

비바도(Vivado) 디자인 수트 사용 지침서

디자인 플로우 개요

UG888 (v2014.1) April 1, 2014





Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.

©Copyright 2012-2014 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, UltraScale, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

Revision History

Date	Version	Revision
04/01/2014	2014.1	Validated with release.

Table of Contents

Revision History.....	2
비바도 디자인 플로우 사용 지침서.....	5
디자인 플로우 개요.....오류! 책갈피가 정의되어 있지 않습니다.	
프로젝트 모드 혹은 비-프로젝트 모드로 작업.....	5
Tcl 명령어 이용.....	6
지침 디자인(Tutorial Design) 디스크립션.....	7
소프트웨어 요건.....	8
하드웨어 요건.....	8
지침 디자인 파일 준비.....	8
Lab 1: 비-프로젝트 디자인 플로우 이용하기.....오류! 책갈피가 정의되어 있지 않습니다.	
Step 1: 예제 스크립트 검토.....	9
Step 2: 예제 디자인으로 비바도 시작하기.....	9
Step 3: 디자인 합성.....	10
Step 4: 비바도 IDE 시작.....	11
Step 5: Timing Constraints 및 I/O 플래닝 정의.....	13
Timing Constraints 정의.....	13
I/O 플래닝.....	14
Step 6: 수정된 Constraints 내보내기.....	16
Step 7: 디자인 구현.....	17
Step 8: 디자인 체크포인트 시작.....	18
Step 9: 구현 결과 분석.....	18
Step 10: 비바도 툴 종료.....	21
Lab 2: 프로젝트 디자인 플로우 이용하기.....	22

Step 1: 프로젝트 생성	22
비바도 시작.....	22
새로운 프로젝트 생성.....	23
Step 2: Sources Window 및 Text Editor 이용하기	27
Sources Window 및 Project Summary 살펴보기	28
Text Editor 살펴보기	28
Step 3: RTL 디자인 살펴보기.....	30
Step 4: IP 카탈로그 이용하기.....	31
Step 5: Behavioral Simulation 실행하기.....	32
Step 6: 디자인 실행 설정 검토	33
Step 7: 디자인 합성 및 구현	35
Step 8: 합성된 디자인 분석.....	36
Step 9: 구현된 디자인(Implemented Design) 분석.....	39
구현된 디자인 오픈	39
라우팅 분석.....	40
Step 10: 비트스트림 파일 생성.....	42
Step 11: Journal 파일에서 Tcl 스크립트 생성.....	43
Log 및 Journal 검토	43
일괄(Batch) 프로젝트 스크립트 편집.....	45
일괄(Batch) 프로젝트 스크립트 실행.....	46
Step 12: 디자인 상태 확인	47
요약.....	오류! 책갈피가 정의되어 있지 않습니다.

비바도(Vivado) 디자인 플로우 사용 지침서



필독사항: 이 사용 지침서는 킨텍스-7(Kintex[®]-7) 디바이스 제품군의 사용을 전제로 한다. 만약 이 디바이스 제품군을 구비하지 않았다면, 비바도(Vivado) 툴 설치를 업데이트해야 할 수도 있다. 디자인 툴이나 디바이스를 추가하기 위한 보다 자세한 정보는 비바도 디자인 스위트 사용자 가이드의 릴리스 노트 및 설치, 라이선싱([UG973](#))을 참조하세요.

디자인 플로우 개요

이 지침서는 자일링스(Xilinx[®]) 비바도(Vivado[®]) IDE(Integrated Design Environment) 사용을 추천하는 이용 모델 및 디자인 플로우를 소개한다. 이 지침서는 아래 설명한 것처럼, 각기 다른 두 개의 디자인 플로우를 이용해 RTL에서 비트스트림까지 소규모 디자인 예제와 관련된 기본적인 단계들을 기술하고 있다. 두 플로우는 비바도 IDE를 이용하거나 배치 Tcl 스크립트를 통해 실행이 가능하다. 비바도 Tcl API는 상당한 유연성을 제공하며, 설정을 용이하게 하고, 디자인 실행은 물론, 분석 및 디버깅을 수행할 수 있다.



팁: 비디오를 통해서도 빠르게 비바도 디자인 스위트 디자인 플로우에 대한 보다 자세한 정보를 확인할 수 있습니다. <http://www.xilinx.com/training/vivado/vivado-design-flows-overview.htm>

프로젝트 모드 혹은 비-프로젝트 모드로 작업

일부 사용자는 디자인 플로우 프로세스 및 디자인 데이터를 자동으로 관리하는 디자인 툴을 선호하는 반면, 또 다른 사용자는 직접 소스 및 프로세스를 관리하는 것을 선호한다. 비바도 디자인 스위트는 디자인 소스 파일을 관리하기 위해 프로젝트 파일(.xpr) 및 디렉토리 구조를 이용하며, 각각의 합성 및 구현 결과를 저장하고, 디자인 플로우 전반에 걸쳐 프로젝트 상태를 추적한다. 디자인 데이터, 프로세스, 상태에 대한 자동화된 관리는 프로젝트 인프라가 필요하다. 자일링스는 이러한 플로우를 프로젝트(Project Mode)라고 지칭하고 있다.

다른 사용자들은 FPGA 디자인 프로세스를 소스 파일 편집에 더 가까운, 즉 간단하게 소스를 컴파일하고, 디자인을 구현하고, 결과를 리포트하는 형태로 실행하기를 선호한다. 이러한 편집 스타일 플로우를 비-프로젝트 모드(Non-Project Mode)라 명기했다. 비바도 디자인 스위트는 이러한 이용 모델 두가지 모두 쉽게 수용할 수 있다.

다음은 프로젝트 모드 및 비-프로젝트 모드에 대한 간략한 개요를 살펴보고자 한다. 이러한 디자인 모드에 대한 보다 자세한 설명과 각각에 대한 특징 및 장점에 대해서는 *비바도 디자인 수트 사용자 가이드: 디자인 플로우 개요(UG892)*에서 확인할 수 있다.

비-프로젝트 모드

이 이용 모델은 디자인 데이터를 관리하고, 디자인 상태를 추적하기 위해 비바도 툴을 원하지 않는 스크립트-기반 사용자를 위한 것이다. 비바도 툴은 플로우 전반에 걸쳐 간단하게 메모리 상의 여러 소스 파일을 판독하고, 디자인을 컴파일한다. 구현 프로세스의 모든 단계에서 다양한 리포트를 생성하고, DRC(Design Rule Checks)를 실행하고, 디자인 체크포인트를 작성할 수 있다.

플로우 전반에 걸쳐, 메모리 상의 디자인을 열거나, 디자인 분석 및 넷리스트/Constraints 수정을 위해 비바도 IDE에서 모든 저장된 디자인 체크포인트를 오픈할 수 있다. 하지만 소스 파일은 비-프로젝트 모드로 실행 중인 경우 IDE에서 수정할 수 없다. 또한 이 모드는 소스 파일 및 실행 관리, 소스 파일과의 크로스-프로빙, 디자인 상태 리포트 등과 같은 프로젝트-기반 기능들을 수행할 수 없다는 점을 반드시 알아두어야 한다. 기본적으로 소스 파일은 매번 디스크 상에서 업데이트됨으로 반드시 이를 인지하고 있어야 하며, 디자인을 재로드해야 한다.

또한 비-프로젝트 모드 내에서 생성된 디폴트 리포트나 중간 파일은 없다. 따라서 Tcl 명령어로 디자인 체크포인트나 리포트를 직접 생성해야 한다.

프로젝트 모드

이 이용 모델은 소스 파일, Constraints, 결과 관리, 통합 IP 디자인, 소스 파일과의 크로스-프로빙 등의 기능을 비롯한 전반적인 디자인 프로세스 관리를 위해 비바도 툴을 필요로 하는 사용자를 위한 것이다. 프로젝트 모드에서 비바도 툴은 디자인 소스 파일과 IP 데이터, 합성 및 구현 실행 결과와 관련 리포트를 관리하기 위해 디렉토리를 생성한다. 비바도 디자인 수트는 소스 파일 상태 및 컨피규레이션, 디자인 상태를 관리하고 리포트한다. 여러분은 Constraints이나 명령어 옵션을 분석하기 위해 다중 실행을 생성하고 컨피규레이션할 수 있다. 비바도 IDE에서 구현 결과를 RTL 소스 파일과 크로스-프로빙할 수 있으며, 또한 Tcl 명령어로 전체 플로우를 스크립트하고, 필요할 때 비바도 IDE를 오픈할 수 있다.

Tcl 명령어 이용

Tcl 명령어 및 스크립팅 접근방식은 사용하는 디자인 플로우에 따라 각기 다르다. 비-프로젝트 모드를 사용하는 경우 소스 파일은 `read_verilog`, `read_vhdl`, `read_edif`, `read_ip`, `read_xdc` 명령어를 이용해 로드된다. 비바도 디자인 수트는 인-메모리 디자인 데이터베이스를 생성하고, 합성 및 시뮬레이션, 구현으로 넘어간다. 프로젝트 모드의 경우 소스 파일을 관리하고, 디자인 상태를 추적하는데 필요한 프로젝트 인프라를 생성하기 위해 `create_project`, `add_files`, `import_files`, `add_directories` 명령어를 이용할 수 있다. 배치 플로우에 있는 `synth_design`,

`opt_design`, `place_design`, `route_design`, `write_bitstream` 등 각각의 "atomic" 명령어는 all-inclusive 명령어로 불리는 `launch_runs`로 대체된다. `launch_runs` 명령어는 atomic 명령어와 디폴트 리포트를 생성하고, 실행 상태를 추적하기 위한 다른 명령어와 함께 분류된다. 프로젝트 모드를 위한 Tcl 실행 스크립트 결과는 비-프로젝트 모드와는 다르다. 이 지침서는 프로젝트 모드와 비-프로젝트 모드는 물론, 비바도 IDE까지 커버하고 있다.

이 지침서에서 논의된 많은 분석 기능들은 다른 지침서에서 보다 상세하게 기술되어 있다. 모든 명령어나 명령어 옵션을 여기에서는 언급하지 않았다. 틀에서 제공되는 전체 Tcl 명령어 리스트를 확인하기 위해서는 비바도 디자인 수트 Tcl 명령어 레퍼런스 가이드([UG835](#))를 참조하면 된다.

이 지침서는 독립적으로 수행이 가능한 두 개의 Lab이 포함되어 있다.

Lab 1: 비-프로젝트 디자인 플로우 이용하기

- bft 디자인을 구현하는 샘플 실행 스크립트 검토
- 각 단계에서 다양한 리포트 확인
- `vivado.log` 파일 리뷰
- 디자인 체크포인트 작성
- Timing Constraints 정의 및 I/O 플래닝을 리뷰하고, Constraints 업데이트 방법을 시연하기 위해 합성 후 비바도 IDE 오픈

Lab 2: 프로젝트 기반 디자인 플로우 이용하기

- 새로운 프로젝트 생성
- 비바도 IDE를 이용해 bft 디자인 구현 검토
- 각 단계에서 다양한 리포트 확인
- 합성된 디자인을 오픈하고, Timing Constraints 정의 및 I/O 플래닝, 디자인 분석 리뷰
- 타이밍, 전력, 리소스 활용도, 라우팅, 크로스-프로빙을 분석하기 위해 구현된 디자인 오픈
- `vivado.jou` 파일에서 Tcl 스크립트 종료 및 생성(`launch_runs`)
- 새롭게 생성된 Tcl 스크립트를 이용해 디자인 재실행
- 비바도 IDE에서 프로젝트 오픈하고, 배치 실행으로 디자인 상태가 올바른지 확인

지침 디자인(Tutorial Design) 디스크립션

이 지침서 전반에 사용된 샘플 디자인은 bft라고 불리는 소형 디자인으로 구성된다. bft 디자인에는 여러 VHDL 및 Verilog 소스 파일은 물론, XDC Constraints 파일이 있다.

이 디자인은 xc7k70T 디바이스를 타겟하고 있다. 소형 디자인은 지침서가 최소한의 하드웨어 요건으로 구동되도록 하고, 적시에 지침서를 완성하고, 데이터 크기를 최소화하기 위해 사용되었다.

소프트웨어 요건

이 지침서는 2014.1 비바도 디자인 수트 소프트웨어 버전이나 이후 버전의 설치가 필요하다. 설치 명령어 및 정보는 *비바도 디자인 수트 사용자 가이드: 릴리스 노트, 설치, 라이선싱(UG973)*을 참고하면 된다.

하드웨어 요건

지원되는 OS는 Redhat 5.6 Linux 64 및 32bit, Windows 7, 64, 32bit 등이다.

자일링스는 비바도 툴을 사용할 때 최소 2GB의 RAM을 추천한다.

지침 디자인 파일 준비

아래 위치의 비바도 디자인 수트 예제 디렉토리에서 이 지침서를 위한 파일을 찾을 수 있다.

- **<Vivado_install_area>/Vivado/<version>/examples/Vivado_Tutorial**

또한 로컬 디렉토리에 지침 파일을 작성하거나, 혹은 시작 조건에 따라 파일을 저장하기 위해 제공되는 zip 파일을 언제든지 추출할 수 있다.

소프트웨어 인스톨에서 모든 작성 가능한 위치로 zip 파일 콘텐츠를 추출한다.

- **<Vivado_install_area>/Vivado/<version>/examples/Vivado_Tutorial.zip**

추출된 Vivado_Tutorial 디렉토리는 이 지침서에서는 *<Extract_Dir>*로 지칭하였다.

참조: 이 지침서를 검토하는 동안 여러분은 지침 디자인 데이터를 수정하게 될 것이다. 따라서 이 지침서를 시작할 때마다 원래의 Vivado_Tutorial 디렉토리의 새로운 카피를 이용해야 한다.

Lab 1: 비-프로젝트 디자인 플로우 이용하기

이 lab은 비-프로젝트 모드와 관련 Tcl 명령어에 초점을 맞추고 있다.

Step 1: 예제 스크립트 검토

1. text editor 에서 예제 스크립트 `<Extract_Dir>/Vivado_Tutorial/create_bft_batch.tcl` 을 열고, 각기 다른 단계별로 리뷰,

STEP#0: 출력 디렉토리 위치 정의.

STEP#1: 디자인 소스 및 Constraints 설정.

STEP#2: 합성 실행, 디자인 체크포인트 작성, 리포트 생성.

STEP#3: 배치 실행 및 명령어 최적화, 디자인 체크포인트 작성 및 리포트 생성.

STEP#4: 라우팅 명령어 실행, 디자인 체크포인트 작성, 리포트 실행.

STEP#5: 비트스트림 실행.

대부분의 Tcl 명령어는 코멘트 아웃(Comment Out) 임을 주목해야 한다. 따라서 차례로 하나씩 수동으로 실행해야 한다.

2. 이 지침서 후반에 명령어를 복사 및 붙여넣기 해야 하기 때문에, 예제 스크립트를 **오픈한 상태로 놔둔다**.

Step 2: 예제 디자인으로 비바도 시작하기

- 리눅스의 경우

1. lab 자료가 저장된 디렉토리로 변경:

```
cd <Extract_Dir>/Vivado_Tutorial
```

2. 비바도 디자인 수트 Tcl Shell 및 지침 디자인을 생성하기 위한 Tcl 스크립트 소싱 시작:

```
vivado -mode tcl -source create_bft_batch.tcl
```

- 윈도우의 경우

1. 비바도 디자인 수트 Tcl Shell 시작:

```
Start > All Programs > Xilinx Design Tools > Vivado 2014.x > Vivado 2014.x Tcl Shell1
```

¹ 여러분의 비바도 디자인 수트가 **Start** 메뉴 상의 **Xilinx Design Tools**과 다르다면 설치가 요청될 수 있다.

2. Tcl Shell 에서 lab 자료가 저장된 디렉토리로 변경:

```
Vivado% cd <Extract_Dir>/Vivado_Tutorial
```

3. 지침 디자인을 생성하기 위한 Tcl 스크립트 소싱:

```
Vivado% source create_bft_batch.tcl
```

스크립트 소싱이 완료된 후, 비바도 디자인 수트 Tcl shell(이후 Td shell로 표기)이 Tcl 프롬프트에 디스플레이된다: **Vivado%**

```

C:\Xilinx\Vivado\2013.1\bin\vivado.bat -mode tcl
Hello from User's init.tcl in C:/Users/<user_ID>/AppData/Roaming/Xilinx/Vivado
# proc findCmd <option> <
# # This returns a list of commands having the specified option
# # Could be enhanced to handle multiple options in AND or OR configuration
#   foreach cmd [lsort [info commands *]] {
#     catch {
#       if {[regexp "$option" [help -syntax $cmd]]} {
#         puts $cmd
#       }
#     }
#   }
# }
# ;
# proc args < cmdName > <
# # This command returns the short form of the specified command
# # help $cmdName -args
# }
Vivado% cd C:/Data/Vivado_Tutorial
Vivado% source create_bft_batch.tcl
# set outputDir ./Tutorial_Created_Data/bft_output
# file mkdir $outputDir
# read_vhdl -library bftLib [ glob ./Sources/hdl/bftLib/*.vhd1 ]
# read_vhdl ./Sources/hdl/bft.vhd1
# read_verilog [ glob ./Sources/hdl/*.v ]
# read_xdc ./Sources/bft_full.xdc
C:/Data/Vivado_Tutorial/Sources/bft_full.xdc
Vivado% _
  
```

그림 1. 비바도 및 Tcl 스크립트 소싱 시작

Tcl 프롬프트에 Tcl 명령어를 추가로 입력할 수 있다.

