11에서 20/0/10이란 20ns 주기를 가지고 rise time은 0ns, fall time은 10ns를 의미한다. Clock column에 있는 

에서 Define을 선택하면 그림 12와 같이 값을 다시 설정할 수 있다

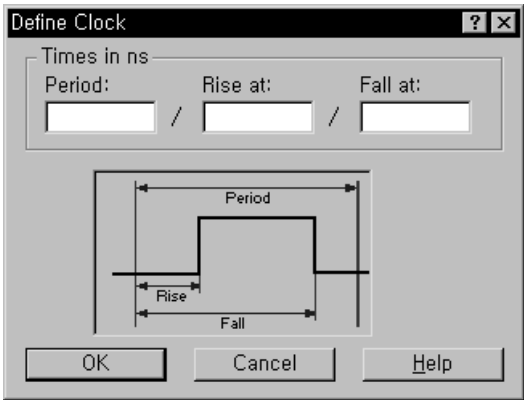


그림 12. Define clock

나. Path

Edit Constraint에서 Path를 선택하면 그림 13과 같이 starting group(From column), end group(To column) 그리고 default clock으로 지정된 path의 최대 delay(Required delay column)이 나타난다.

○ From : Path의 starting group이며, 모든 input과 주기를 가진 signal에 의해 clock이 지정된 모든 edge를 가진 sequential elements 이다.

○ To : Path의 end group이며, 모든 output과 주기를 가진 signal에 의해 clock이 지정된 모든 edge를 가진 sequential elements 이다.

○ Req. Delay : Path의 최대 delay 값이며, end group의 active edge와 starting group의 active edge 간의 delay이다. Path group의 값을 변경하려면, 이 칸을 마우스로 선택하고 값을 입력하면 된다.

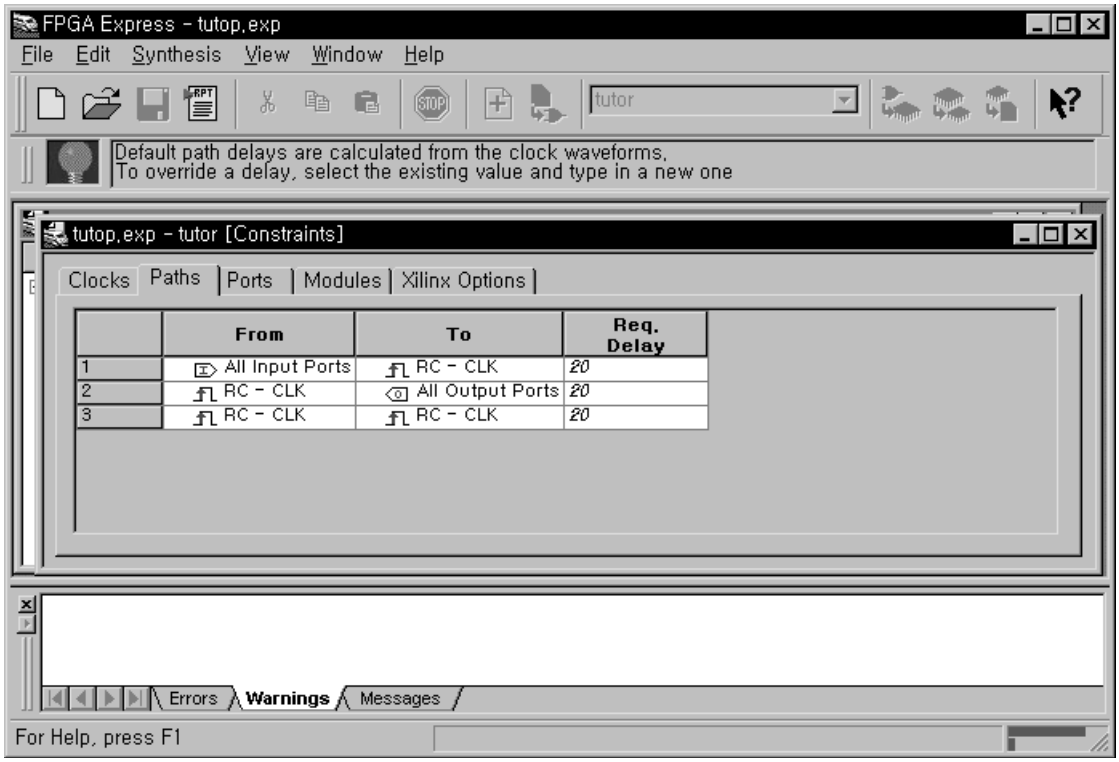


그림 13. Edit constraint - Paths

Starting group이나 end group을 선택하고 마우스 오른쪽 버튼을 누르면 선택된 path에 대한 sub path의 delay를 지정하거나 editing 할 수 있다.

다. Port

Edit Constraint에서 Ports를 선택하면 그림 14, 15와 같이 나타난다. Name과 direction은 design에 있는 Port의 이름과 mode이다. 그리고 각 column의 의미는 아래와 같다.

○ Input Delay(ns) : Input이나 inout port로부터 flip-flop이나 latch같은 sequential elements까지의 최대 delay 값이다. 변경하고자 하는 field를 선택하고 define을 선택하면 값을 설정할 수 있다.

○ Output Delay(ns) : output이나 inout port로부터 flip-flop이나 latch같은 sequential elements까지의 최대 delay 값이다. 변경하고자 하는 field를 선택하고 define을 선택하면 값을 설정할 수 있다.

○ Global Buffer 

: Port에 global buffer를 지정하거나 자동 또는 사용 안함을 선택할 수 있다. 앞서 설명한 Create Implementation에서 Do not insert I/O pads를 선택했다면, 여기서 global buffer를 선택할 수 없다. Global buffer는 clock이나 fanout이 많은 net를 drive할 경우에 사용하며, 이를 사용하면 회로가 빨라지고, clock skew가 최소화되고, routing이 더 쉽게 된다. Global buffer는 모든 net를 drive할 수 있는 것과 clock만 drive할 수 있는 것이 있으며 선택한 architecture에 따라 숫자도 제한되어 있다.

○ Pad Direction : Port를 three-state(bidirectional)로 지정할 경우에 사용한다.

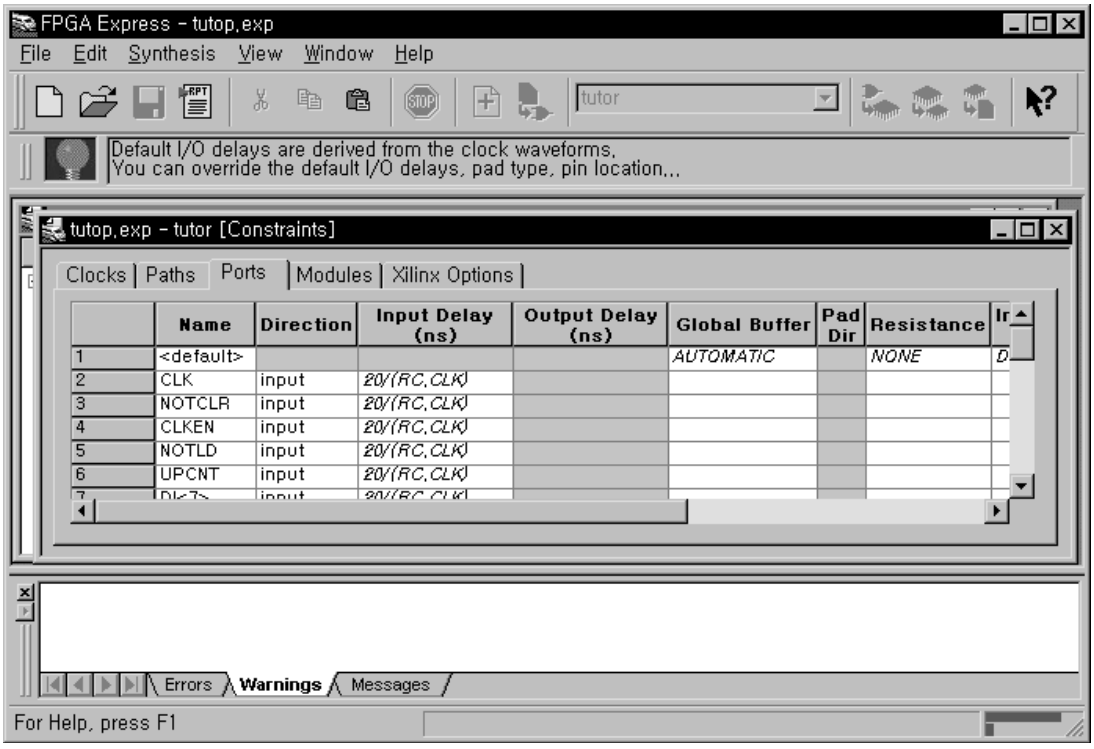


그림 14. Edit constraint - Ports(1)