Step 3: 디자인 합성

1. create_bft_batch.tcl 스크립트의 **synth_design** 명령어 라인을 Tcl shell 로 복사하여 붙여넣고, 합성이 완료되기를 기다린다. 우측 마우스 버튼을 클릭해 팝업 메뉴를 이용하여 Tcl shell 로 붙여넣기 할 수 있다.

```
synth_design -top bft -part xc7k70tfbg484-2 -flatten rebuilt
```

참조: 예제 스크립트의 명령어는 코멘트이다. 앞의 '#' 문자는 복사하지 않는다. 그렇지 않으면 여러분의 명령어는 코멘트로 해석될 것이다.

2. scrolls by 함으로써 합성 리포트를 검토한다.

- 비바도 Tcl 프롬프트가 리턴되면, 합성 이후의 **write_checkpoint**, **report_timing_summary**, **report_power** 명령어를 복사 및 붙여넣기 한다.

```
write_checkpoint -force $outputDir/post_synth
report_timing_summary -file $outputDir/post_synth_timing_summary.rpt
report_power -file $outputDir/post_synth_power.rpt
```

- 출력 디렉토리에서 생성된 파일을 검토하기 위해 또 다른 창을 오픈한다. 윈도우 상에서는 파일 브라우저를 사용하는 것이 더 쉬울 수 있다.

<Extract_Dir>/Vivado_Tutorial/Tutorial_Created_Data/bft_output

- 생성된 다양한 리포트 (*.rpt) 파일을 오픈하기 위해 text editor 를 이용한다.

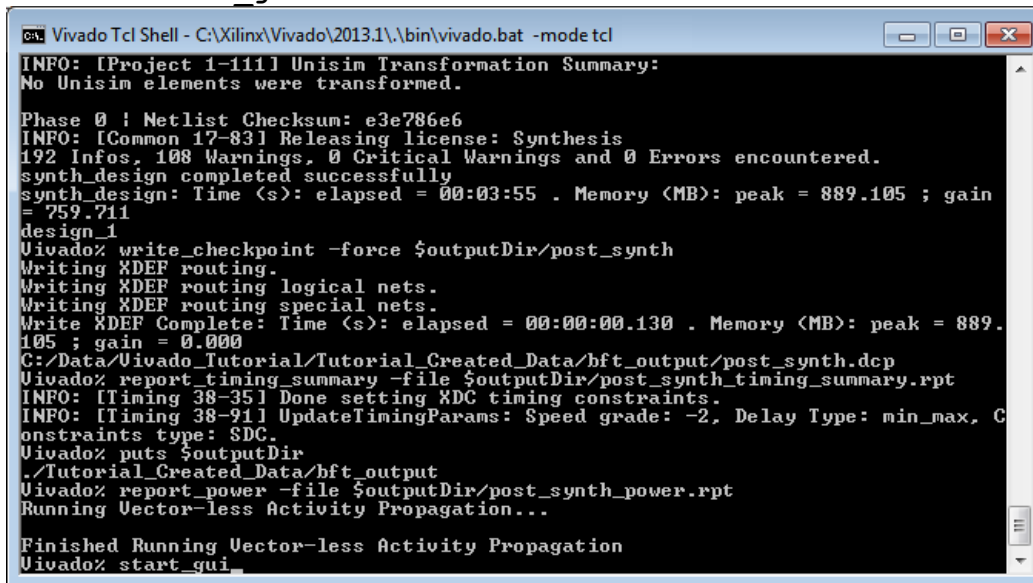
Step 4: 비바도 IDE 시작

비바도 프로젝트가 디스크 상에 생성되지 않았다 하더라도, 인-메모리 디자인을 틀에서 이용할 수 있기 때문에, Tcl shell에서 디자인 확인을 위해 비바도 IDE를 오픈할 수 있다.

비-프로젝트 모드는 비바도 IDE를 디자인 프로세스의 여러 단계에서 이용할 수 있다. 현 넷리스트 및 Constraints은 IDE에서 메모리로 로드되며, 분석 및 수정이 가능하다. 로직이나 Constraints에 대한 모든 변경은 메모리에 실시간으로 저장되며, 다운스트림 톨로 넘어간다. 이는 파일 저장 및 재로드가 요구되는 ISE 톨과는 상당히 다른 개념이다.

- start_gui** 명령어를 이용해 IDE 를 오픈한다.

Vivado% start_gui



```
Vivado Tcl Shell - C:\Xilinx\Vivado\2013.1\bin\vivado.bat -mode tcl
INFO: [Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.

Phase 0 : Netlist Checksum: e3e786e6
INFO: [Common 17-83] Releasing license: Synthesis
192 Infos, 108 Warnings, 0 Critical Warnings and 0 Errors encountered.
synth_design completed successfully
synth_design: Time (s): elapsed = 00:03:55 . Memory (MB): peak = 889.105 ; gain
= 759.711
design_1
Uivado% write_checkpoint -force $outputDir/post_synth
Writing XDEF routing.
Writing XDEF routing logical nets.
Writing XDEF routing special nets.
Write XDEF Complete: Time (s): elapsed = 00:00:00.130 . Memory (MB): peak = 889.
105 ; gain = 0.000
C:/Data/Uivado_Tutorial/Tutorial_Created_Data/bft_output/post_synth.dcp
Uivado% report_timing_summary -file $outputDir/post_synth_timing_summary.rpt
INFO: [Timing 38-35] Done setting XDC timing constraints.
INFO: [Timing 38-91] UpdateTimingParams: Speed grade: -2, Delay Type: min_max, C
onstraints type: SDC.
Uivado% puts $outputDir
./Tutorial_Created_Data/bft_output
Uivado% report_power -file $outputDir/post_synth_power.rpt
Running Vector-less Activity Propagation...

Finished Running Vector-less Activity Propagation
Uivado% start_gui
```

그림 1: start_gui 로 비바도 IDE 열기

비바도 IDE는 디자인 시각화 및 검색 능력을 제공한다. 비바도 IDE에서 디자인의 추가 분석 및 Constraints 처리를 수행할 수 있다.

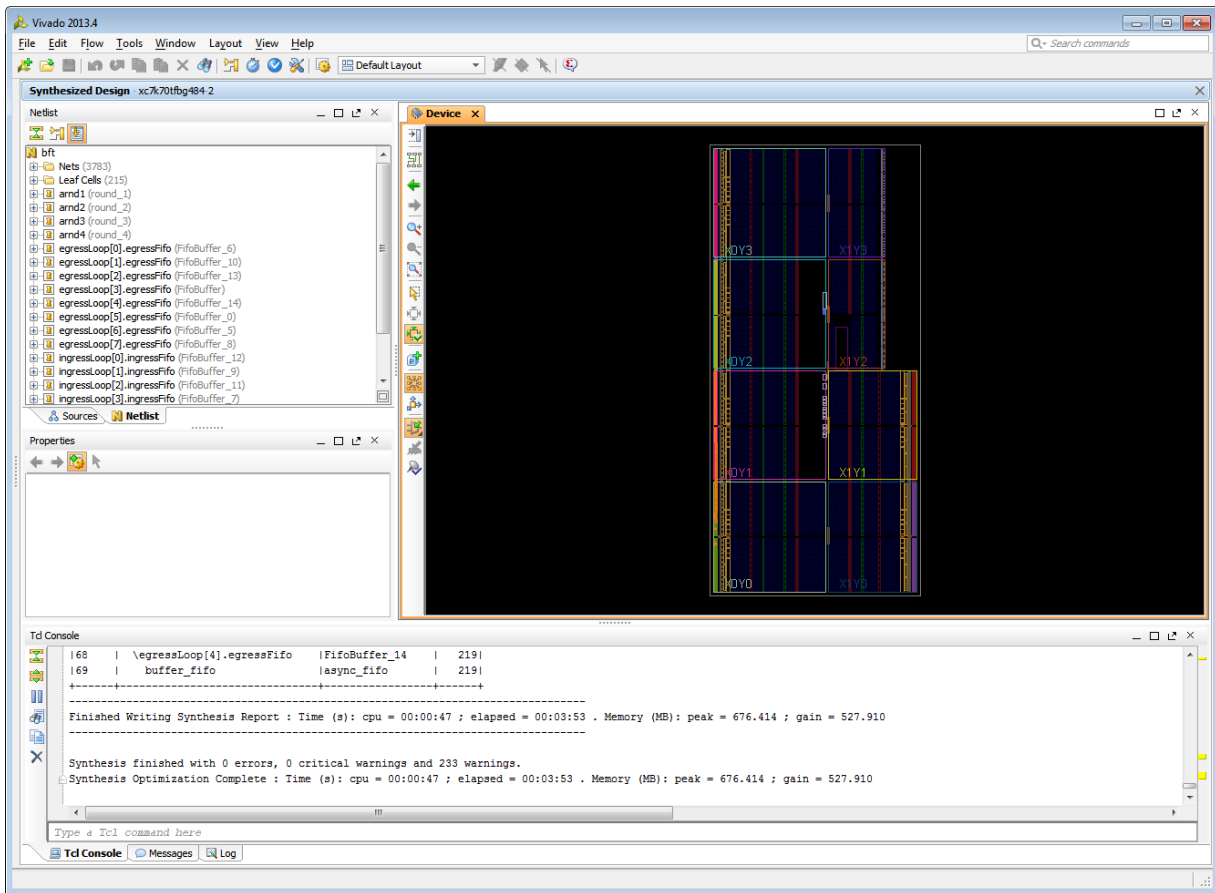


그림 2: 비바도 IDE -비-프로젝트 모드



팁: GUI를 중단하고 비바도 디자인 슈트 Tcl shell로 돌아가기 위해 **stop_gui** 명령어를 사용한다. 비바도 IDE에서 **File > Exit** 명령어를 사용하면, 비바도 툴에서 완전히 빠져나오게 된다.

디자인은 비-프로젝트 모드에서 프로젝트를 가지고 있지 않기 때문에, 비바도 IDE는 소스 파일이나 실행관리를 할 수 없다. 효과적으로 현재의 인-메모리 디자인을 분석한다. 또한 비바도 Flow Navigator 및 다른 프로젝트 기반 명령어를 비-프로젝트 모드에서 이용할 수 없다.

Step 5: Timing Constraints 및 I/O 플래닝 정의

구현 이전에 디자인을 위한 타이밍 및 물리적 제약조건(Physical Constraints)을 종종 정의해야 한다. 비바도 툴은 파일에서 Constraints을 로드하거나, IDE를 이용해 쌍방향으로 조건을 입력할 수 있도록 해준다.

Timing Constraints 정의

1. Timing Constraints 창 열기: **Window > Timing Constraints.**

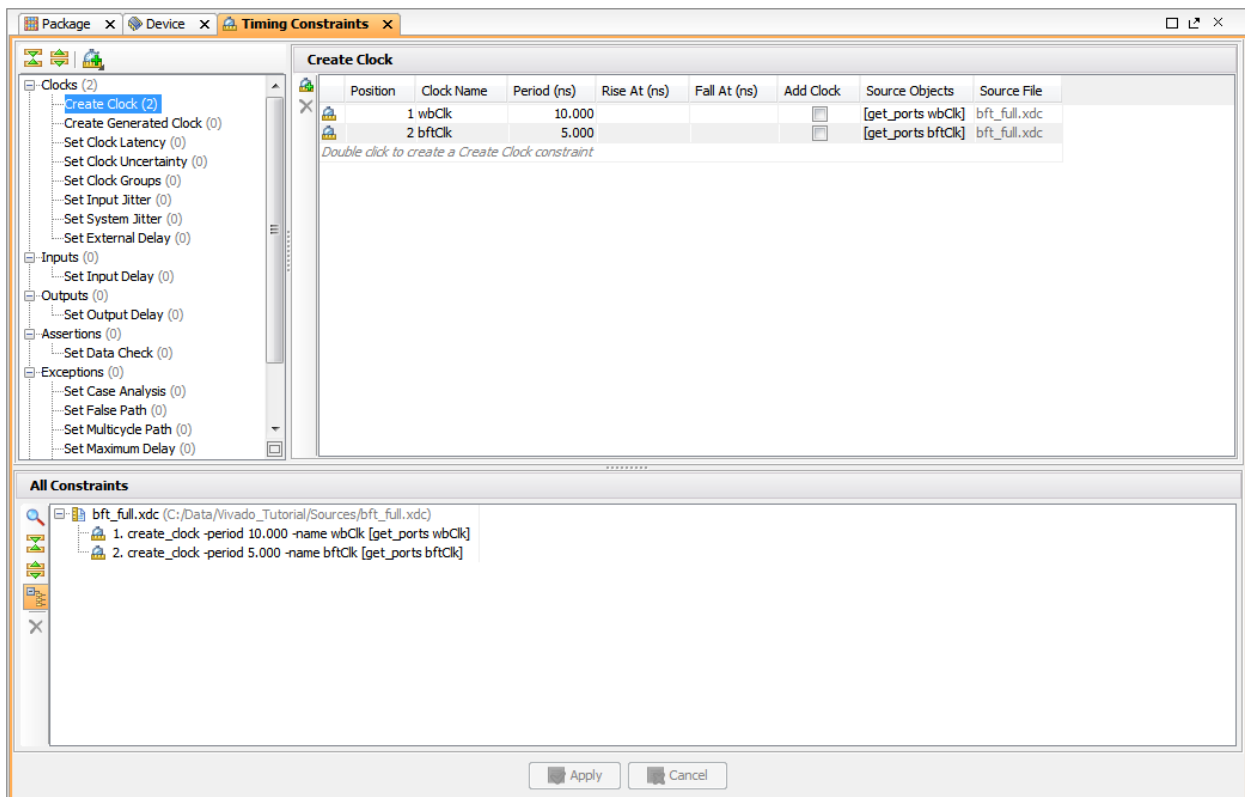


그림 3: Timing Constraints 정의

Timing Constraints 창의 좌측에 각기 다른 형태의 Constraints 트리 뷰가 디스플레이된다. 이는 Timing Constraints을 빠르게 정의할 수 있는 메뉴이다.

두 개의 클럭 Constraints, wbClk 및 bftClk가 Timing Constraints 창 우측의 Timing Constraints 스프레드시트에 디스플레이된다. 현재 정의되어 있는 Constraints 값을 스프레드시트에서 직접 편집하여 수정할 수 있다.

2. [그림 3](#)에 나타난 것처럼 Timing Constraints 창 좌측의 트리 뷰에서 클럭 카테고리 아래 **Create Clock** 을 더블-클릭한다.

참조: '+'를 클릭하면 필요시 클럭 카테고리 확장할 수 있다.

Create Clock 마법사는 클럭 Constraints을 정의하는데 도움을 준다. 하단의 Tcl 명령어는 실행될 XDC 명령어를 디스플레이한다는 점을 기억하자.

여기에서는 어떠한 Timing Constraints도 생성하거나 수정해서는 안된다.

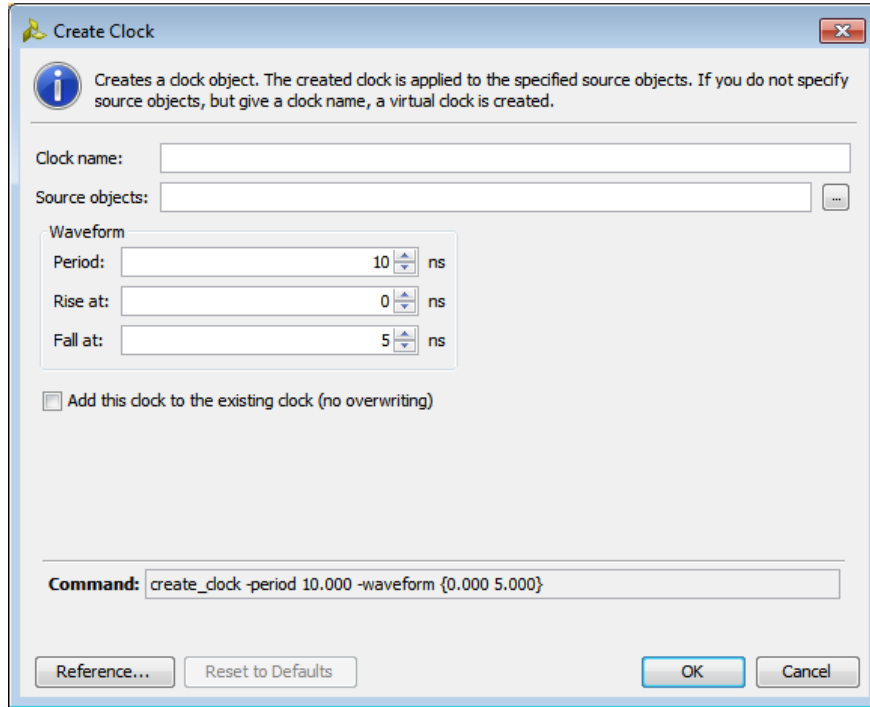


그림 4: Create Clock 대화상자

3. **Cancel** 클릭
4. 윈도우 창의 X 를 클릭해 **타이밍 제약조건(Timing Constraints)** 창을 닫는다.

비바도 디자인 수트는 디자인 분석 및 Constraints 할당을 위해 다양한 기능을 제공한다. 다른 지침서에서는 이러한 기능을 상세하게 다루고 있지만, 여기에서는 언급만 하기로 한다. 툴 메뉴에 있는 일부 기능들을 자유롭게 검토해 보기를 바란다.

I/O 플래닝

비바도는 I/O 핀 할당을 수행하고, 검증하기 위한 포괄적인 기능 세트를 갖추고 있다. 이에 대해서는 I/O 플래닝 지침서에서 보다 상세하게 다루고 있다.

1. [그림 그림 5](#)에 나타낸 것처럼 Layout Selector 폴다운에서 **I/O 플래닝**을 선택함으로써 I/O 플래닝 뷰를 오픈한다.

2. 활성화되지 않으면, **active** 뷰 **Package** 창을 만든다.

참조: 패키지 창이 열리지 않으면, 메인 메뉴에서 **Windows > Package** 명령어를 이용해 열 수 있다.

3. 우측 그림의 패키지 핀 내부의 오렌지 블록에 보이는 것처럼, 패키지 창에서 배치된 **I/O 포트**를 선택하기 위해 **더블-클릭**한다.

4. 동일한 I/O बैं크에서 또 다른 핀 사이트 상의 선택된 **I/O 포트**를 **드래그**한다.

5. 포트 이름과 패키지 핀 사이트 컬럼을 살펴보고, **I/O 포트 창**을 검토한다.

6. **I/O 포트 Properties** 창에 디스플레이된 데이터를 검토한다. 윈도우 아래에 있는 각 탭을 클릭한다.

7. 여러분이 변경한 포트 이름과 포트 사이트를 기억해야 한다.

필요 시 이를 적어두어야 한다. 이는 구현 후 XDC 파일에서 배치된 포트의 LOC Constraints을 찾아야 하기 때문이다.

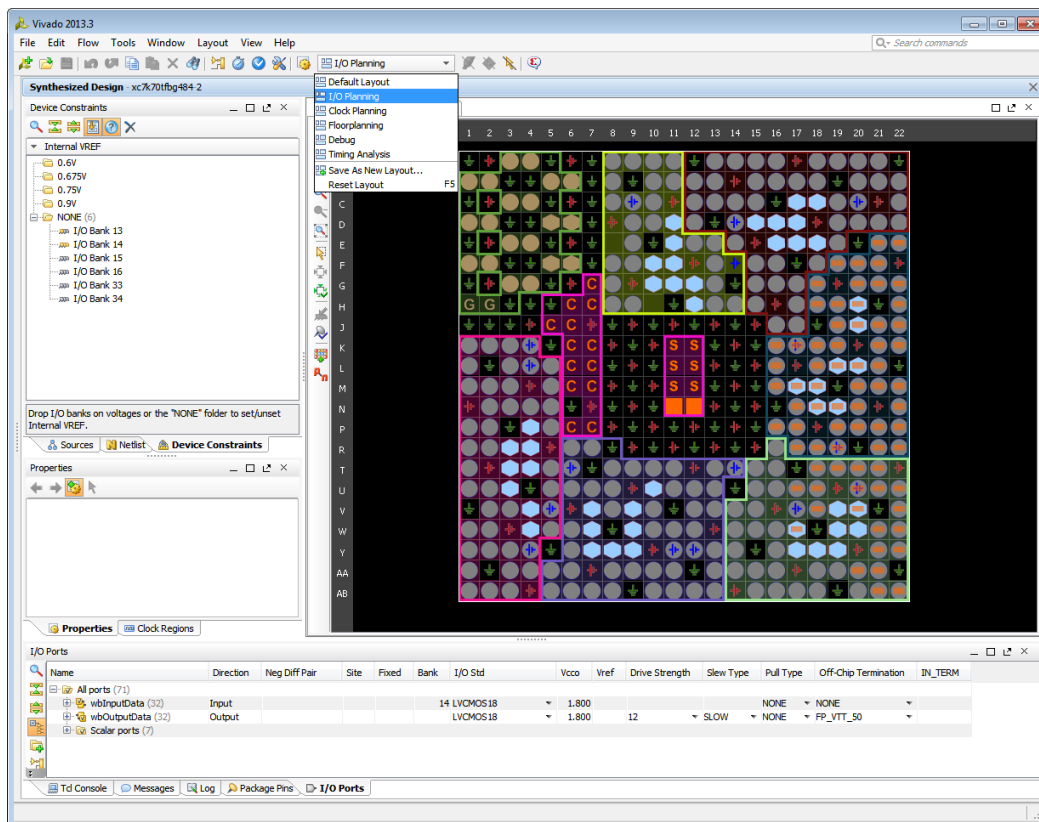


그림 5: I/O 플래닝 뷰 레이아웃 오픈

Step 6: 수정된 Constraints 내보내기

수정된 Constraints은 추후 사용할 수 있도록 출력할 수 있다. 또한 최신 변경내용을 포함한 디자인 체크포인트도 저장할 수 있다. 이 지침서 후반에 디자인 체크포인트를 검토하게 될 것이다.



필독사항: 비바도 디자인 수트는 NCF/UCF Constraints을 지원하지 않는다. 따라서 기존 UCF Constraints을 XDC 포맷으로 마이그레이션해야 한다. 보다 자세한 정보는 ISE에서 비바도 디자인 수트로의 마이그레이션 가이드([UG911](#))를 참조하면 된다.

1. 새로운 I/O LOC Constraints 값을 가지고 있는 수정된 XDC Constraints 파일을 출력하기 위해서는 Export Constraints 명령어를 사용한다.

File > Export > Export Constraints

Export Constraints 대화상자가 열리면 생성할 파일 이름을 명시할 수 있다.

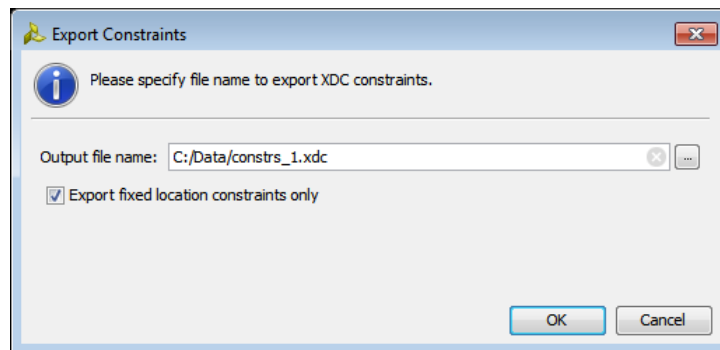


그림 6: Export Constraints

2. 파일에 대한 이름과 위치를 입력하고 **OK** 를 클릭한다.

Export fixed location constraints only에 체크박스가 있음을 주목하자. 이것이 설정되면, 모든 배치된 셀이 아닌, 고정된 셀의 LOC Constraints 만을 내보내게 된다. 보다 상세한 고정 및 비고정 셀에 대한 설명은 비바도 디자인 수트 사용자 가이드; 디자인 분석 및 클로저 기법에서 확인할 수 있다.[\(UG906\)](#)

3. Text Editor 에서 Constraints 파일을 열려면, **File > Open File** 명령어를 이용한다.
4. 새롭게 추출된 Constraints 파일을 선택하고 **OK** 를 클릭한다.
5. 이 파일은 여러분이 앞서 만든 I/O 배치 변경이 반영되어 있다.

Text Editor에서 어떠한 ASCII 파일이라도 열 수 있다. 이는 Tcl 스크립트 및 Constraints 파일을 편집하고, 리포트를 검토하는데 매우 유용하다. Text Editor는 컨텍스트 센서티브(Context Sensitive)하며, Verilog, VHDL, XDC, Tcl과 같은 파일 타입을 디스플레이할 때 키워드나 코멘트를 하이라이트한다.

6. IDE 하단에 있는 **Tcl Console** 탭을 선택하고, **stop_gui** 명령어를 입력한다.
비바도 IDE를 종료하고, Tcl shell의 Tcl 프롬프트로 돌아간다.

Step 7: 디자인 구현

1. **run_bft_batch.tcl** 스크립트를 열거나 앞에서 가져온다.
2. **opt_design** 에서 **write_bitstream** 까지 차례대로 스크립트에 있는 Tcl 명령어를 각각 따로 따로 복사 및 붙여넣기를 한다:

```
opt_design
place_design
phys_opt_design
write_checkpoint -force $outputDir/post_place
report_timing_summary -file $outputDir/post_place_timing_summary.rpt
route_design
write_checkpoint -force $outputDir/post_route
report_timing_summary -file $outputDir/post_route_timing_summary.rpt
report_timing -sort_by group -max_paths 100 -path_type summary -file
$outputDir/post_route_timing.rpt
report_clock_utilization -file $outputDir/clock_util.rpt
report_utilization -file $outputDir/post_route_util.rpt
report_power -file $outputDir/post_route_power.rpt
report_drc -file $outputDir/post_imp_drc.rpt
write_verilog -force $outputDir/bft_impl_netlist.v
write_xdc -no_fixed_only -force $outputDir/bft_impl.xdc
write_bitstream -force $outputDir/bft.bit
```

3. 각 명령어를 검토하고, 이 명령어들이 실행되면 여러 메시지가 생성된다.
4. 출력 디렉토리에서 생성된 파일을 검토한다.

<Extract_Dir>/Vivado_Tutorial/Tutorial_Created_Data/bft_output

5. 생성된 다양한 리포트 파일(*.rpt)을 열기 위해 text eidtor 를 이용한다.
6. **bft_impl.xdc** 파일을 엽니다.
7. 디자인이 앞서 수정한 I/O Port Constraints 에 따라 구현되었는지 확인한다.

Step 8: 디자인 체크포인트 시작

비바도 IDE는 모든 저장된 디자인 체크포인트를 열 수 있다. 이 디자인 '스냅샷'은 합성, 구현, 분석을 위해 비바도 IDE나 Tcl shell에서 열 수 있다.

1. 비바도 IDE 다시 열기: **start_gui**

이는 IDE의 인-메모리 액티브 디자인에 로드한다.

이제 구현된 디자인 체크포인트에 로드하게 되며, 지금의 인-메모리 디자인은 종료된다.

2. 구현된 체크포인트 열기.

File > Open Checkpoint 이용하여 체크포인트 파일 선택을 확인한다.

```
<Extract_Dir>/Vivado_Tutorial/Tutorial_Created_Data/bft_output/  
post_route.dcp
```

3. 지금의 인-메모리 디자인을 종료하기 위해 **Yes** 를 선택한다.

4. 프롬프트가 나오면 **Close Without Saving** 을 선택한다.

이제 IDE의 시각화 및 분석 기능을 이용할 수 있으며, 배치 및 라우팅된 디자인 체크포인트로 진행된다.

Step 9: 구현 결과 분석

비바도는 수많은 측면에서 디자인 및 디바이스 데이터를 검토하기 위한 방대한 기능 세트를 갖추고 있다. 전력 및 타이밍, 활용도, 클럭 등을 위한 표준 리포트를 생성할 수도 있다. Tcl API를 통한 비바도 툴의 커스텀 리포팅 기능은 매우 방대하다.

1. 타이밍 데이터를 분석하기 위해 `report_timing_summary` 명령어를 실행한다.

Tools > Timing > Report Timing Summary

2. Report Timing Summary 대화상자에서, 디폴트 실행 옵션을 수락하기 위해 **OK** 를 클릭한다.

Timing Summary 창에서 제공되는 정보를 검토한다. Timing Summary 창의 좌측 트리에서 다양한 카테고리를 선택하고, 디스플레이된 데이터를 검토한다.

3. 이제 타이밍 분석을 수행하기 위해 `report_timing` 명령어를 실행한다.

Tools > Timing > Report Timing

4. Report Timing 대화상자에서, 디폴트 실행 옵션을 수락하기 위해 **OK** 를 클릭한다.

5. Timing Results 창의 리스트 중에서 **첫 번째 경로**를 선택한다.

6. 경로를 자세히 보기 위해 **Path Properties** 창을 **최대화**하거나 움직일 수 있다.

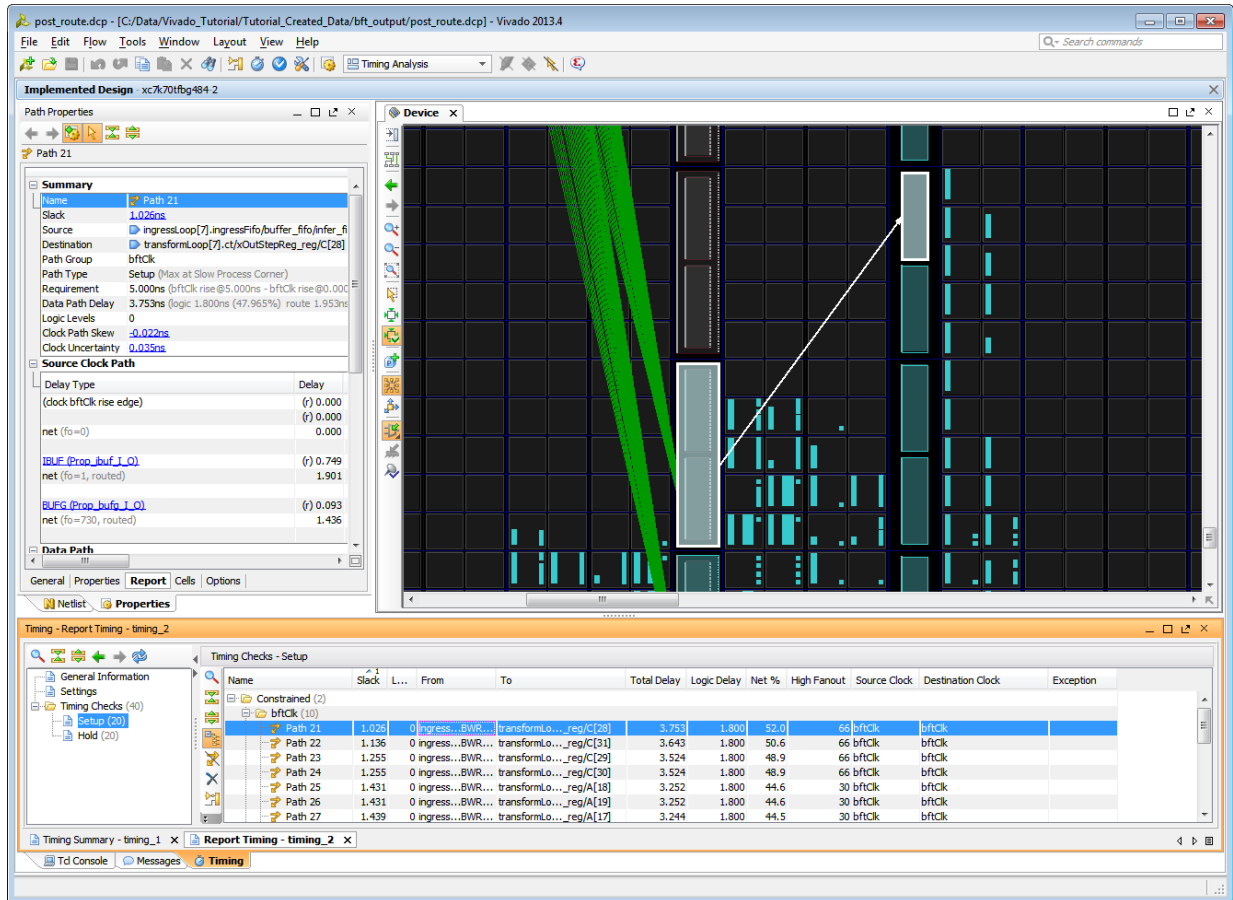



그림 7: 최대화시킨 Path Properties 창

7. 윈도우 배너에서 **Restore** 버튼이나 **Dock** 버튼을 클릭해서 Path Properties 창을 복원시킨다.
8. **Timing – Report Timing** 창에서, 팝업메뉴를 열기 위해 **우측 클릭**한 다음, 선택 경로를 위한 Schematic 창을 열기 위해 **Schematic** 명령어를 선택한다.