○ Resistance <default> NONE PULLUP PULLDOWN : Port에 pullup이나 pulldown 저항을 연결할 경우에 사용한다.

○ Input Register Delay : Input Register 내부의 input delay를 최적화 시킬 것인지 아닌지를 선택한다. Delay를 선택하면, input transition에 대한 hold time을 감소시킬 수 있다.

○ Use I/O Register : I/O register pad를 연결하기에 적절한 port를 선택하여 true를 선택한다.

○ Slew Rate : Output이나 bidirectional Port에 대하여 slew rate를 빠르게 할 것인지 느리게 할 것(fast/slow)인지를 선택한다.

○ Pad Location : Port에 대한 pad의 위치(pin 번호)를 정의한다. 이 field는 Create Implementation에서 "Do not insert I/O pads"를 선택하지 말아야만 정의가 가능하다.

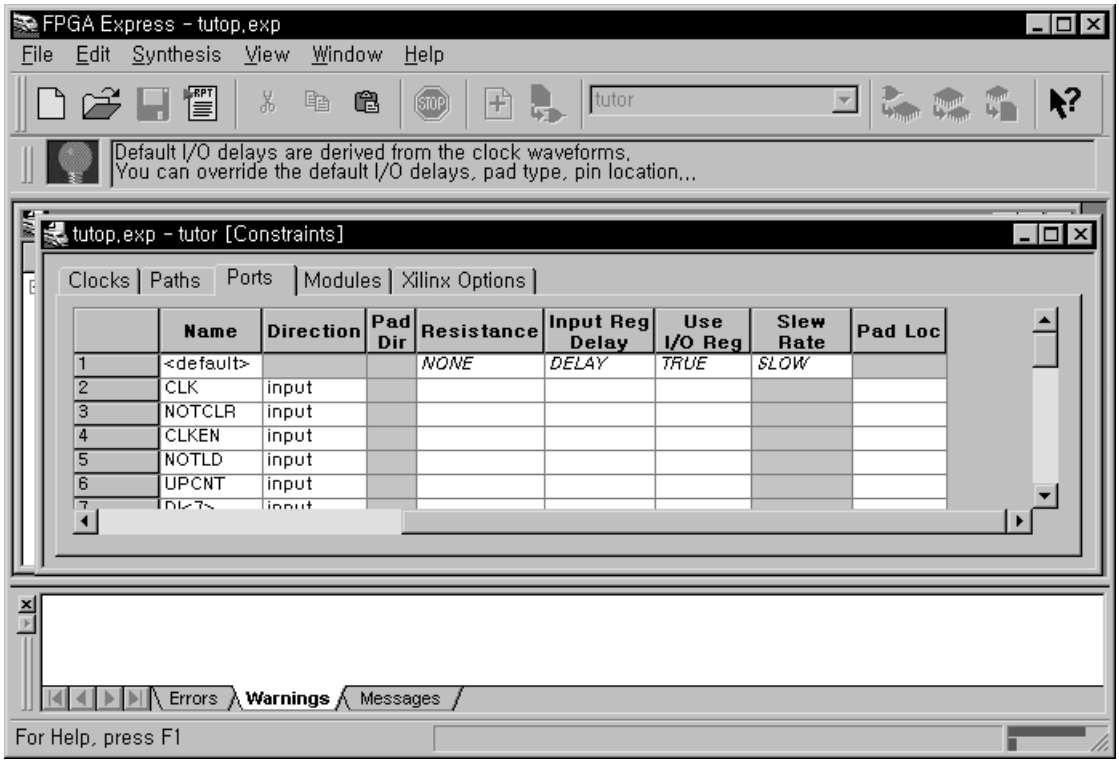


그림 15. Edit constraint - Ports(2)

라. Module

Edit Constraint에서 Modules를 선택하면 그림 16과 같이 parent module 아래 submodule과 이름이 나타나며 각 column의 의미는 다음과 같다.

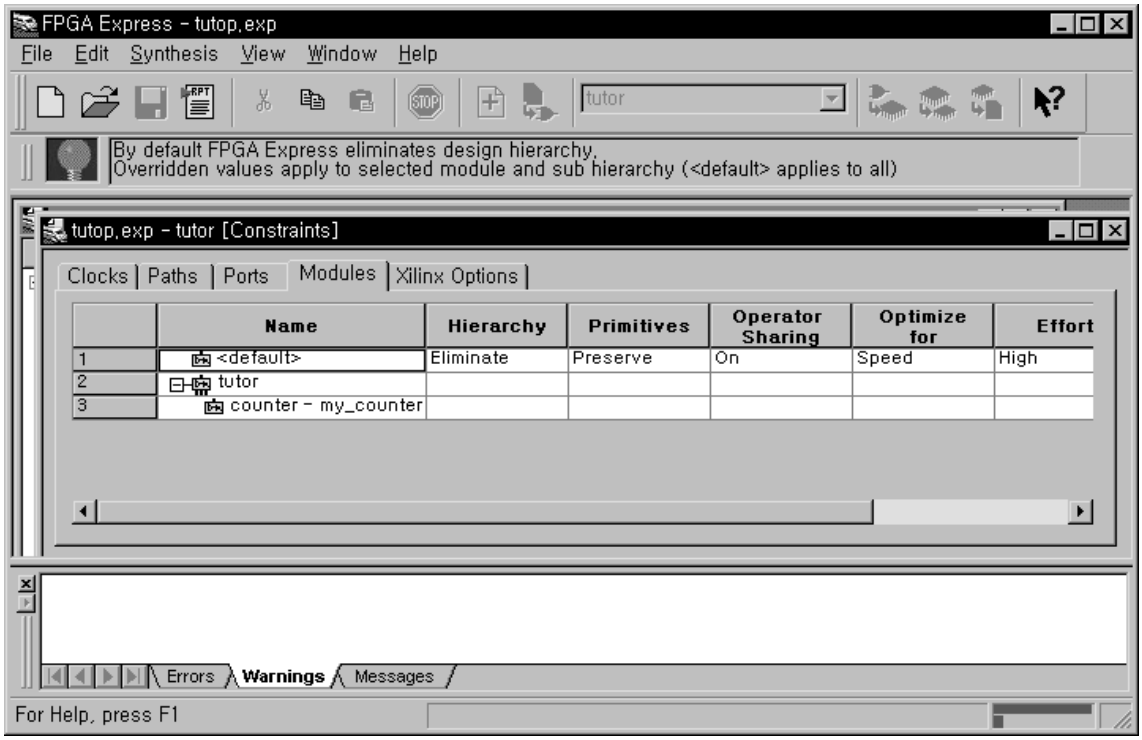


그림 16. Edit constraint - Modules

○ Hierarchy : Optimization 하는 동안 module의 hierarchy를 유지할 것인지, 제거할 것인지를 선택한다. Preserve를 선택하면, module의 경계가 유지되어 다른 logic과 독립적으로 optimize가 된다. Eliminate를 선택하면 선택한 module의 경계가 제거되며, 다른 design과 함께 optimize가 된다.