참조: Schematic 창을 열기 위해 F4 기능 키를 눌러도 된다.

9. 스케매틱 커넥션을 확장하고, 디자인 계층(Hierarchy)을 이동하기 위해 셀이나 핀, 혹은 와이어와 같은 스케매틱 객체를 더블 클릭한다.
10. **Schematic** 창을 **종료**하거나, 앞에서 가져온 Device 윈도우 탭을 클릭한다.
11. Device 창에서, **Routing Resources** 버튼  이 상세한 디바이스 라우팅을 디스플레이할 수 있도록 인에이블되어 있는지 확인한다.

Device 창은 선택된 경로에 대한 라우팅을 디스플레이하고 하이라이트 해준다.

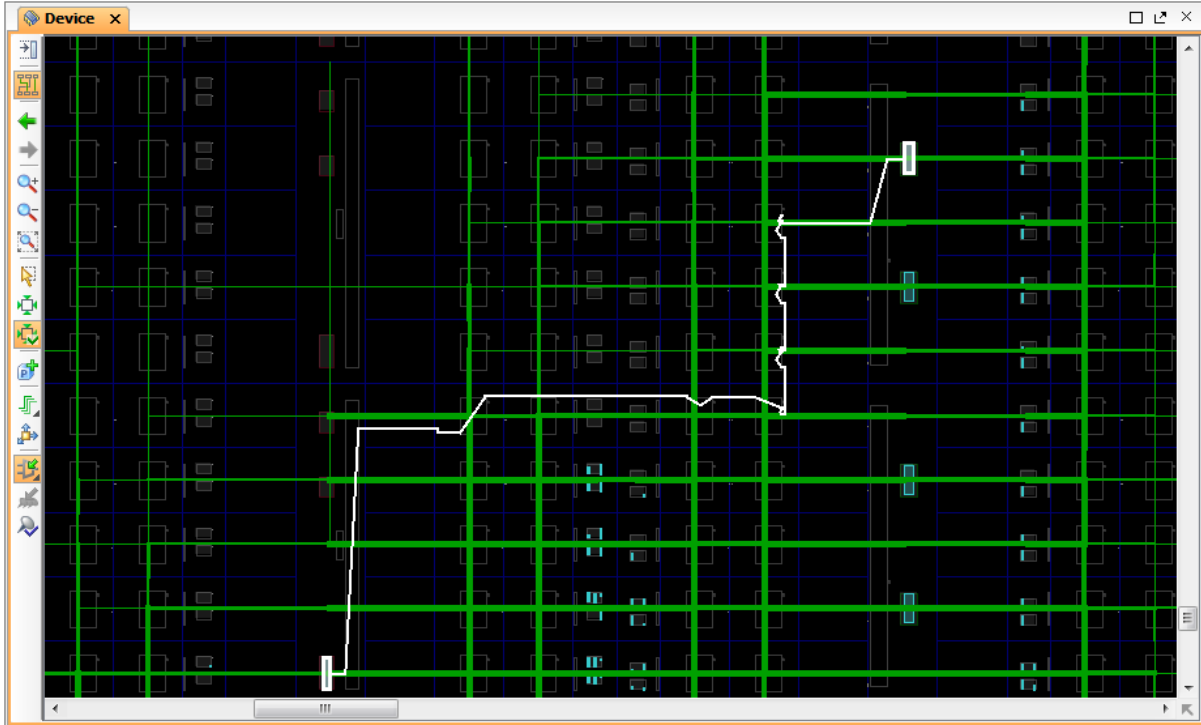


그림 8: 디바이스 라우팅 디스플레이

12. 선택된 객체를 자동으로 확대하기 위해 비바도 IDE 가 인에이블되도록 Device 창의 툴바 메뉴에서 **Auto Fit Selection** 버튼을 선택한다.
13. Timing 결과 창에서 추가 **경로** 일부를 선택한다.
14. Device 창에서 선택된 경로의 **라우팅**을 검토한다.
15. 툴의 메인 **메뉴**를 확장하고, **Timing** 및 **Report** 와 같은 각기 다른 서브-메뉴 아래에 제공되는 분석 기능을 검토한다.
16. 제공되는 분석 명령어 일부를 실행한다: **Report Power, Report Clock Interaction, Report Clock Networks, Report Utilization** 등.

대부분의 Design Analysis 기능들은 다른 비바도 지침서에서 다루고 있다.

Step 10: 비바도 툴 종료

비바도 툴은 비바도가 시작되면서, 디렉토리 안에 `vivado.jou`라고 불리는 저널 파일과 `vivado.log`라고 지칭된 로그 파일을 작성한다. 로그 파일은 디자인 세션이 진행되는 동안의 Tcl 명령어 실행 기록이며, 이러한 명령어의 결과로서 툴에 의해 리턴되는 메시지다. 저널은 세션이 진행되는 동안의 Tcl 명령어 실행 기록으로, 새로운 Tcl 스크립트를 생성하는 시작 지점으로 사용할 수 있다.

비바도 IDE 종료:

1. **Tcl Console** 창 탭과 타입을 선택한다: `stop_gui`

2. 비바도 종료:

```
Vivado% exit
```

3. 비바도 로그(`vivado.log`) 파일을 검토한다.

윈도우 상에서 로그파일의 위치를 확인하고, 오픈하기 위한 파일 브라우저를 사용하는 것이 더 간편할 수 있다. 비바도 로그 및 저널 파일의 위치는 비바도 툴이 시작되었던 디렉토리가 되며, 윈도우 데스크톱 아이콘에서 별도로 설정이 가능하다. 여기에서는 Lab #2에서 컨피규레이션할 것이다.

이 경우, 아래 위치에서 로그 파일을 찾는다:

```
<Extract_Dir>/Vivado_Tutorial/vivado.log2
```

로그 파일은 비바도 세션이 진행되는 동안 실행된 모든 Tcl 명령어 히스토리 및 결과를 포함하고 있다.

4. 비바도 저널(`vivado.jou`) 파일을 검토한다.

윈도우 상에서 파일 브라우저를 사용하는 것이 더욱 용이할 것이다. 아래 위치에서 저널 파일을 찾는다:

```
<Extract_Dir>/Vivado_Tutorial/vivado.jou
```

저널 파일은 비바도 세션이 진행되는 동안 실행된 Tcl 명령어만 포함하고 있으며, 로그 파일에 기록된 부가적인 상세정보는 없다. 저널 파일은 디자인 세션 이전에 Tcl 스크립트를 생성할 때 유용하며, 다음 lab에서 확인할 수 있다.

² `vivado.log` 및 `vivado.jou`는 `%APPDATA%\Xilinx\vivado`, 혹은 `/home` 디렉토리에 작성할 수 있다.

Lab 2: 프로젝트 디자인 플로우 이용하기

이 lab에서는 프로젝트 생성, 소스 파일 관리, 디자인 분석, Constraints 정의, 합성 및 구현 실행 관리를 위한 프로젝트 모드 기능들을 배우게 될 것이다.

비바도를 시작하고, 예제 디자인을 이용해 FPGA 디자인 플로우 전반을 살펴보게 될 것이다. 그런 다음 IDE의 일부 주요 기능들을 검토할 것이다. 대부분의 기능들은 다른 지침서에서 보다 상세하게 다루고 있다. 마지막으로, 디자인 프로젝트를 구현하기 위한 배치 실행 스크립트를 생성하고, 얼마나 쉽게 Tcl 스크립트 실행과 비바도 IDE 작업을 전환할 수 있는지 확인할 것이다.

Step 1: 프로젝트 생성

비바도 시작

- 리눅스의 경우,
 1. lab 자료들이 저장된 디렉토리를 변경한다:


```
cd <Extract_Dir>/Vivado_Tutorial
```
 2. 비바도 IDE 시작:


```
vivado
```
- 윈도우에서 비바도 툴을 시작하는 데스크톱 아이콘을 클릭하기 전에, `vivado.log` and `vivado.jou` 파일을 어디에 작성할 것인지를 보여주는 아이콘을 컨피규레이션한다.
 1. 비바도 2014.x Desktop 아이콘을 우측 클릭하고, 팝업 메뉴에서 **Properties** 를 선택한다.
 2. [그림 9](#)에 나타난 것처럼, 추출된 비바도 사용 지침서 디렉토리로 **Start in** 값을 설정한다.


```
<Extract_Dir>/Vivado_Tutorial/
```
 3. **OK** 를 클릭하고 Properties 대화상자를 종료한다. .

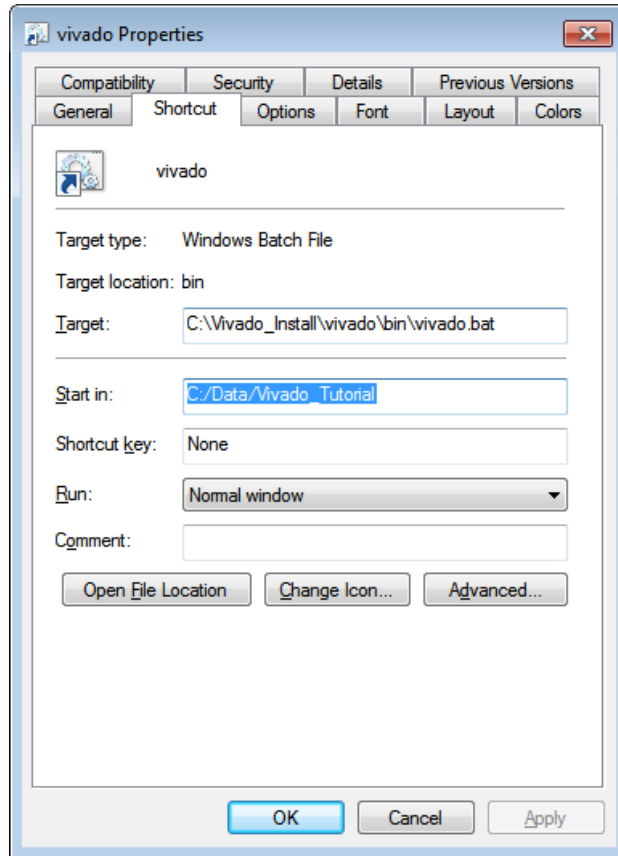


그림 9: Start in 디렉토리 컨피규레이션

4. 비바도 IDE 를 시작하기 위해 **Vivado 2014.x** Desktop 아이콘을 클릭한다.

새로운 프로젝트 생성

1. 비바도를 오픈한 후, Getting Started 페이지에서 **Create New Project** 를 선택한다.
2. New Project 마법사에서 **Next** 를 클릭한다.
3. Project Name 및 Location 을 지정한다:
4. Project name: **project_bft**
5. Project Location: `<Extract_Dir>/Vivado_Tutorial/Tutorial_Created_Data`
6. **Next** 를 클릭한다.

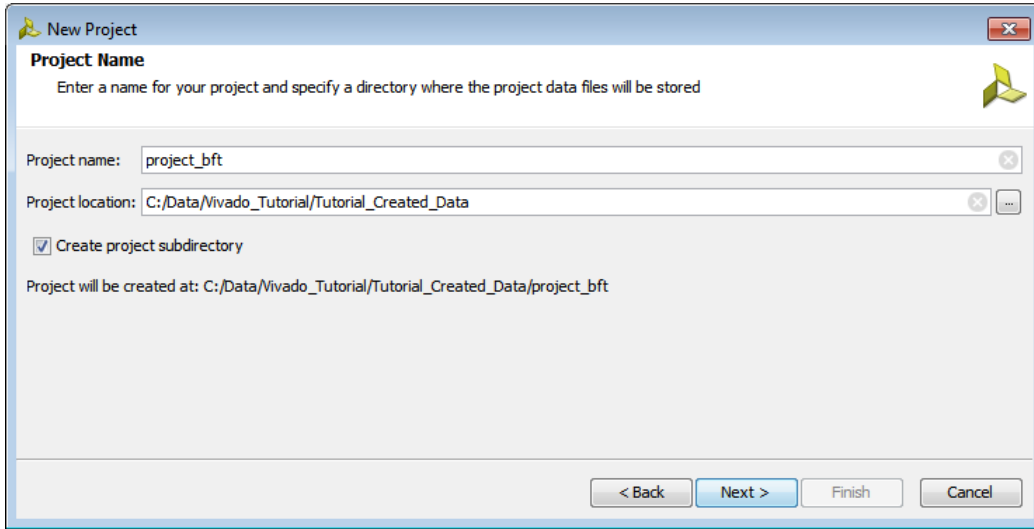


그림 10: 새로운 프로젝트 생성

7. **Project Type** 으로 **RTL Project** 를 선택하고, **Next** 를 클릭한다.
8. Add Files 을 클릭한다...
 - a. **<Extract_Dir>/Vivado_Tutorial/Sources/hdl/**를 찾는다.
 - b. **Ctrl** 키를 누른 상태로 아래의 파일을 선택, 클릭한다:
async_fifo.v, bft.vhdl, bft_tb.v, FifoBuffer.v
 - c. **OK**를 클릭하고 File Browser를 종료한다.
9. Add Directories 를 클릭한다...
 - a. **<Extract_Dir>/Vivado_Tutorial/Sources/hdl/bftLib** 디렉토리 선택하고,
 - b. **Select** 버튼을 누른다.
10. 아래 나타낸 것처럼, **bft_tb.v** 를 위해 **HDL Sources for** 컬럼을 클릭하고, Synthesis and Simulation 을 **Simulation only** 로 변경한다.
11. [그림 11](#) 에 나타낸 것처럼, **bftLib** 를 위해 **Library** 컬럼을 클릭하고, **work** 에서 **bftLib** 으로 변경하기 위해 값을 수동으로 편집한다.

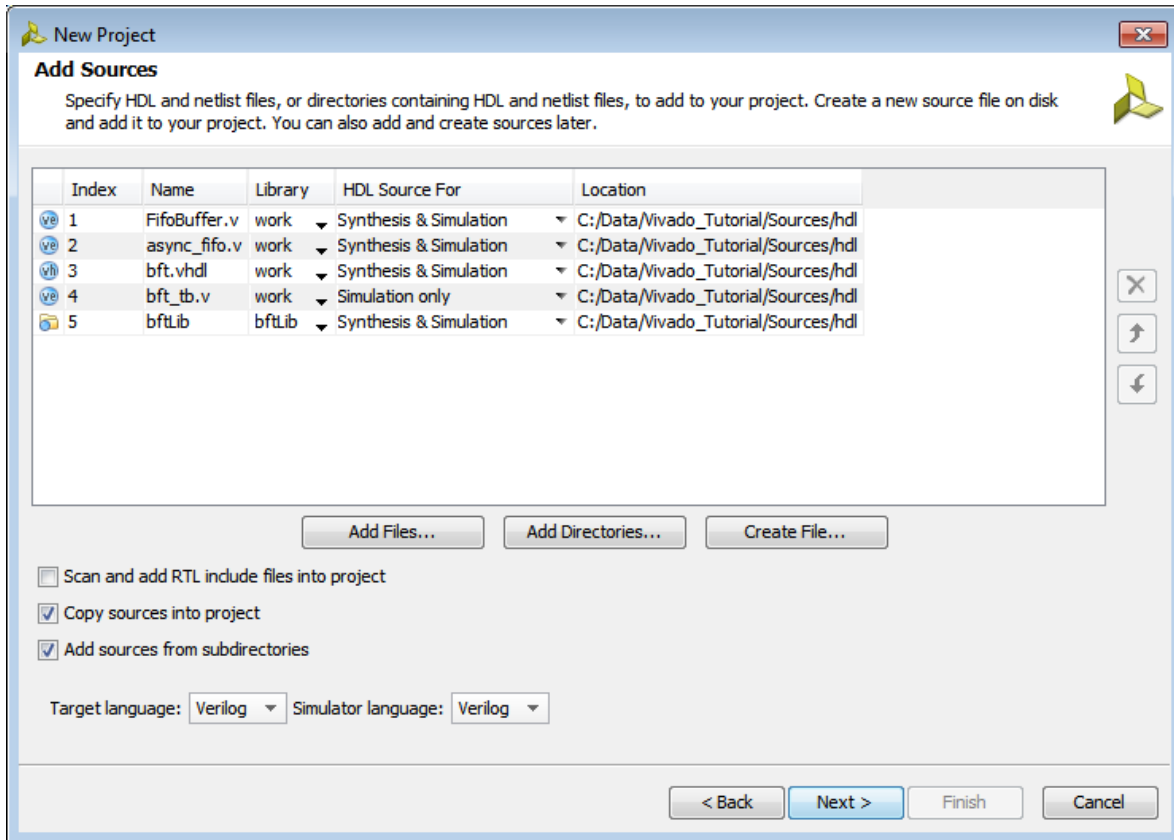


그림 11: RTL 소스 추가

12. **Copy sources into project** 및 **Add sources from subdirectories** 체크박스에 체크한다.
13. 비바도 합성으로 생성된 넷리스트 언어를 정의하기 위해 Target Language 를 **Verilog** 로 설정한다.
14. 로직 시뮬레이터에서 요구되는 언어를 정의하기 위해 Simulator Language 를 **Verilog** 로 설정한다.
15. **Next** 를 클릭한다.
16. 여기에서 IP 를 추가하지 않는다면, Add Existing IP 페이지를 건너뛰기 위해 다시 **Next** 를 클릭한다.
17. Add Constraints 페이지에서 **Add Files...**을 클릭한다.
18. `<Extract_Dir>/Vivado_Tutorial/Sources/bft_full.xdc` 를 찾아 선택한다.
19. **OK** 를 클릭하고 File Browser 를 닫는다.
20. Constraints 파일을 프로젝트 안에 Copy 하기 위해 체크박스를 체크한다.
21. Default Part 페이지로 이동하기 위해 **Next** 를 클릭한다.

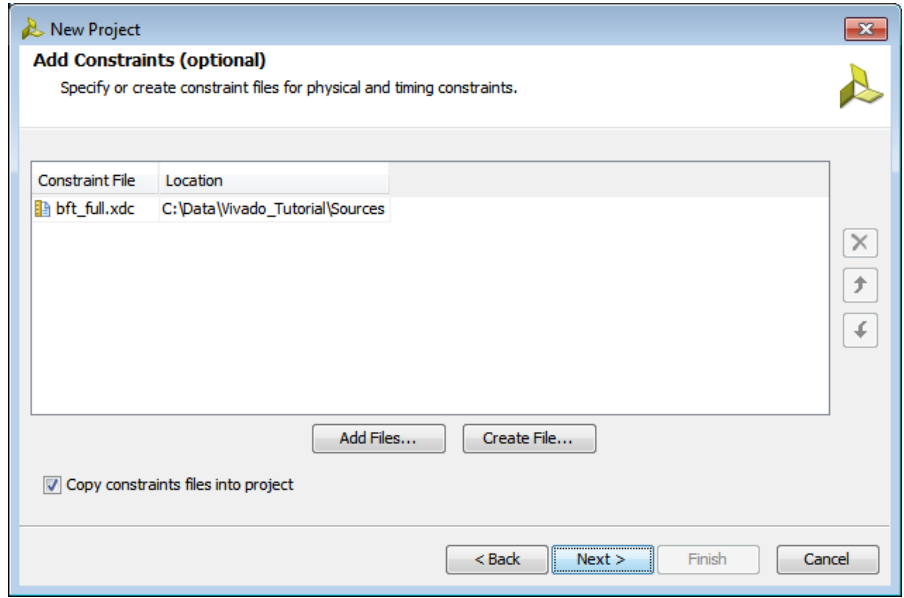


그림 12: Constraints 추가

- 22. Default Part 페이지에서 **Family** 필터를 클릭하고, **킨텍스-7** 제품군을 선택한다.
- 23. 리스트 상단으로 스크롤해서 **xc7k70tfg484-2** 부품을 선택하고, **Next** 를 클릭한다.
- 24. New Project Summary 페이지를 닫기 위해 **Finish** 를 클릭하고, 프로젝트를 생성한다.

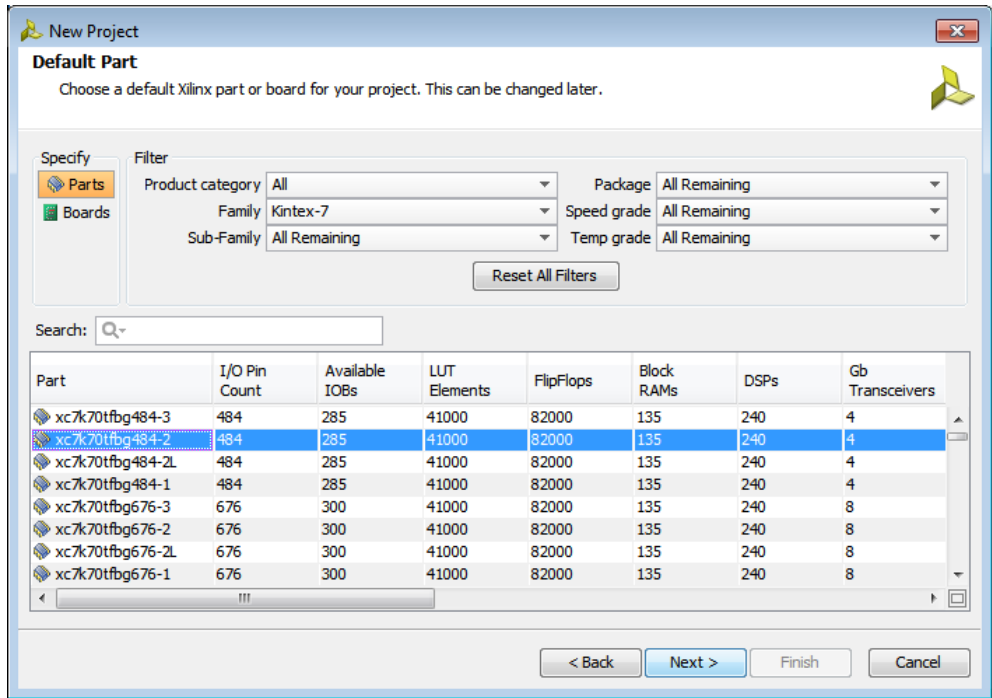


그림 13: Default Part 선택

비바도 IDE는 디폴트 레이아웃에서 project_bft를 오픈한다.

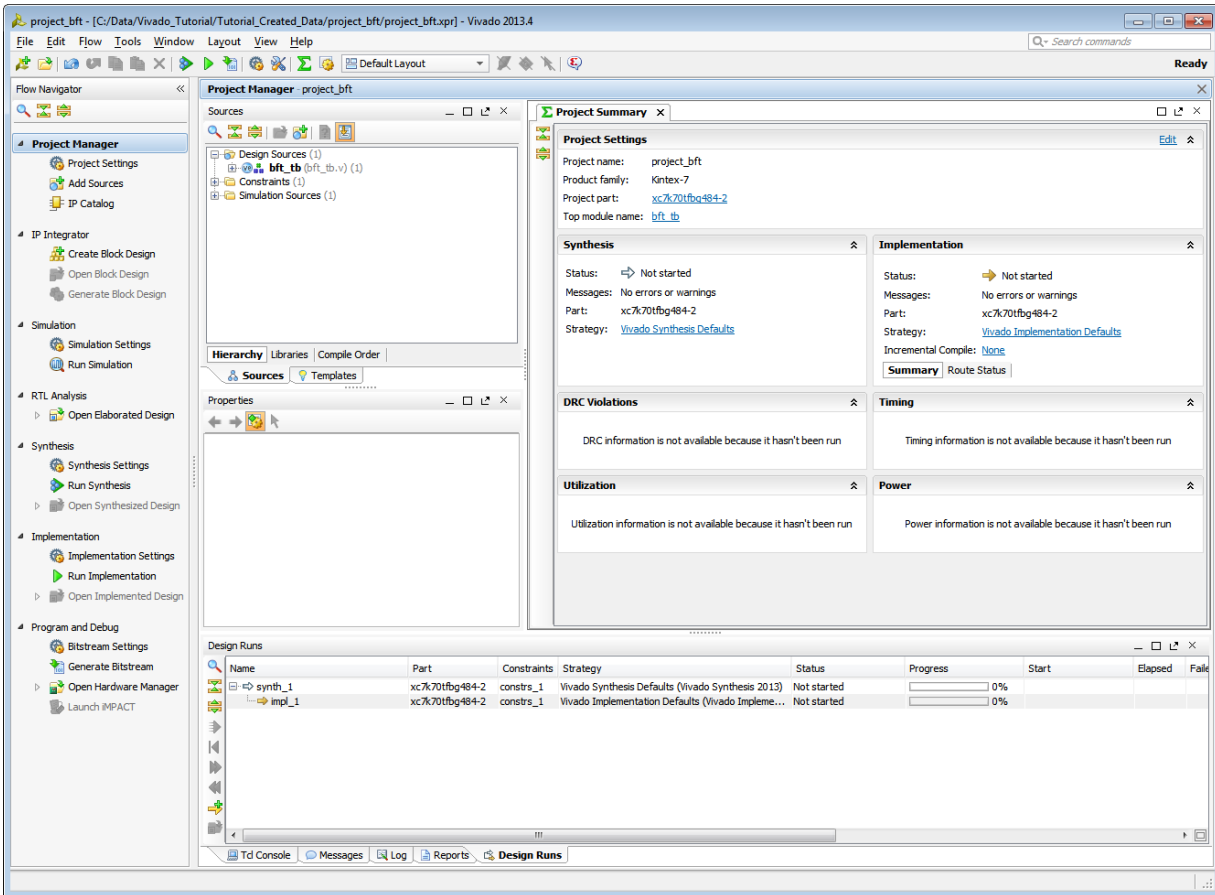


그림 14: 비바도 IDE의 프로젝트 BFT

Step 2: Sources Window 및 Text Editor 이용하기

비바도 툴은 Verilog, VHDL, EDIF, NGC 포맷의 코어나 SDC, XDC, TCL Constraints 파일, 시뮬레이션 테스트 벤치를 비롯한 각기 다른 디자인 소스를 추가할 수 있도록 해준다. 이러한 파일은 Sources 하단에 있는 Hierarchy, Library, Compile Order와 같은 탭을 이용해 여러 방법으로 분류가 가능하다.

비바도 IDE는 RTL 소스 및 Constraints 파일, Tcl 스크립트를 생성, 개발하기 위해 컨텍스트 센서티브 text editor를 포함하고 있다. 또한 써드파티 text editor를 사용하기 위해 비바도 IDE를 컨피규레이션할 수 있다. 비바도 툴 컨피규레이션에 대한 정보는 *비바도 디자인 수트 사용자 가이드: IDE 이용하기*(UG893)에서 확인할 수 있다.

Window 및 Project Summary 살펴보기

1. **Project Summary** 의 정보를 검토한다. 보다 상세한 정보는 디자인 플로우 전반에 걸쳐 디자인이 진행되면서 제공된다.
2. Sources 창을 검토하고, **Design Sources, Constraints, Simulation Sources** 폴더를 확장한다.

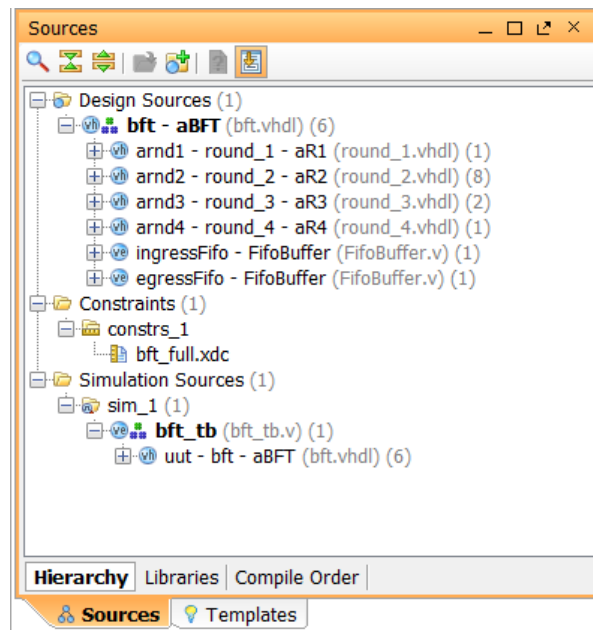


그림 15: Sources 보기

Design Sources 폴더는 VHDL 및 Verilog 소스 파일 및 라이브러리를 지속적으로 추적하는데 도움을 준다. **Hierarchy** 탭은 디폴트로 디스플레이된다.

3. Sources 창에서 **Libraries** 탭 및 **Compile Order** 탭을 선택한다. Sources 는 각기 다른 방식으로 분류되어 있다.

Libraries 탭은 파일 타입으로 소스 파일을 분류하고 있다. Compile Order 탭은 합성에 사용되는 파일 순서를 보여준다.

4. **Hierarchy** 탭을 선택한다.

Text Editor 살펴보기

1. Sources 창에서 **VHDL** 소스 중 하나를 선택한다.
2. **팝업 메뉴**에서 이용 가능한 명령어를 리뷰하기 위해 **우측 클릭**한다.
3. **Open File** 을 선택하고, Text Editor 에 있는 파일 콘텐츠를 찾기 위해 스크롤 바를 이용한다.

또한 Text Editor에서 이를 열기 위해 Sources 창에서 소스 파일을 더블 클릭할 수 있다.

```

40 use bftLib.bftPackage.all;
41
42 entity round_1 is
43     port (
44         clk: in std_logic;
45         x : in xType;
46         xOut : out xType
47     );
48 end entity round_1;
49
50 architecture aR1 of round_1 is
51     constant u : uType :=
52         (X"0123",
53          X"4567",

```

그림 16: 컨텍스트 센서티브 Text Editor

Text Editor는 키워드 및 명령어가 컬러로 되어있는 컨텍스트 센서티브로 RTL 코드를 디스플레이한다. 예약어를 디스플레이하기 위해 사용된 컬러 및 폰트는 **Tools > Options** 명령어로 설정이 가능하다. 보다 자세한 정보는 *비바도 디자인 수트 사용자 가이드: IDE 이용하기(UG893)*에서 확인할 수 있다.

4. Text Editor 에서 커서를 이용해 우측 클릭하고, **Find in Files** 을 선택한다. 또한 Files 명령어에서 Replace 한다.

Find in Files 대화상자는 다양한 검색 옵션으로 시작된다.

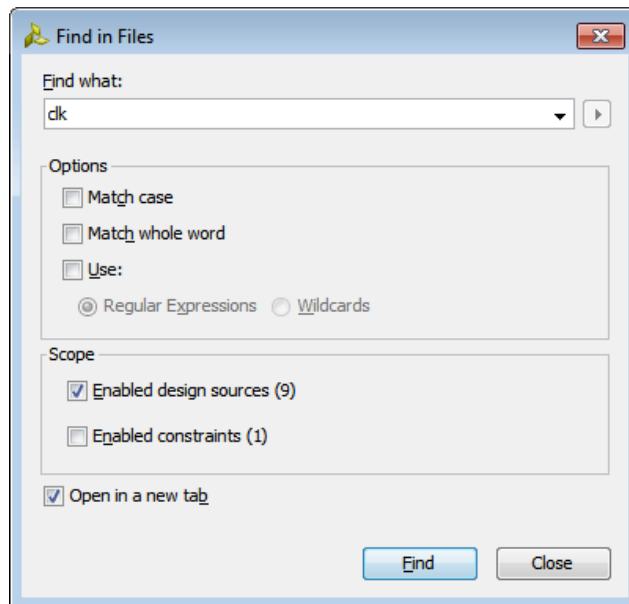


그림 17: Find in Files 이용하기

5. **Find what** 에 **clk** 를 입력하고 **Find** 를 클릭한다.

Find in Files 창은 비바도 IDE 하단에 메시징 영역을 디스플레이한다.

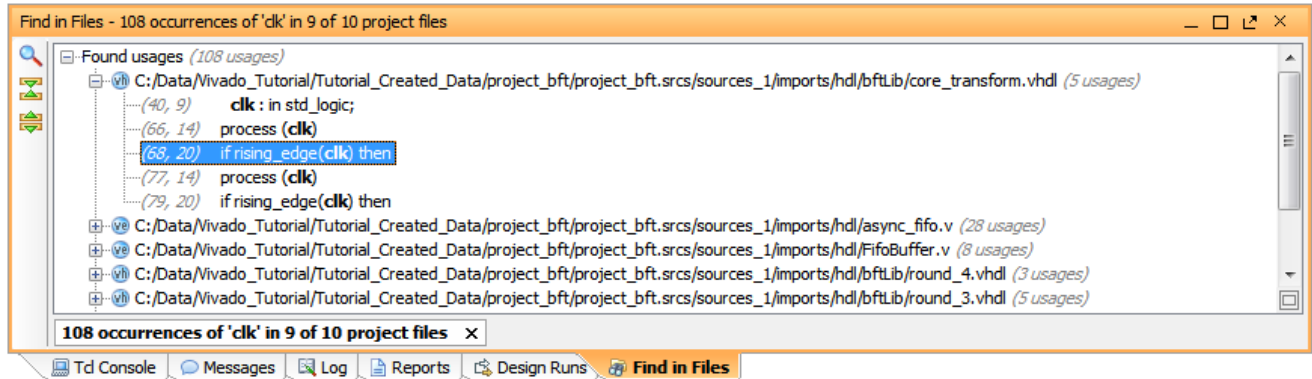


그림 18: Find in Files 결과 보기

6. Find in Files 창에서 디스플레이된 파일 중 하나를 확장하고, 파일에서 **clk** 가 나타난 항목을 선택한다.

Text Editor는 선택된 파일을 열고, 파일에서 선택된 **clk**가 있는 항목을 디스플레이한다.