○ Primitives : Primitive logic을 module과 함께 optimize할 것인지를 선택한다. Preserve를 선택하면, module에 instantiate된 모든 primitive logic의 경계가 유지되는 반면에, Optimize를 선택하면, module에 있는 모든 primitive cell의 경계가 제거되며 module의 logic과 함께 primitive logic도 optimize 된다. VHDL에서 technology vendor의 library를 component declaration과 instance declaration하는 것만으로 primitive logic을 사용할 수 있다.

○ Operator Sharing : Module에 있는 operator를 공유할 것인가를 선택한다. Off를 선택하면, 각각의 hardware를 사용하여 각 operator를 optimize시키고, On을 선택하면 design의 area와 speed를 향상시키기 위하여 같은 hardware 자원을 사용하여 operator를 optimize 시킨다.

○ Optimize for : Speed를 위주로 더 빠른 회로로 optimize 할 것인지, area를 위주로 더 작은 회로로 optimize 할 것인지를 선택한다.

○ Effort : 이것은 area에 약간의 영향을 주면서 최고의 timing optimization을 하는 것과 관련이 있다. Low를 선택하면, 가장 적은 시간으로 compile을 하기 때문에 area나 timing을 만족시키지 않을 수 있는 반면에 high를 선택하면, compile 시간이 길고 가장 좋은 design을 생성할 수 있다.

마. Xilinx option

Edit Constraint에서 Xilinx options를 선택하면 그림 17과 같이 나타난다. FPGA Express는 Global Set/Reset(GSR) 신호를 자동으로 찾아서 optimize를 한다. Unlinked cell은 GSR 신호에 의한 비동기 set이나 reset이 없는 sequential elements인데, 원칙적으론 unlinked cell이 하나도 없어야만 GSR mapping을 할 수 있다. 그러나 Ignore unlinked cells during GSR mapping을 마크하면, unlinked cell을 GSR 신호에 의한 set이나 reset이 아닌 sequential elements로 가정하고 GSR mapping을 한다. 이 옵션이 마크되지 않은 상태에서 unlinked cell이 하나라도 포함되어 있으면, GSR 자원을 사용할 수 없고 warning을 발생한다. Optimized for place and route는 place and route tool을 선택하는 것이다. 여기에는 XACT와 M1이 있다. 물론 M1은 synopsys의 synthesis core를 이용하여 만들었으므로 XACT보다 성능이 좋다. 이렇게 이전 버전과의 차이점에 대한 것은 S/W가 발표될 때 나온 설명서를 참조하면 알 수 있다.

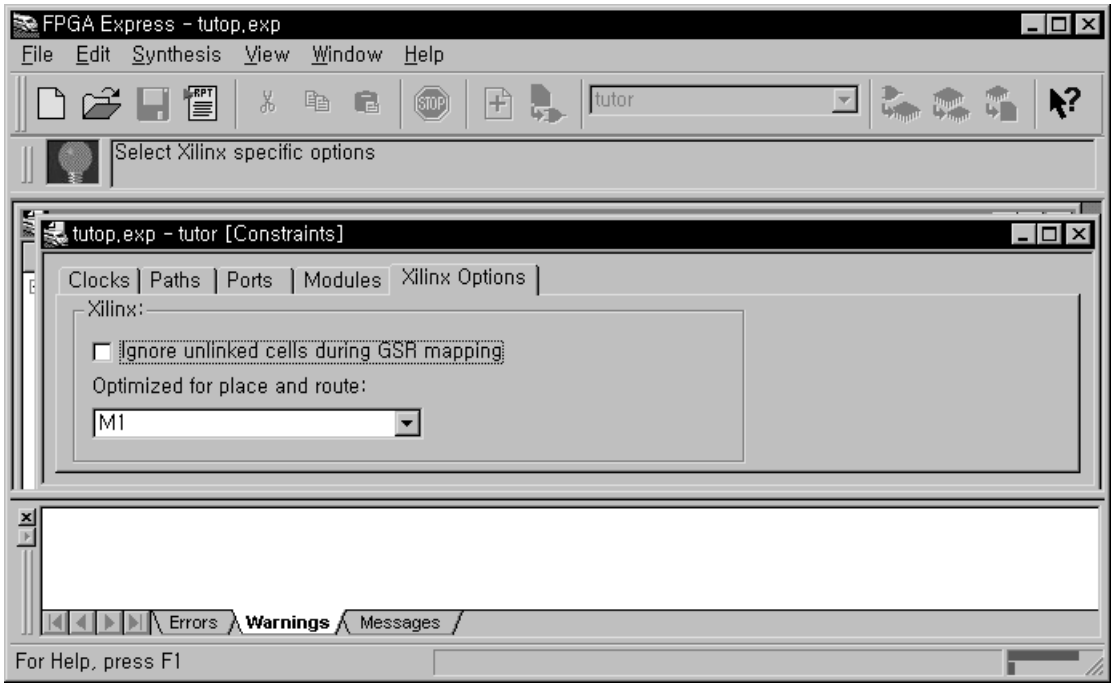


그림 17 Edit constraint - Xilinx options

1.4 Optimization

Constraint에 대한 입력을 마친 후에 design을 optimization을 하는 방법은 다음과 같다. Chip 창에 있는 design implementation을 선택한 후 오른쪽 마우스를 눌러 Optimize Chip을 선택하거나 tool bar에 있는  를 클릭하면 FPGA Express는 Optimization을 수행한다. Design implementation을 optimize할 때, FPGA Express는 요구사항들을 만족시키기 위해서 실제 timing과 area를 분석하여 최적화하며, optimization이 완료한 후의 FPGA Express 창의 모습은 그림 18과 같다.

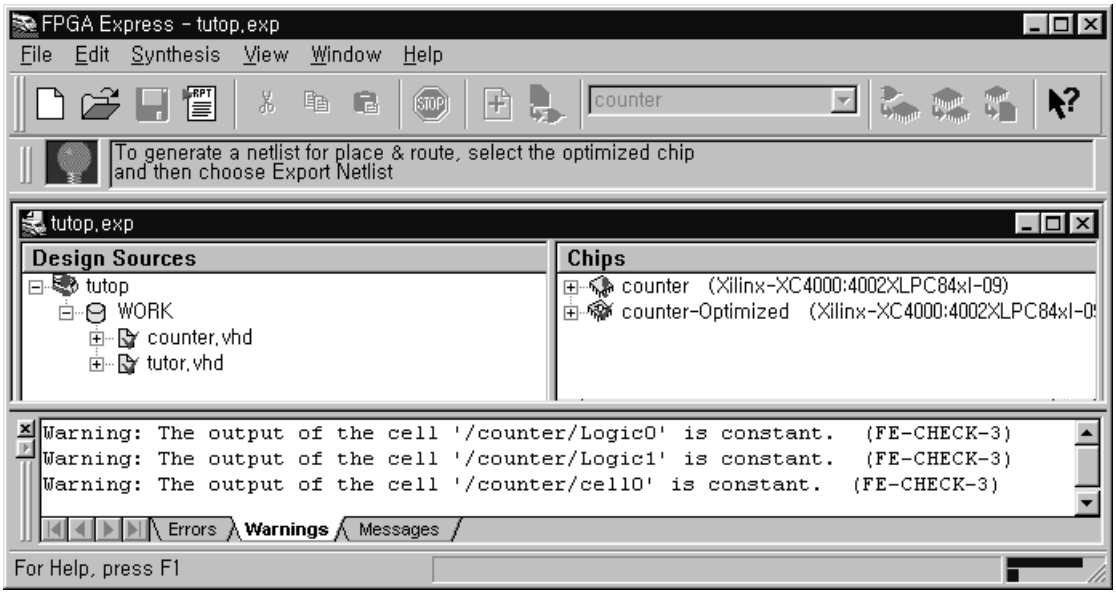


그림 18. Optimization 완료의 모습

1.5 Timing analyze

그림 19와 같이 constraint table상에 나타난 optimization의 결과를 검토하고 timing 정보를 분석하는 것에 의해 회로의 성능을 결정할 수 있다. Chips 창에서 optimize된 implementation을 선택한 후 오른쪽 마우스 버튼을 눌러 View Results를 선택한다. 그러면 Edit Constraint 창과 같은 항목들이 나타나는데, 여기서는 앞에서 설정한 값과 FPGA Express에서 계산한 값이 동시에 나타나서 optimization 결과를 판단할 수 있다.