7. Find in Files 의 Occurrences 창을 종료한다.

8. 열려 있는 **Text Editor** 창을 종료한다.

다음 몇 단계에서는 합성을 구동하기 전에 수행 가능한 일부 디자인 컨피규레이션 및 분석 기능을 하이라이트 할 것이다.

Step 3: RTL 디자인 살펴보기

비바도 IDE는 RTL 분석 및 IP 커스터마이징 환경을 포함하고 있다. 또한 RTL 디자인 상에서 성능 및 전력소모를 개선하는 방법을 검토할 수 있는 여러 RTL DRC(Design Rule Checks)도 있다.

1. 디자인을 세밀하게 살펴보기 위해 Flow Navigator 에서 **Open Elaborated Design** 을 선택한다.

2. 메인 툴바의 Layout Selector 풀다운 메뉴에서 **Default Layout** 을 선택할 수 있다.

Elaborated Design은 RTL Netlist, Schematic, Graphical Hierarchy를 비롯한 다양한 분석 뷰를 실행할 수 있다. 이 뷰에는 RTL을 디버깅하고 최적화하는데 도움을 주는 "cross-select" 기능이 있다.

3. RTL Netlist 창에서 로직 계층을 살펴보고, Schematic 을 검토한다.

계층 내부를 보기 위해 셀을 더블 클릭하거나, Schematic 팝업 메뉴에서 **Expand Cone**이나 **Expand/Collapse**와 같은 명령어를 이용하여 스케매틱을 살펴볼 수 있다. Schematic 창을 이용하기

위한 보다 상세한 정보는 *비바도 디자인 수트 사용자 가이드: 비바도 IDE 이용하기(UG893)*에서 확인할 수 있다.

4. Schematic 에서 로직 인스턴스 중 하나를 선택하고, 우측 클릭하여 **Go to Source** 나 **Go to Definition** 명령어를 선택한다.

Text Editor는 로직 인스턴스에 해당하는 셀을 선택하고, 하이라이트하기 위해 RTL 소스 파일을 엽니다. **Go to Definition** 명령어의 경우, 모듈 정의를 포함하고 있는 RTL 소스 파일을 엽니다. **Go to Source**로는 선택된 셀의 인스턴스를 포함하고 있는 RTL 소스를 엽니다.

5. 비바도 IDE 하단의 Messages 창을 클릭하고, 메시지를 검토한다.
6. 메시지 툴바에서 **Collapse All** 버튼을 클릭한다.
7. Elaborated Design 메시지를 확장한다.

메시지와 결부된 RTL 소스 파일을 열기 위한 메시지에는 링크가 들어 있다.

8. 링크 중 하나를 클릭하면, Text Editor 는 해당 라인에 하이라이트된 RTL 소스 파일을 엽니다.
9. Text Editor 창을 종료한다.
10. Elaborated Design 창 배너 우측의 **X** 를 클릭하여 **Elaborated Design** 을 종료하고, 확인을 위해 **OK** 를 클릭한다.

Step 4: IP 카탈로그 이용하기

자일링스 IP 카탈로그는 비바도 IP 컨피규레이션 및 생성 기능에 액세스할 수 있도록 해준다. 다양한 방법으로 카탈로그를 분류하고 검색할 수 있다. IP는 커스터마이징하거나 생성 및 예시화가 가능하다.

1. Flow Navigator 에서 **IP Catalog** 버튼을 클릭한다.
2. IP **Catalog** 를 찾아서, 여러 카테고리 및 IP 필터링 성능을 검토한다.
3. **Basic Elements** 폴더를 확장한다.
4. **DSP48 Macro** 를 더블 클릭한다.

Customize IP 대화상자를 비바도 내에서 직접 엽니다. 비바도 디자인 수트는 툴 내에서 IP 커스터마이징 및 컨피규레이션을 수행할 수 있도록 해준다. IP 컨피규레이션 및 구현에 대한 보다 자세한 정보는 *비바도 디자인 수트 사용자 가이드: IP 디자인(UG896)* 및 *비바도 디자인 수트 사용자 지침서: IP 디자인(UG939)*에서 확인할 수 있다.

5. 현 디자인에 IP 를 추가하지 않고 Customize IP 대화상자를 종료하려면 **Cancel** 을 클릭한다.
6. 윈도우 배너 우측의 **X** 를 클릭하여 **IP Catalog** 를 종료한다.

Step 5: Behavioral Simulation 실행하기

비바도 IDE는 프로젝트에서 시뮬레이션 소스를 추가하고 관리할 수 있도록 해주는 Vivado Simulator를 통합하고 있다. 이를 통해 시뮬레이션 옵션을 컨피규레이션하고, 시뮬레이션 소스 세트를 생성하고 관리할 수 있다. 또한 합성 이전에 RTL 소스 상에서 Behavioral Simulation을 실행할 수 있다.

1. **Flow Navigator** 에서 시뮬레이션 내의 **Simulation Settings** 명령어를 클릭한다.

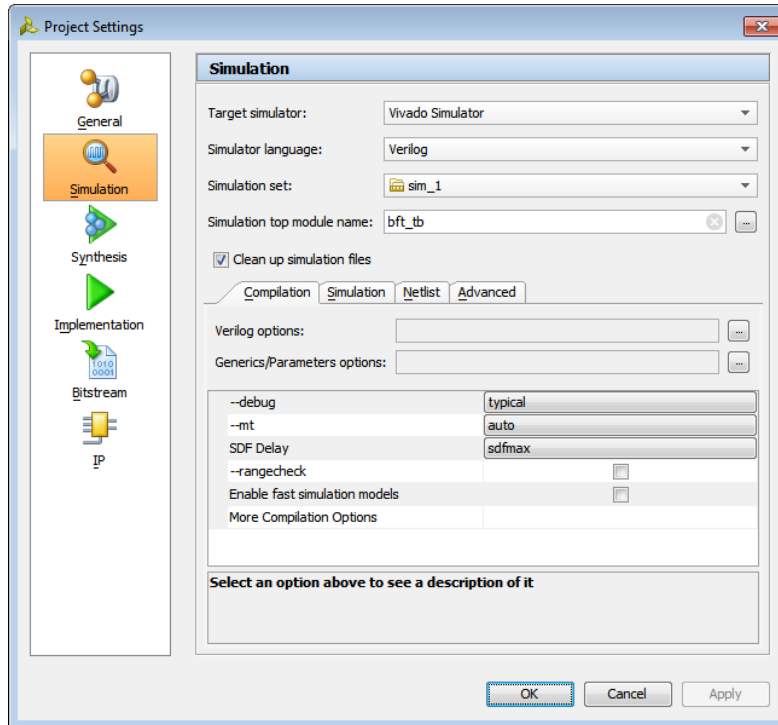


그림 19: Simulation Settings - 업데이트

2. Compilation, Simulation, Netlist, Advanced 등의 **각 탭** 내에서 이용 가능한 설정을 **살펴본** 다음, 대화상자를 종료하기 위해 **Cancel** 을 클릭한다.
3. Flow Navigator 에서 **Run Simulation** 명령어를 클릭한 다음, 서브-메뉴에서 **Run Behavioral Simulation** 을 클릭한다.
4. **Simulation** 환경을 **검토**하고, 분석한다.

시뮬레이션은 *비바도 디자인 수트 사용자 가이드: 로직 시뮬레이션(UG900)* 및 *비바도 디자인 수트 사용 지침서: 로직 시뮬레이션(UG937)*에서 보다 상세하게 다루고 있다.

5. Simulation 뷰 배너의 **X** 아이콘을 클릭하여 **Simulation** 을 **종료**한다.
6. 즉시 변경을 저장한다면, **No** 를 클릭한다.

Step 6: 디자인 실행 설정 검토하기

Lab #1에서 사용한 비-프로젝트 모드와 현재 사용하고 있는 프로젝트 모드의 가장 크게 다른 점 중 하나는 합성 및 구현을 위한 디자인 실행 지원 여부이다. 비-프로젝트 모드는 디자인 실행을 지원하지 않는다.

디자인 실행은 합성 및 구현 프로세스의 각기 다른 단계에서 이용할 수 있는 여러 옵션을 설정하고, 저장할 수 있는 방법이다. 여러분은 이러한 옵션을 설정하거나, 향후 디자인 실행 사용 전략에 대한 컨피규레이션을 저장할 수 있다. 또한 프로세스의 각 단계 사전 사후에 실행하거나, 디자인 절차별로 사전 사후에 리포트를 생성하기 위해 Tcl.pre 및 Tcl.post 스크립트를 정의할 수 있다.

합성 및 구현 실행을 시작하기 전에, 이러한 실행 설정 및 전략을 검토할 수 있다.

1. Flow Navigator 에서, Synthesis 내의 **Synthesis Settings** 을 선택한다.

Project Settings 대화상자를 연다. Synthesis Settings은 비바도 합성을 설정하는데 이용할 수 있는 여러 옵션에 액세스할 수 있도록 해준다. 이러한 옵션에 대한 총체적인 설명은 *비바도 디자인 수트 사용자 가이드: 합성(UG901)*에서 확인할 수 있다.

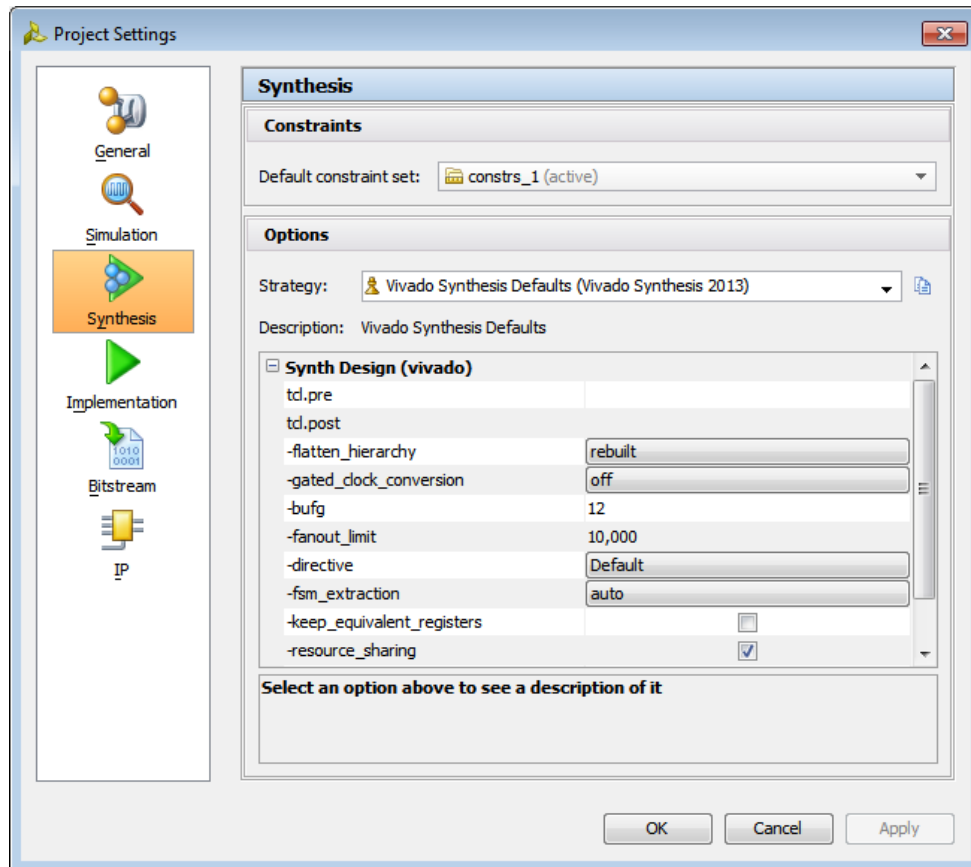


그림 20: 합성 설정

2. 합성 옵션을 검토한 후, [그림 20](#)에 나타난 것처럼, Project Settings 대화상자 좌측의 **Implementation** 버튼을 선택한다.

Project Settings은 Implementation 설정을 반영하여 변경된다. 구현 실행을 위해 이용할 수 있는 옵션을 확인할 수 있다. 이러한 옵션에 대한 총체적인 설명은 *비바도 디자인 수트 사용자 가이드: 구현(UG904)*에서 확인할 수 있다.

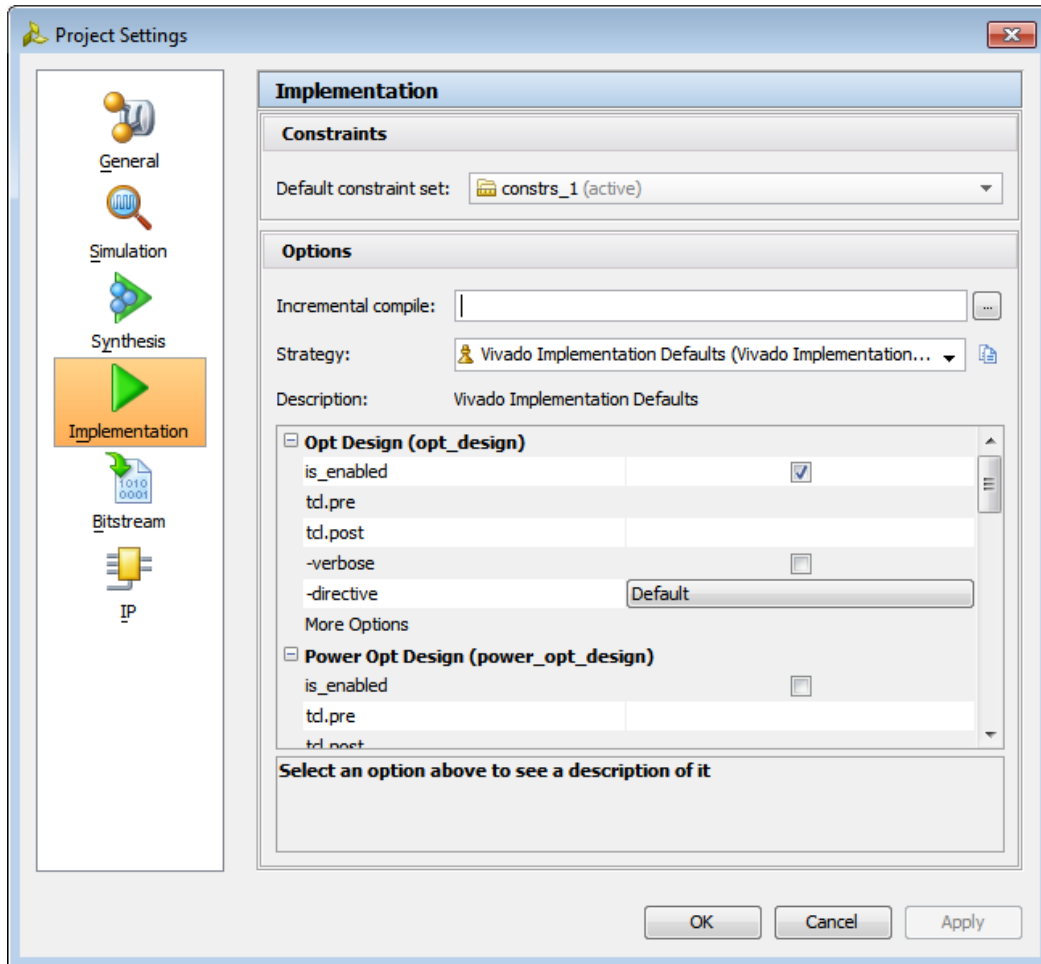


그림 21: 구현 설정

3. Project Settings 대화상자를 종료하기 위해 **Cancel** 을 클릭한다.

이제 비바도 합성 및 구현을 시작할 준비가 완료되었다.

Step 7: 디자인 합성 및 구현

합성 및 구현 실행 옵션을 설정한 후, 다음을 처리할 수 있다:

- 합성 만 실행하기 위해서는 **Run Synthesis** 명령어를 이용한다.
- 실행되지 않았다면 합성을 먼저 실행한 다음, 구현이 실행되도록 **Run Implementation** 명령어를 이용한다.
- 먼저 합성을 실행한 다음, 실행되지 않은 구현을 실행하고, 자일링스 디바이스를 프로그래밍하기 위한 비트스트림을 작성하기 위해 **Generate Bitstream** 명령어를 이용한다.

이 지침서에서는 이러한 단계를 한번에 하나씩 실행할 것이다.

1. Flow Navigator 에서, **Run Synthesis** 버튼을 클릭하고, 이 작업이 완료되기를 기다린다.

비바도 IDE의 상단 우측 코너에 있는 진행 바는 진행되는 실행을 보여준다. 비바도는 틀이 다른 작업을 할 수 있도록 백라운드 프로세스에서 합성 엔진을 시작한다. 합성 프로세스가 백그라운드에서 실행되는 동안, 여러분은 계속해서 비바도 IDE 창을 브라우징하고, 리포트를 실행하고, 다른 디자인 평가를 진행할 수 있다. Log 창은 IDE 하단에 합성 로그를 디스플레이한다. 또한 Reports 창을 통해서도 이용할 수 있다.