○ Clocks : 그림 19와 같이 optimization 이전에 주어진 clock의 최대 주파수 값과 optimization 이후의 값이 나타난다. Clock 주파수의 violation은 빨간색으로 나타나게 된다.

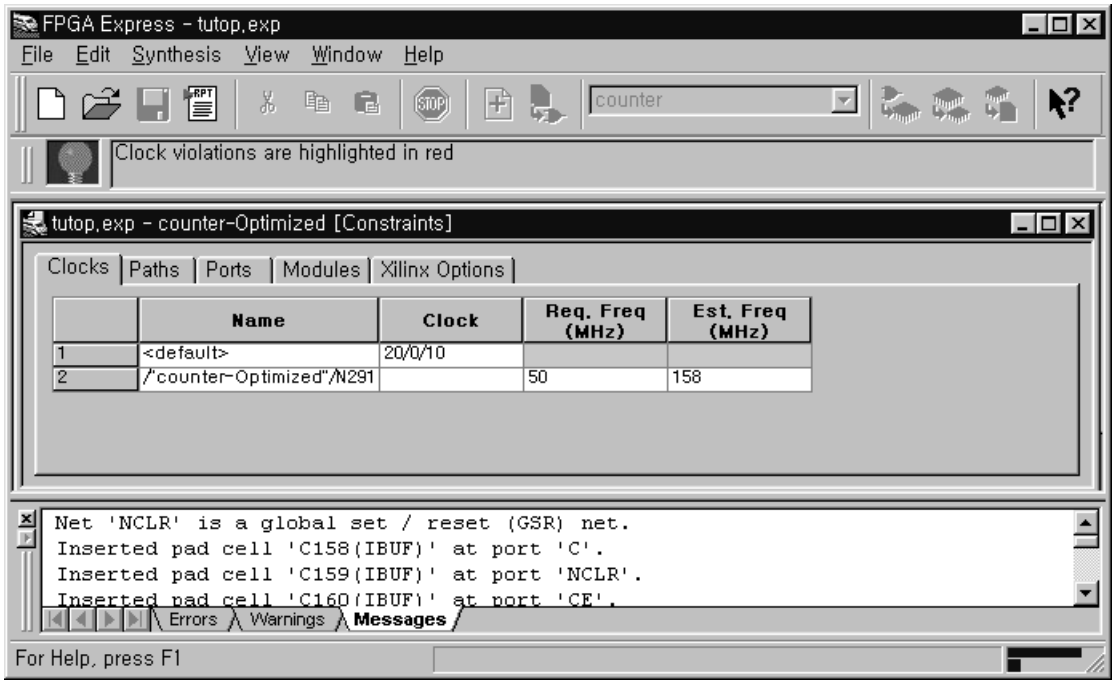


그림 19. View Results - Clocks

○ Paths : 그림 20과 같이 optimization 이전에 주어진 path의 delay값과 optimization 이후의 값이 나타난다. 각 Path group을 선택하면 delay와 fanout의 더 자세한 값을 알 수 있다.

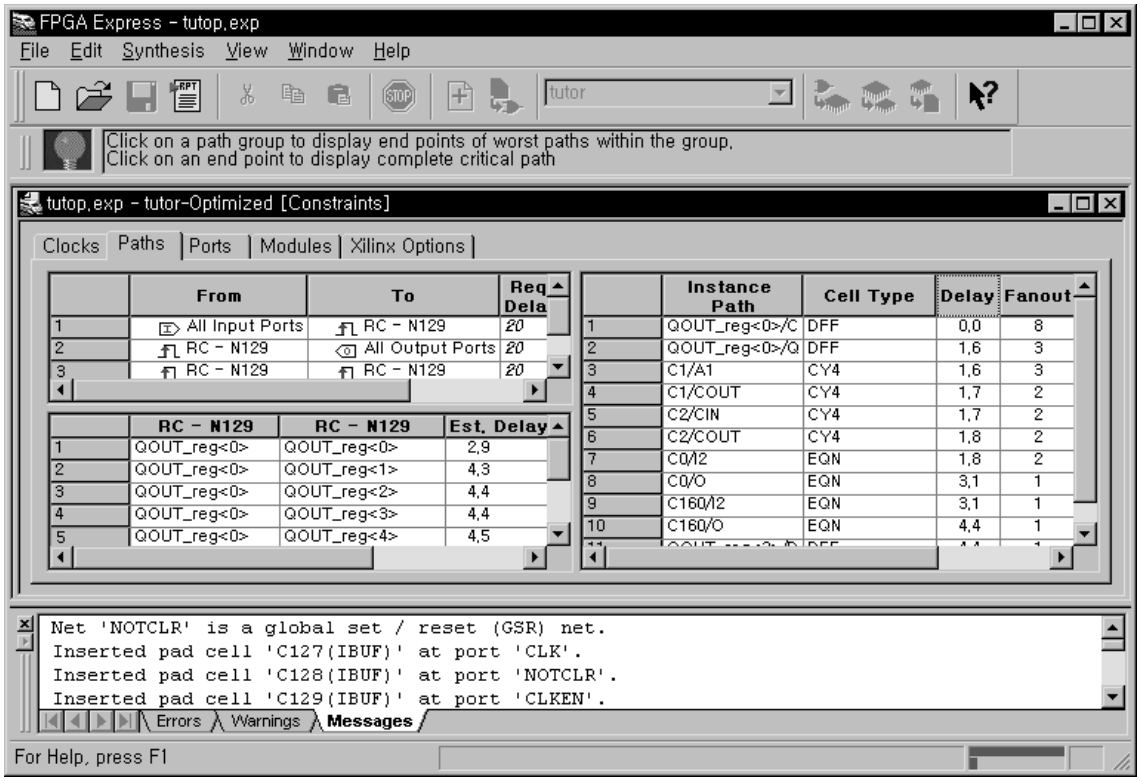


그림 20. View Results - Paths

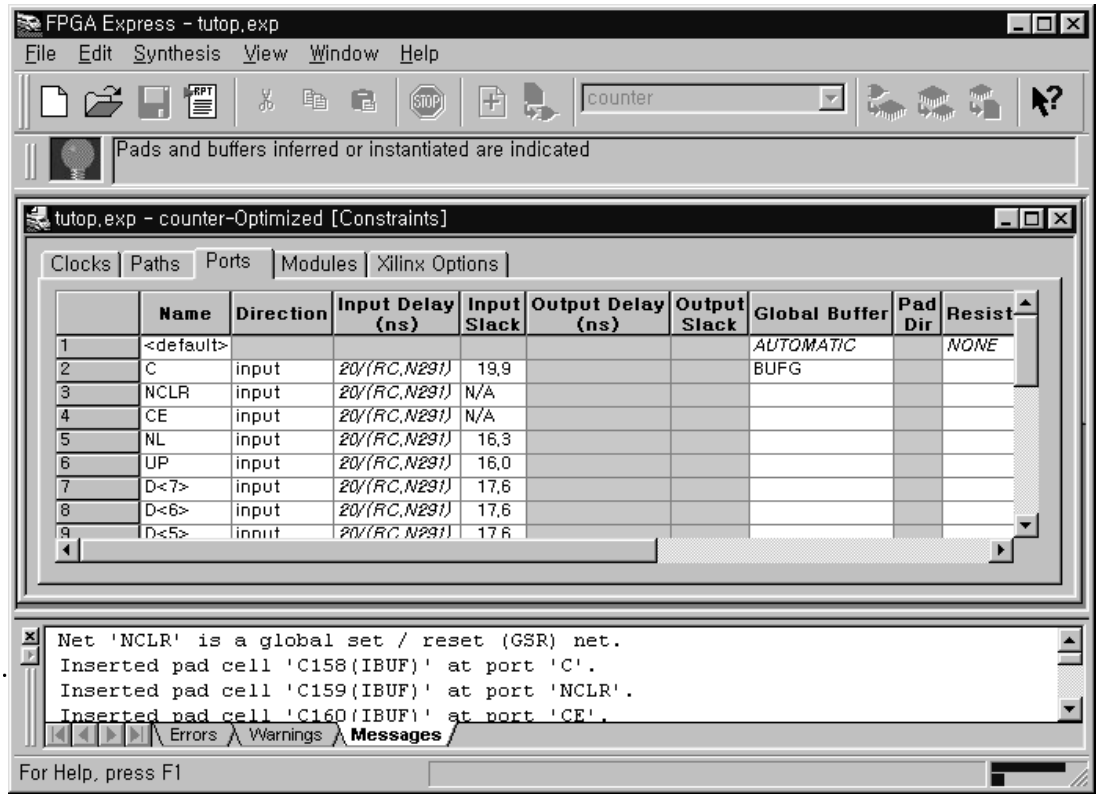


그림 21. View Results - Ports(1)

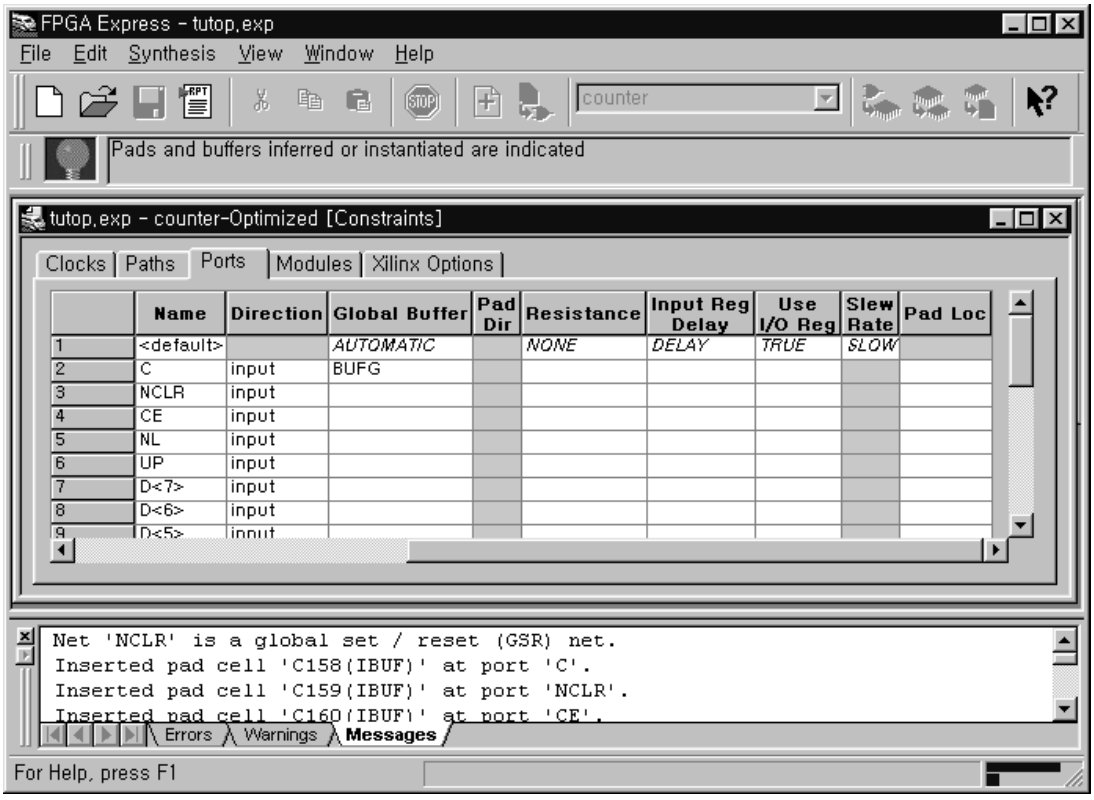


그림 22. View Results - Ports(2)

○ Ports : Optimization 이후의 port의 input과 output delay에 대한 정보가 그림 21, 22와 같이 나타난다. 각 port의 input arrival time에 대한 slack과 output delay에 대한 slack등의 값을 알 수 있다.

○ Modules : 사용된 device 자원에 대한 정보가 그림 23과 같이 나타난다. Area column에서 detail을 선택하면 해당 module에서 어떤 cell이 몇 개 사용되었는지를 자세히 알 수 있다.

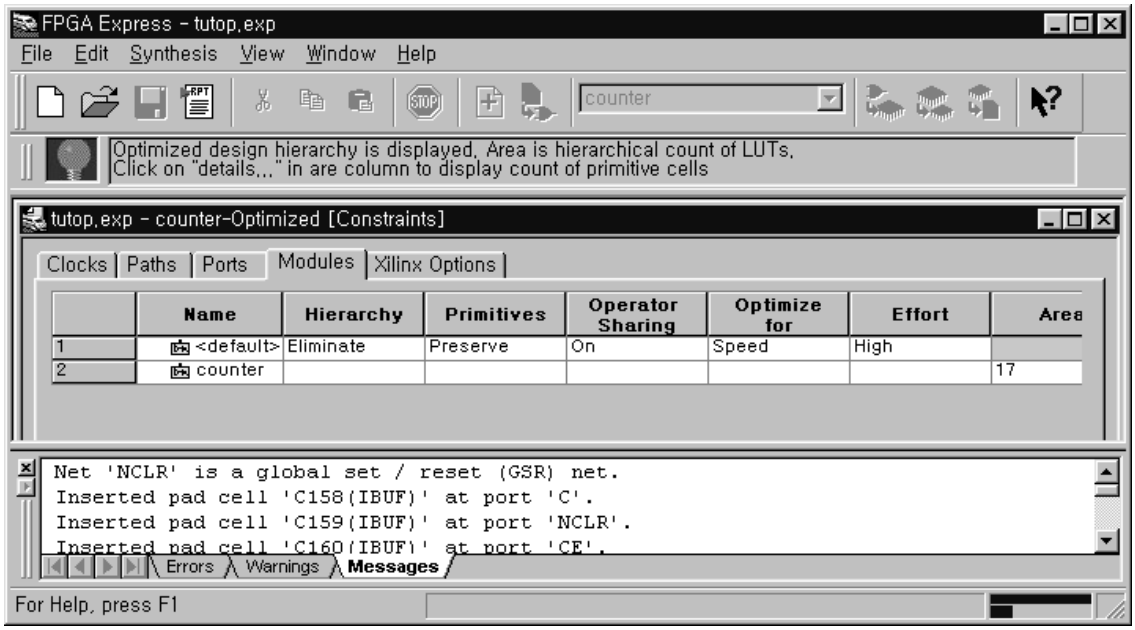


그림 23. View Results - Modules

○ Xilinx Options : 선택한 option에 대한 정보를 그림 24와 같이 알 수 있다.