합성이 완료된 후, Synthesis Completed 대화상자가 다음 단계를 선택할 수 있도록 나타난다.

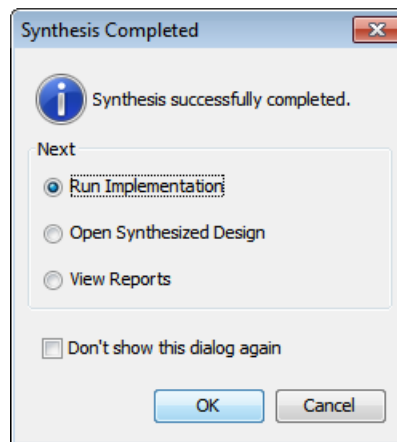


그림 22: 합성 완료

2. **Run Implementation** 을 선택하고, **OK** 를 클릭한다.

일부 초기화 이후 구현 프로세스가 백그라운드 프로세스로 배치되고 시작된다. 이 지침서의 다음 단계에서는 여러분이 구현이 완료되기를 기다리는 동안 어떻게 합성된 디자인의 디자인 분석을 수행할 수 있는지 보여줄 것이다.

Step 8: 합성된 디자인 분석

합성된 디자인을 열어 디자인 분석, Timing Constraints 정의, I/O 플래닝, 플로어플래닝, 디버깅 코어 삽입을 수행할 수 있다. 이러한 기능에 대한 설명은 다른 지침서에서 다루고 있지만, 이 단계에서 간단히 살펴볼 수 있다.

1. Flow Navigator 에서 **Open Synthesized Design** 을 선택하고, 로드할 디자인을 기다린다.

비바도 IDE는 합성된 디자인을 열고, 구현은 백그라운드에서 계속해서 실행된다. 합성된 디자인을 검토하는 동안, 구현이 완료될 것이며, Implementation Completed 대화상자가 다른 단계를 선택할 수 있도록 나타난다.

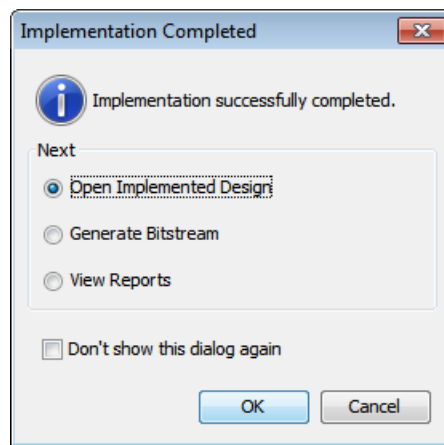


그림 23: 구현 완료

2. 아무런 액션도 취할 필요없이 **Cancel** 을 클릭하고 대화상자를 닫는다.

이 상태에서 합성된 디자인은 계속 열려있다. 합성된 디자인의 기능에 대한 검토가 완료되면, 구현된 디자인을 엽니다.

3. 메인 툴바의 Layout Selector 풀다운 메뉴에서 **Default Layout** 을 선택한다.
4. 비바도 IDE 의 하단에 있는 **Reports** 윈도우 탭을 클릭한다.
Reports 창이 열리지 않으면, **Windows > Reports**로 열 수 있다.
5. 리포트를 검토하기 위해 **Vivado Synthesis Report** 를 더블 클릭한다.
6. 리포트를 검토하기 위해 **Utilization Report** 를 더블 클릭한다.
7. 이에 대한 검토가 끝나면, **모든 리포트를 닫는다**.
8. 비바도 IDE 의 하단에 있는 **Messages** 윈도우 탭을 클릭한다

Messages 창이 열리지 않으면, **Windows > Messages**로 열 수 있다.

Messages 창은 Errors, Critical Warnings, Warnings, Info, Status 등과 같은 숨겨진 각기 다른 메시지 타입을 디스플레이하기 위해 배너에 메시지 타입 필터를 제공한다.

9. 모든 Messages 를 압축하기 위해 **Collapse All** 버튼을 클릭한다.
10. **Synthesis** 메시지를 확장한다.
11. Synthesis 메시지 전체를 스크롤하면, 소스 파일 내의 특정 링크와 연결되어 있는 것을 알 수 있다. 이 링크들 중 일부를 클릭하면, Text Editor 내에서 적합한 라인으로 하이лай트된 소스 파일을 오픈할 수 있다.

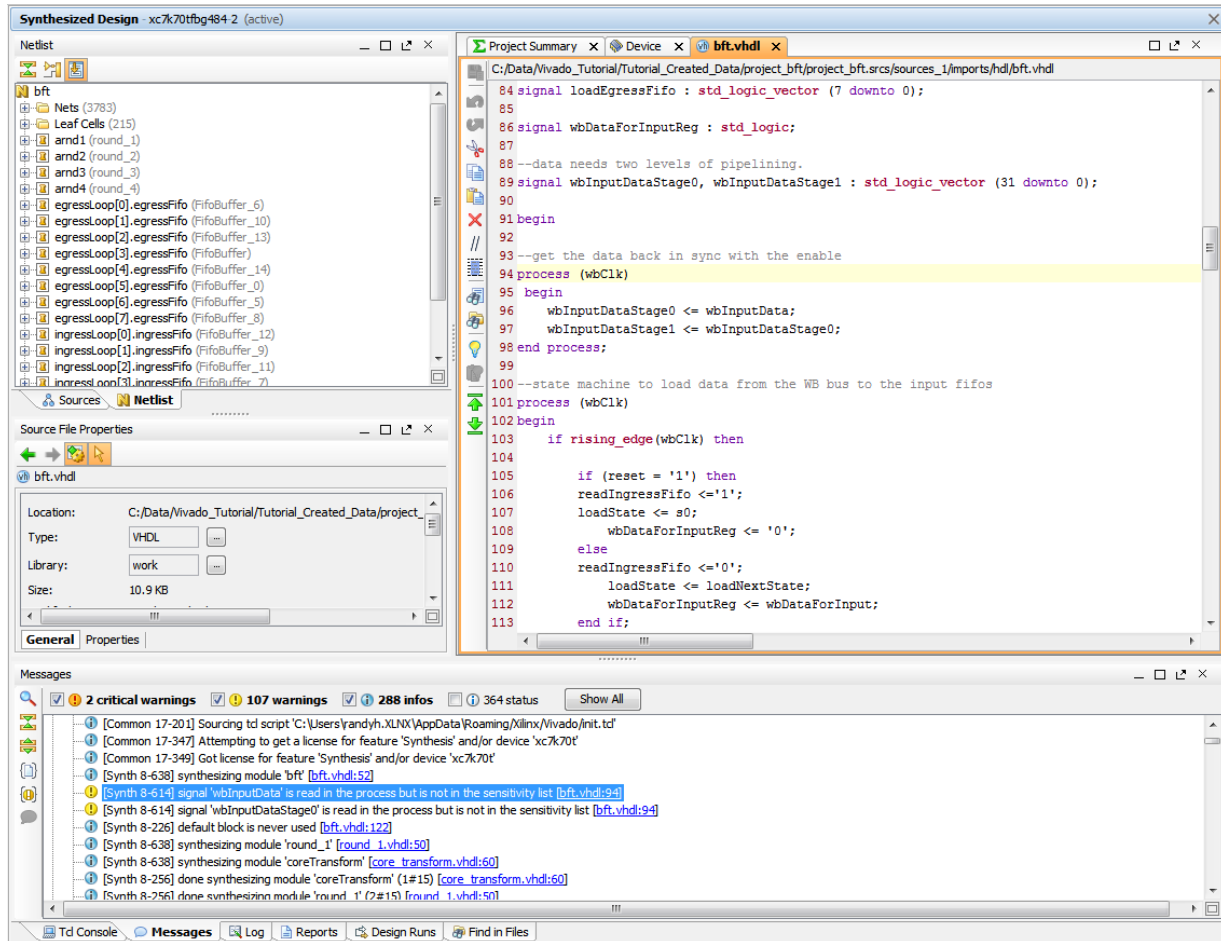


그림 24: 소스 파일과 연결된 합성 메시지

12. Flow Navigator 에서, Synthesized Design 아래의 **Report Timing Summary** 를 선택한다. Report Timing Summary 대화상자가 열린다. 이 명령어의 여러 필드와 옵션들을 검토한다.
13. 디폴트 옵션으로 실행되도록 **OK** 를 클릭한다.
Timing Summary Results 창을 연다.

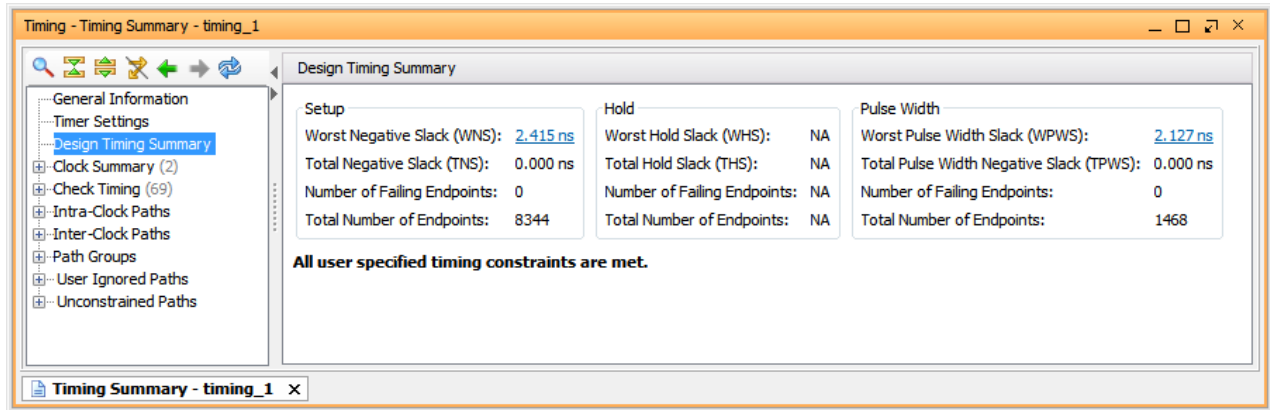


그림 25: Report Timing Summary

구현 이전에 예측한 타이밍을 보여주는 Timing Summary Results 창을 검토한다. Timing Summary Results 창의 좌측에 있는 트리에서 리포팅 카테고리 하나를 클릭한다.

14. Flow Navigator 에서 **Report Power** 를 선택한다.

Report Power 대화상자를 엿는다. 이 명령어의 다양한 필드와 옵션을 검토한다.

15. 디폴트 옵션으로 실행되도록 **OK** 를 클릭한다.

Power Results 창을 엿는다. 구현 이전에 예측된 전력을 보여주는 Power Results 창을 검토한다.

리포트는 역동적이며, 리포트에 마우스를 이동하면, [그림 26](#)에 나타난 것처럼, 리포트의 특정 섹션에 대한 상세정보를 제공하는 툴팁을 갖추고 있다.

제공되는 서로 다른 정보를 검토하기 위해 Power Results 창 좌측의 트리에서 리포팅 카테고리 중 일부를 클릭한다.

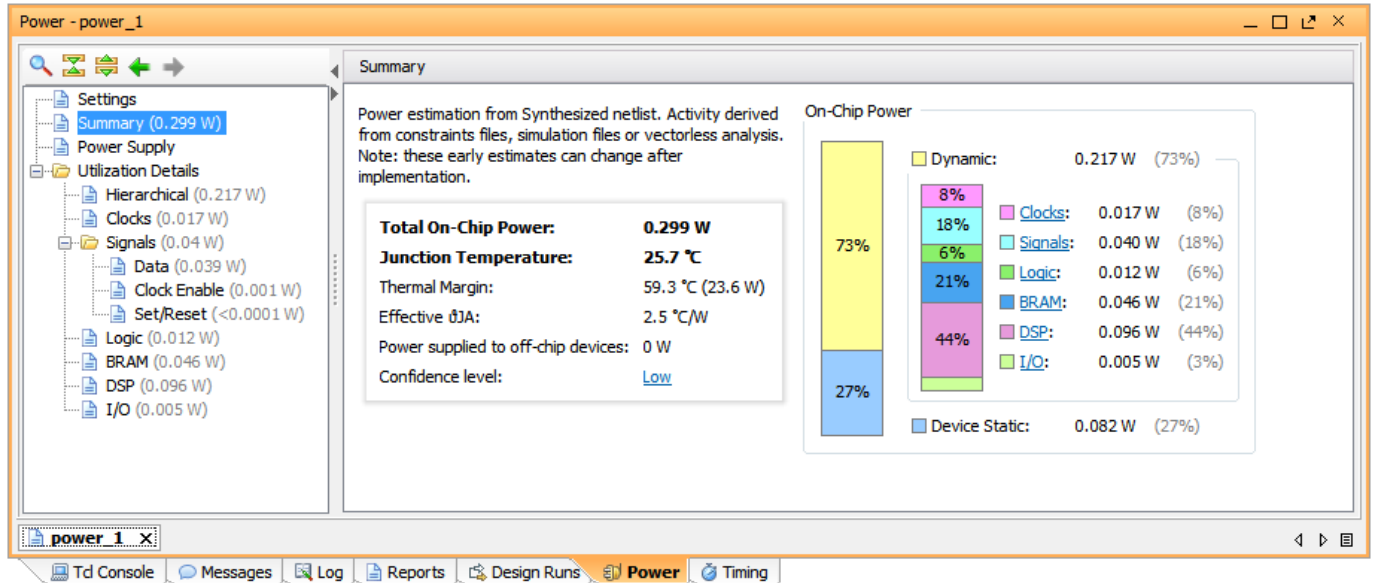


그림 26: 전력 리포트

16. **Report Timing Summary** 창과 **Power Report** 창, 그리고 열려있는 모든 **Text Editor** 창을 종료한다.


Step 9: 구현된 디자인(Implemented Design) 분석

비바도 IDE는 쌍방향으로, 인-메모리 디자인 상의 디자인 Constraints 및 넷리스트를 편집할 수 있다. 디자인을 저장하면, 변경된 Constraints은 원래의 XDC 소스 파일에 작성된다. 반면, 변경내용을 원래의 Constraints을 유지한 상태로 새로운 Constraints 파일에 저장할 수도 있다. 이러한 유연성은 플로어플래닝을 비롯한 교체된 타이밍 및 물리적 Constraints을 검토할 수 있도록 해주며, 원래의 소스 파일 또한 그대로 유지할 수 있다.

구현된 디자인 오픈

1. Flow Navigator 에서 **Open Implemented Design** 을 선택한다.
2. 합성된 디자인을 종료하기 위해 **Yes** 를 선택하고, **저장은 하지 않는다**.
Implemented Design에 로드한 후, Device 창에서 구현 결과를 확인할 수 있다.
3. 비바도 IDE 하단에 있는 Reports 윈도우 탭을 클릭한다.
Reports 창이 열리지 않으면, **Windows > Reports**로 열 수 있다. Place Design 및 Route Design에서 일부 리포트를 선택하고 검토한다. 검토했다면 각 리포트를 닫는다.
4. IDE 하단에 있는 **Messages** 윈도우 탭을 선택한다.


만약 Messages 창이 열리지 않는다면, **Windows > Messages**로 열 수 있다.

- 모든 Messages 를 압축하기 위해 **Collapse All** 버튼을 클릭한다. 
- Implementation** 폴더를 확장한다.

Design Initialization, Opt_Design, Place_Design, Route_Design에서 메시지를 확인한다.

라우팅 분석

디자인의 P&R이 완료된 후, 모든 Timing Constraints에 부합하는지 검증하기 위해 타이밍 리포트를 생성할 수 있다. Device 창에서 라우팅 경로를 검토하기 위해 Timing Report 창에서 경로를 선택할 수 있다. 타이밍 문제가 있다면, 문제를 해결하기 위해 RTL 소스 파일이나 디자인 Constraints을 다시 확인할 수 있다.

- Device 창에서, 디바이스 라우팅을 디스플레이하기 위해 **Routing Resources** 버튼  을 선택한다.

이는 Device 창에서 라우팅 커백션을 확인할 수 있게 해준다. 세밀하게 라우팅 요소를 확인하기 위해 디바이스 내부를 확대하거나 전반적인 라우팅을 보기 위해 줌아웃할 수 있다.

- 비바도 IDE 가 자동으로 선택된 객체를 줌인하거나 중심을 맞출 수 있도록 Device 윈도우 툴바 메뉴에서 **Auto Fit Selection** 버튼  을 선택한다.
- Flow Navigator 에서, Implemented Design 내의 **Report Timing Summary** 버튼을 클릭한다.
- 디폴트 리포트를 생성하기 위해 Report Summary Timing 대화상자에서 **OK** 를 클릭한다.