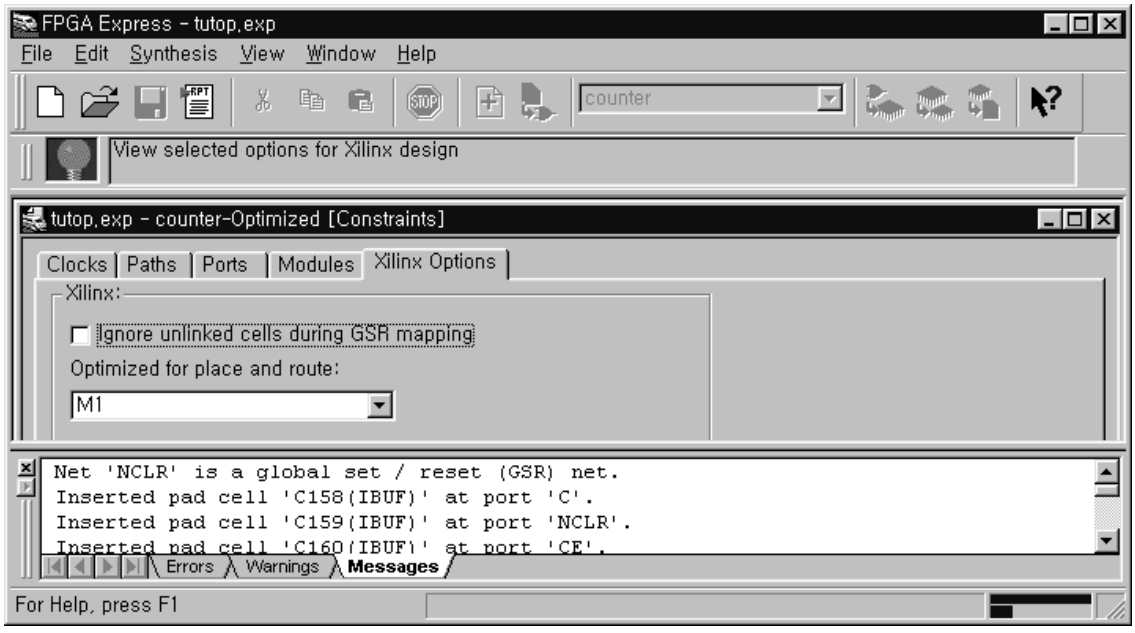


그림 24 View Results - Xilinx Options

1.6 Optimize된 FPGA netlist와 report의 생성

Design implementation이 optimize된 후 여러분은 FPGA netlist file을 생성할 수 있다. 또한 project, library, file, constraint등에 관한 정보를 참조하기 위해 report file을 생성할 수 있다.

가. Netlist file의 생성

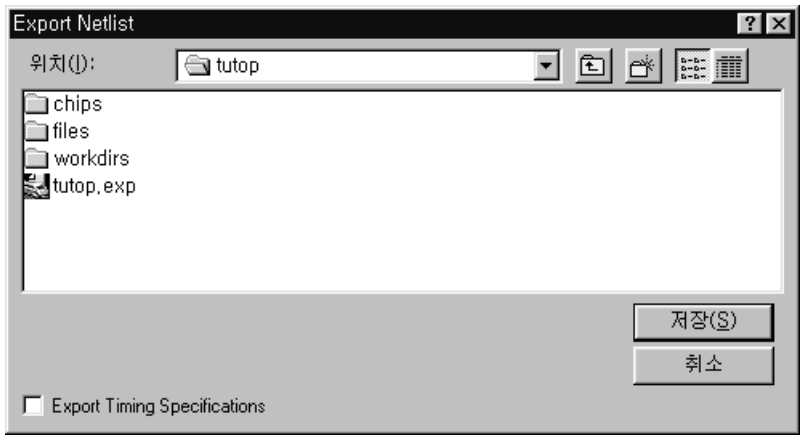


그림 25. Export Netlist

Chips창에서 Optimize된 design implementation을 선택한 후 마우스 오른쪽 버튼을 눌러서 Export Netlist를 선택하거나 tool bar에서  를 선택하면 그림 25와 같은 창이 나타난다. Export Netlist 창에서 netlist file을 생성할 folder를 선택하고, save를 누르면 xnf netlist file이 생성된다. 또한 netlist file에 timing constraint 정보를 포함시키고자 한다면 Export Timing Specification 버튼에 마크한다.

나. Report file의 생성

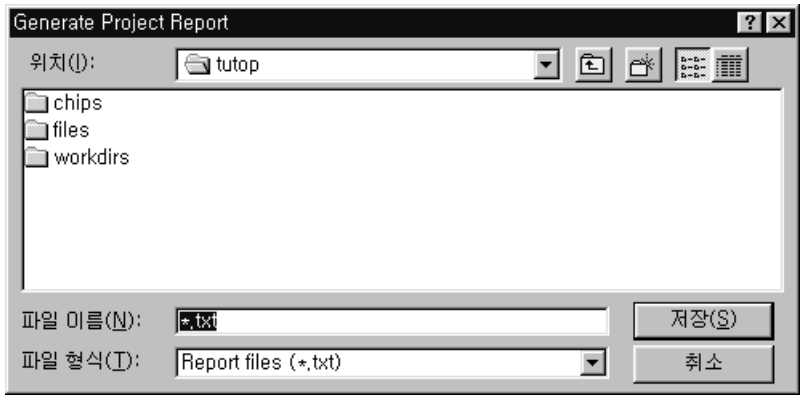


그림 26. Report file의 생성

Project 창에서 project, library, design, chip 중 하나를 선택하고 마우스 오른쪽 버튼을 누르고 해당 항목에 대한 report menu가 나타나며, 이를 선택하거나 tool bar에 있는  을 선택하면 그림 26과 같은 창이 나타난다. Generate Project Report 창에서 저장할 folder와 이름을 정해주면 project, library, design, chip, constraint에 대한 모든 정보가 담긴 text file을 생성할 수 있다.

이렇게 FPGA Express를 사용하여 VHDL 설계를 synthesis한 후 netlist와 report file을 생성할 수 있다.

2. Schematic editor에서의 사용

VHDL을 합성하여 생성한 netlist를 Schematic design에서 사용하기 위해서는 1.6 의 "가"항에서 생성한 xnf file 이용하여 symbol로 만들어야 한다. Xilinx Foundation series의 Project manager에서 Schematic editor(그림 27)를 선택한 후 Hierarchy 메뉴의 Create Macro Symbol from Netlist를 실행하면 그림 28과 같은 창이 나타난다.

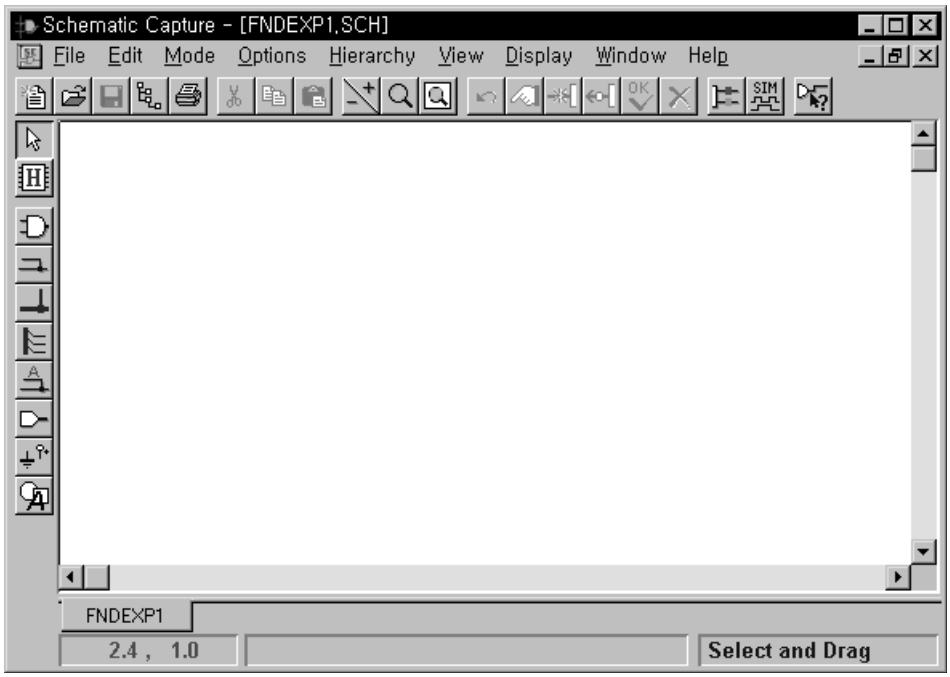


그림 27. Project manager의 schematic editor의 화면

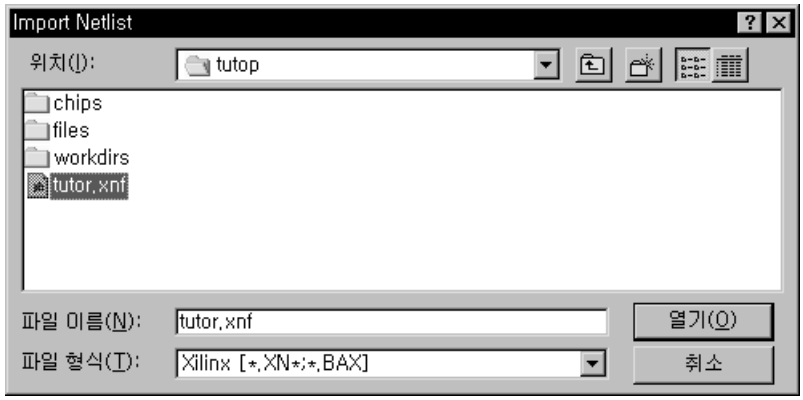
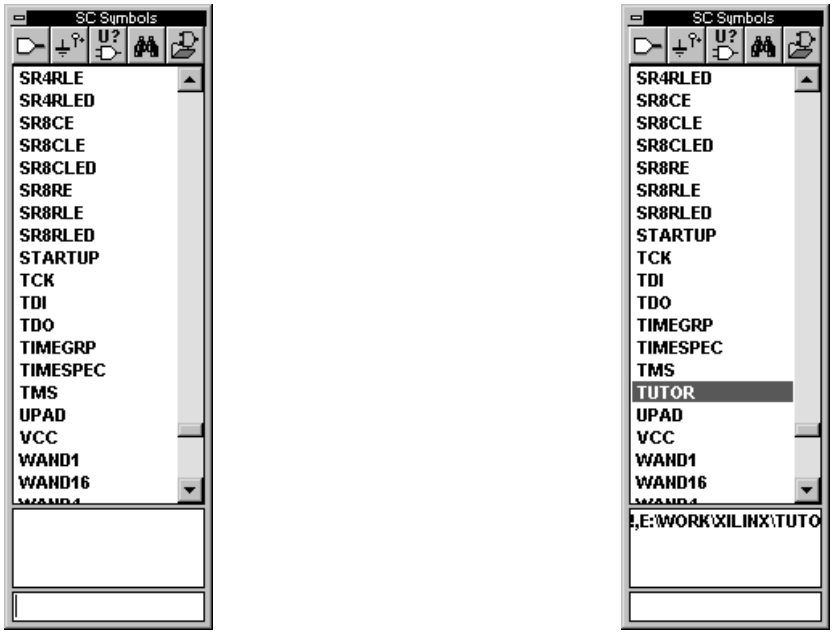


그림 28. Create Macro Symbol from Netlist 메뉴 실행 후 화면

Import netlist 창에서 1.6 의 "가"항에서 생성한 xnf file을 선택한 후 열기를 누르면 xnf netlist를 불러와서 symbol로 변환하는 작업을 수행한다. 그러면 그림 29와 같이 SC symbol창에 생성된 symbol이 나타나며, 이를 그림 30과 같이 불러다 사용할 수 있다.



(가) (나)

그림 29. SC Symbol 창에 나타난 생성된 symbol

(가) Tutor에 대한 Create Macro Symbol from Netlist 명령 실행 이전 모습

(나) Tutor에 대한 Create Macro Symbol from Netlist 명령 실행 이후 모습

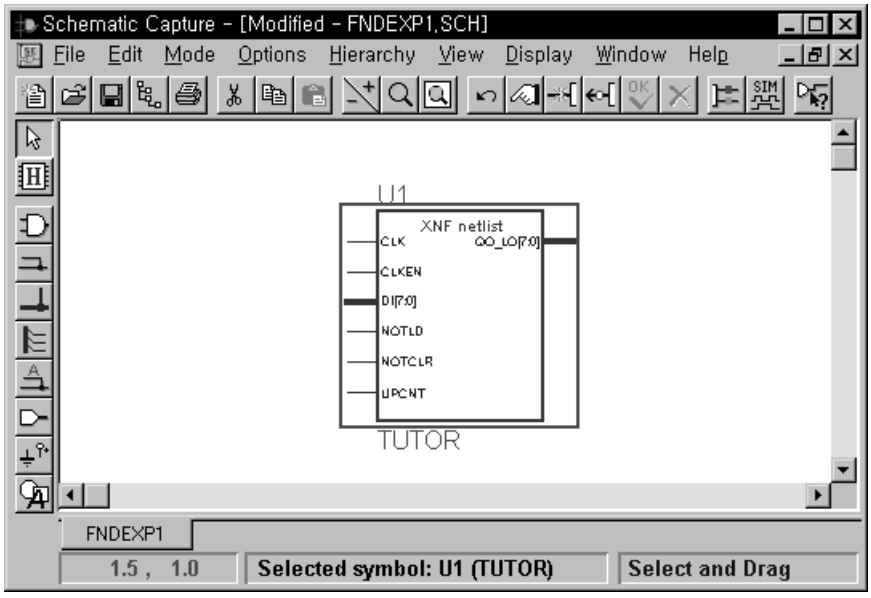


그림 30. Tutor library의 사용

3. Simulation에서의 사용

VHDL을 합성하여 생성한 netlist를 Simulation하기 위해서는 1.6 의 "가"항에서 생성한 xnf file 이용하여 가능하다. Xilinx Foundation series의 Project manager에서 Sim Func을 선택(그림 31)한 후 File 메뉴에서 Load netlist를 선택하면 그림 32와 같이 나타난다.

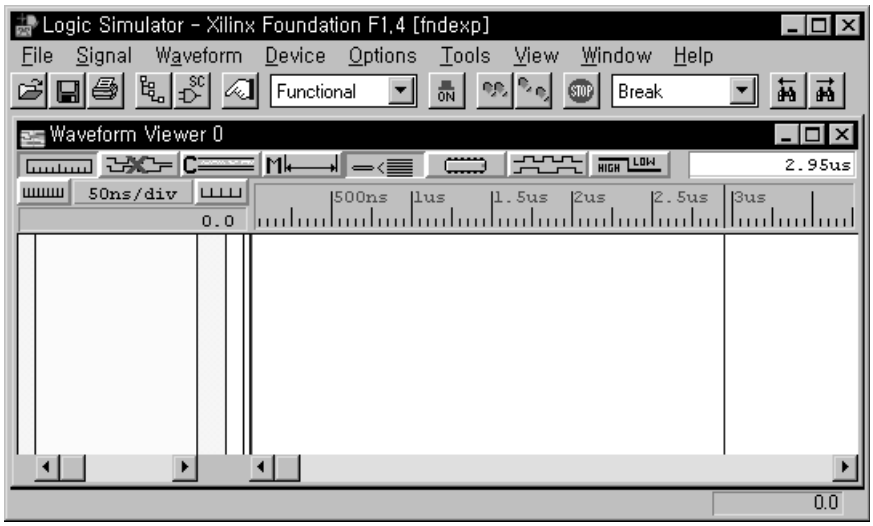


그림 31. Project manager의 Sim Func의 실행 화면

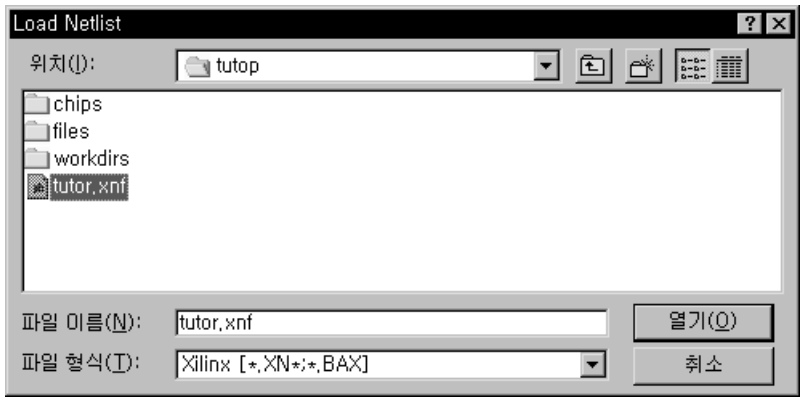


그림 32. Load netlist의 실행 화면

여기서 1.6 의 "가"항에서 생성한 xnf file을 선택하면 netlist를 불러오는 작업을 수행하고 마친다. 다음으로 Logic Simulator창에서 Signal 메뉴 아래의 Add Signals를 선택한 후 signal을 선택하면 선택된 signal이 그림 33과 같이 Logic Simulation 창에 나타난다.

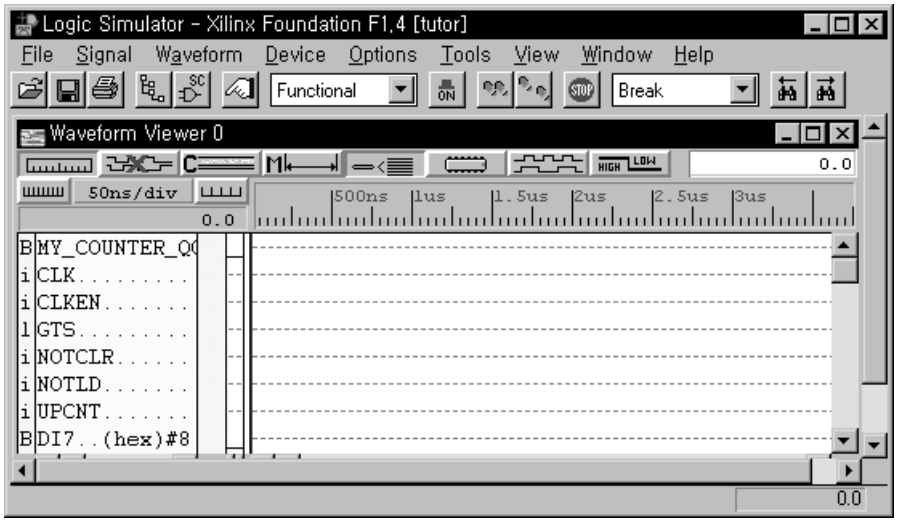


그림 33. Add Signals 이후의 화면

4. Configuration file의 생성

Xilinx Foundation Series의 Design Manager를 실행한 후 File 메뉴 아래 New Project를 선택하면 그림 34와 같이 나타난다. 여기서 1.6 의 "가"항에서 생성한 xnf file을 선택하면 Design manager 창에 그림 35와 같이 project가 생성된다.

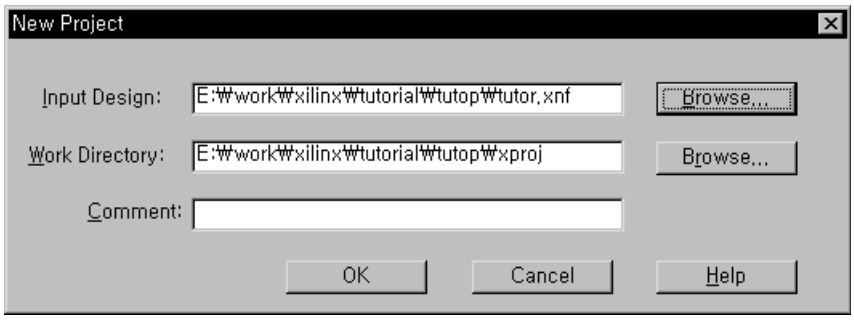


그림 34. Design manager의 New Project 화면

그림 35. Project의 생성

Design 메뉴아래 Implement를 실행하면 그림 36과 같이 Implement 창이 나타난다. 여기서 device를 다시 선택할 수 있으며, 그림 37과 같이 Option을 눌러 Produce Configuration Data를 선택한 후 실행하면 Translate, Map, Place&Route, Timing, Configurate 과정을 거쳐서 자동으로 programming이 가능한 bit file을 생성한다.

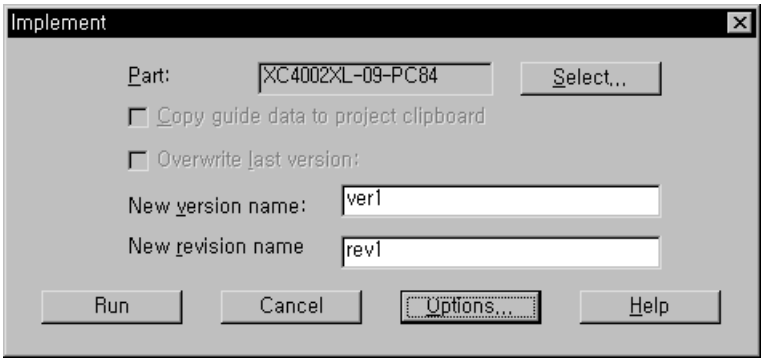


그림 36. Implement 실행 후 화면

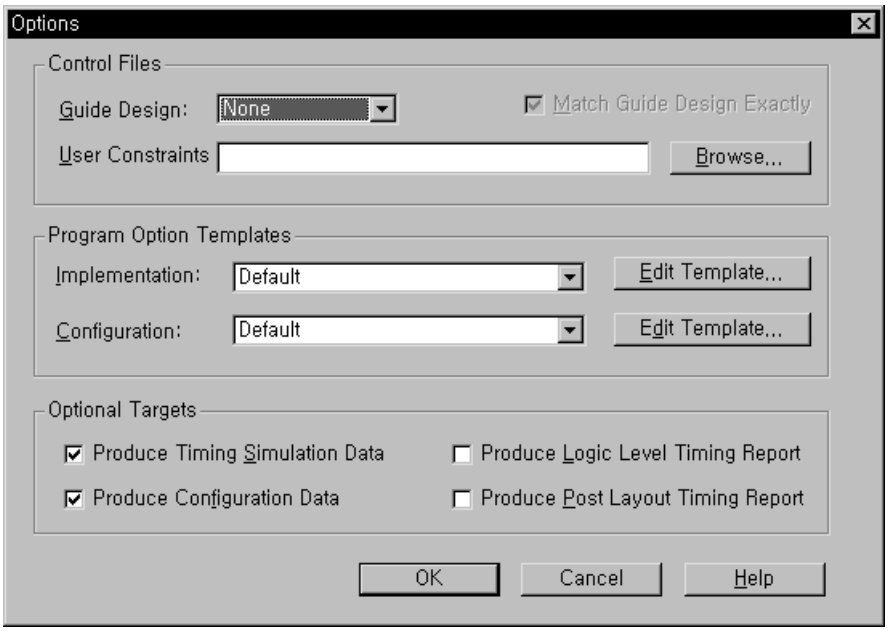


그림 37. Implement의 Option 실행 화면