- Timing Summary Results 창의 좌측 패널에서, 다음을 선택:

Intra-Clock Paths > bftClk > SETUP...

Timing Summary Report 창 우측의 테이블 보기에서, Device 창에서 선택하고 하이라이트할 타이밍 경로를 클릭한다. Timing Summary 창에서 여러 경로를 선택하고 라우팅 경로를 검토한다.

- Timing Summary Results 창 좌측 패널에서, 다음을 선택:

Intra-Clock Paths > bftClk > HOLD...

Timing Summary Results 창 우측의 테이블 보기에서, Device 창에서 선택하고 하이라이트할 경로를 클릭한다. Timing Summary Results 창에서 여러 경로를 선택하고, 라우팅 경로를 검토한다.

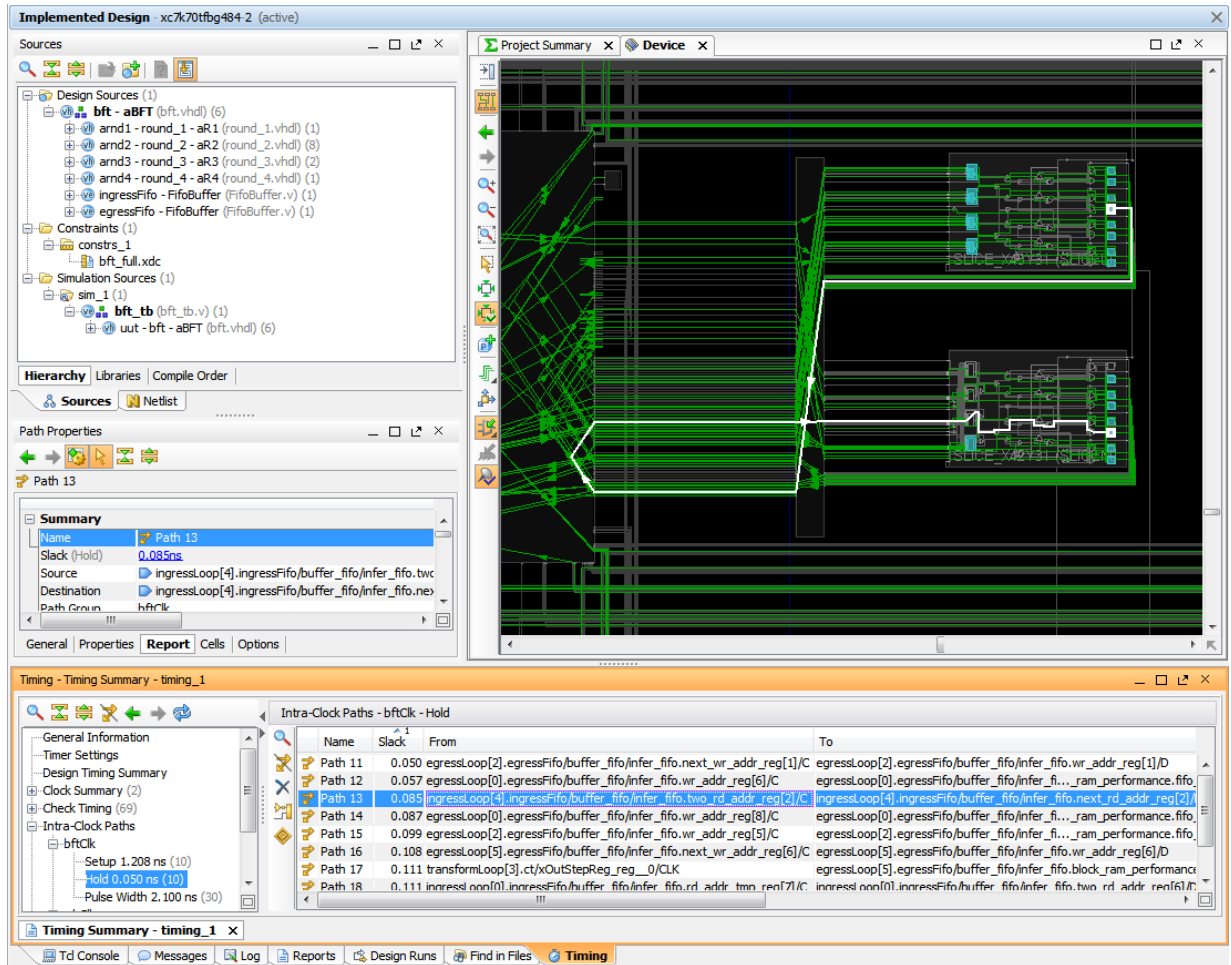


그림 27: 타이밍 경로에 대한 라우팅 검토

7. 선택된 타이밍 경로에서 Device 뷰나 Timing Summary 창에서 팝업 메뉴를 열기 위해 우측 클릭하고, **Schematic** 명령어를 선택한다.

참조: Schematic 창을 열기 위해 **F4** 기능 키를 눌러도 된다.

Schematic 창을 열고, 선택된 타이밍 경로를 위한 스케매틱을 디스플레이한다. 스케매틱을 둘러보기 위해 Schematic 창의 팝업 메뉴에서 Expand나 Collapse Outside, 혹은 Expand Cone과 같은 명령어를 이용하고, 타이밍 경로 상의 로직 셀을 검토한다.

8. Schematic 창을 종료한다.
9. Device 창에서, 라우팅 리소스를 턴 오프하기 위해 **Routing Resources** 툴바 버튼을 다시 선택한다. Device 창은 라우팅 커넥션에 대한 추가 상세정보없이 배치된 인스턴스만을 디스플레이하게 된다.

Step 10: 비트스트림 파일 생성

할당된 LOC로 배치되어 있는 디자인의 모든 I/O 포트 및 로직을 정의하는 IOSTANDARDS Constraints으로 비트스트림을 생성할 수 있다. Write Bitstream을 시작하기 전에 이 명령어의 설정을 검토할 수 있다.

1. Flow Navigator 에서, Program and Debug 아래의 **Bitstream Settings** 을 선택한다.

Project Settings 대화상자를 오픈한다. Bitstream Settings은 write_bitstream 명령어로 이용할 수 있는 옵션에 액세스할 수 있도록 해준다. 이러한 옵션에 대한 총체적인 설명 및 이용방법은 *비바도 디자인 수트 사용자 가이드: 프로그래밍 및 디버깅(UG908)*에서 확인할 수 있다.

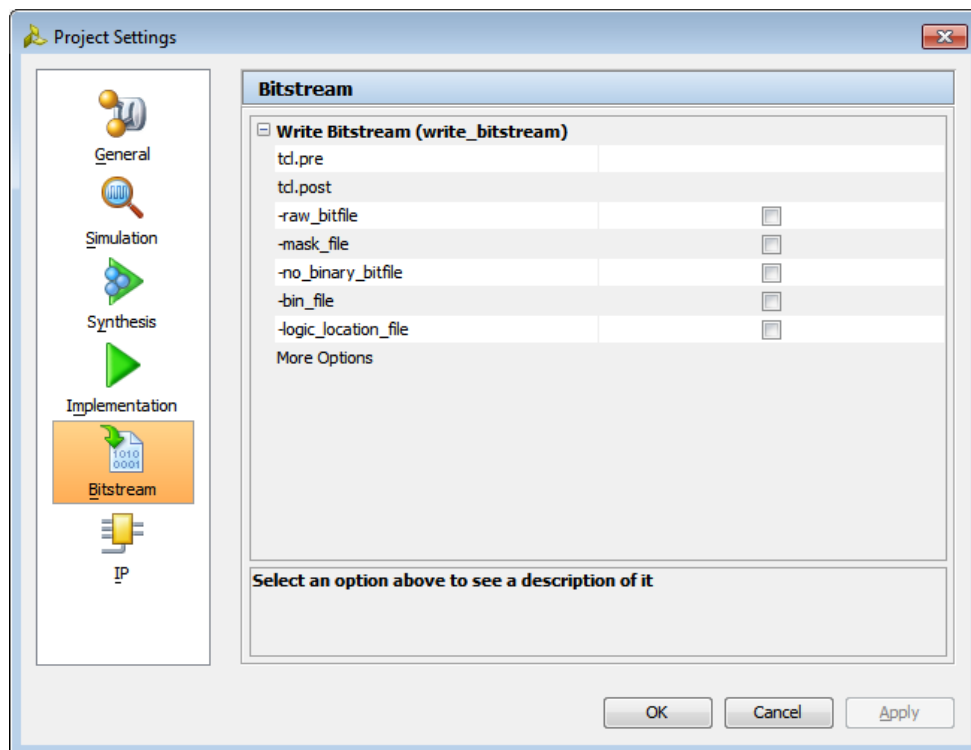


그림 28: 비트스트림 설정

2. Project Settings 대화상자를 종료하기 위해 **Cancel** 을 클릭한다.
3. Flow Navigator 에서, Program 및 Debug 섹션 아래의 **Generate Bitstream** 버튼을 클릭한다.
4. 비트스트림이 생성된 후에, 명령어로 리포트를 보기 위해 Bitstream Generation Completed 대화상자에서 **OK** 를 클릭한다.

Step 11: Journal 파일에서 Tcl 스크립트 생성

일괄 모드(Batch Mode) 실행은 더 빠르고, 비바도 IDE에서 실행하는 것보다 메모리 소모가 적다. 디자인을 완성하기 위해 여러 번 실행이 필요하다면, 플로우를 자동화하기 위해 Tcl 스크립트를 사용하는 것이 좋다. 또한 중요 단계 이후에 리포트 생성 명령어를 스크립트에 추가할 수 있으며, 특정 파일이나 디렉토리에 출력을 내보낼 수 있다.

비바도는 실행에 따라 두 개의 파일을 생성한다:

- 비바도 툴 로그(vivado.log) 파일은 비바도 세션동안 실행된 모든 Tcl 명령어 히스토리와 결과를 포함하고 있다.
- 비바도 툴 저널(vivado.jou) 파일은 로그 파일에 기록된 추가적인 상세정보없이 비바도 세션동안 실행된 Tcl 명령어만을 포함하고 있다.

이러한 파일은 각기 다른 디자인 작업을 수행하기 위해 비바도 툴이 사용하는 Tcl 명령어를 배울 수 있는 매우 좋은 방법이다. 또한 비바도 저널은 새로운 Tcl 스크립트를 생성할 때 매우 유용한 소스가 된다. 완료된 디자인 플로우의 vivado.jou 파일을 이용하여 디자인을 완성하는데 필요한 모든 Tcl 명령어를 확인할 수 있다. Tcl 명령어 및 옵션에 대한 총체적인 설명은 *비바도 디자인 수트 Tcl 명령어 레퍼런스 가이드(UG835)*에서 확인할 수 있다.

Log 및 Journal 검토

이 단계에서는 이번 lab의 Step1에서 Step10까지 작업하는 동안 비바도 툴이 자동으로 생성한 저널 파일에서 Tcl 스크립트를 수동으로 생성하게 된다. 비바도 IDE를 이용해 이러한 단계를 거쳐오면서 수행했던 것처럼, 새로운 스크립트를 실행할 때, 프로젝트 파일(.xpr) 및 디렉토리 구조를 생성한다. 이 프로젝트를 비바도 IDE에 로드하면, 기대하는 모든 결과 및 디스플레이된 프로젝트 상태를 확인할 수 있다.

1. **File > Exit** 를 선택하거나, Tcl 명령어 라인에서 exit 타이핑한다.
2. 비바도 툴을 종료하기 위해 **OK** 를 클릭한다.
3. 비바도 로그(vivado.log) 파일을 검토한다. 윈도우 상에서는 파일 브라우저를 이용하는 것이 더 용이할 수 있다.

`<Extract_Dir>/Vivado_Tutorial/vivado.log`

참조: 이는 Lab #2의 Step 1, [비바도 시작](#)에서 **Start-in** 속성을 입력했던 위치이다.

4. 콘텐츠를 검토하고 파일을 닫는다.
5. Text Editor 에서 **vivado.jou** 파일을 오픈한다.
6. 비바도 저널 파일을 검토한다. 윈도우 상에서는 파일 브라우저를 이용하는 것이 더 용이할 수 있다.

<Extract_Dir>/Vivado_Tutorial/vivado.jou

그림 29을 보면, 각기 다른 파일 경로에서 유사한 것을 확인할 수 있다:

```

#-----
# Process ID: 10736
# Log file: C:/Data/Vivado_Tutorial/vivado.log
# Journal file: C:/Data/Vivado_Tutorial/vivado.jou
#-----
source "C:/Users/xxx/AppData/Roaming/Xilinx/Vivado/init.tcl"
start_gui
create_project project_bft C:/Data/Vivado_Tutorial/Tutorial_Created_Data/project_bft -part xc7k70tfgb484-2
set_property simulator_language Verilog [current_project]
add_files {C:/Data/Vivado_Tutorial/Sources/hdl/bft.vhdl C:/Data/Vivado_Tutorial/Sources/hdl/async_fifo.v
C:/Data/Vivado_Tutorial/Sources/hdl/FifoBuffer.v}
add_files -fileset sim_1 C:/Data/Vivado_Tutorial/Sources/hdl/bft_tb.v
add_files C:/Data/Vivado_Tutorial/Sources/hdl/bftLib
set_property library bftLib {get_files {C:/Data/Vivado_Tutorial/Sources/hdl/bftLib/round_4.vhdl
C:/Data/Vivado_Tutorial/Sources/hdl/bftLib/round_3.vhdl C:/Data/Vivado_Tutorial/Sources/hdl/bftLib/round_2.vhdl
C:/Data/Vivado_Tutorial/Sources/hdl/bftLib/round_1.vhdl C:/Data/Vivado_Tutorial/Sources/hdl/bftLib/core_transform.vhdl
C:/Data/Vivado_Tutorial/Sources/hdl/bftLib/bft_package.vhdl}}
import_files -force
import_files -fileset constrs_1 -force -norecurse C:/Data/Vivado_Tutorial/Sources/bft_full.xdc
update_compile_order -fileset sources_1
update_compile_order -fileset sources_1
update_compile_order -fileset sim_1
synth_design -rtl -name rtl_1
close_design
launch_xsim -simset sim_1 -mode behavioral
close_sim
launch_runs synth_1
wait_on_run synth_1
launch_runs impl_1
wait_on_run impl_1
open_run synth_1 -name netlist_1
report_timing_summary -delay_type min_max -report_unconstrained -check_timing_verbos -max_paths 10 -input_pins -name timing_1
report_power -results {power_1}
close_design
open_run impl_1
report_timing_summary -delay_type min_max -report_unconstrained -check_timing_verbos -max_paths 10 -input_pins -name timing_1
launch_runs impl_1 -to_step write_bitstream
wait_on_run impl_1

```

그림 29: Lab #2 용 비바도 저널 파일

7. 필요하지 않다면, 코멘트 헤더(#으로 시작된 라인)를 삭제한다.
8. 툴이 시작될 때 실행하는 **init.tcl script** 를 소싱하는 모든 라인을 삭제한다.
9. 일괄(Batch) 스크립트에서는 비바도 IDE 를 오픈할 필요가 없기 때문에 **start_gui** 라인을 삭제한다.
10. 파일을 <Extract_Dir>/Vivado_Tutorial/run_bft.tcl 에 저장하기 위해 **Save As** 명령어를 이용한다.
11. **run_bft.tcl** 스크립트를 오픈하고, 모든 "project_bft" Occurrences 를 검색하여 "project_bft_batch"로 대체한다.
12. 스크립트를 검토한다. 이 프로젝트 모드 스크립트는 이 지침서의 Lab #1 에서 사용된 비-프로젝트 모드 스크립트와는 다르다.

프로젝트 생성에 사용된 `add_files` 및 `set_property` 명령어는 물론, Constraints 설정을 위한 명령어에 주목해야 한다. 또한 `launch_runs`는 `synth_design` 등 대신에 사용되었다. 프로젝트 기반 디자인을 생성하거나 실행할 때, `launch_runs` 명령어를 이용한다.



주의! 개별 명령어(`synth_design`, `opt_design`...)를 `launch_runs`와 혼합하면 프로젝트에 손상을 줄 수 있어 추천하지 않는다. `launch_runs` 명령어는 독립적으로 단계를 실행하고, 리포트를 중간 중간 생성할 수 있는 Tcl 옵션을 가지고 있다. 보다 자세한 정보는 비바도 디자인 스위트 Tcl 명령어 레퍼런스 가이드(UG835)에서 확인할 수 있다.

일괄(Batch) 프로젝트 스크립트 편집

하나의 디자인 세션에서 지침서의 Lab #2를 완료하지 못했다면, **vivado.jou** 파일은 Step 1에서 Step12까지의 완벽한 디자인 플로우를 반영하지 않는다. 이 경우, Vivado_Tutorial 디렉토리에서 찾을 수 있는 **run_bft_project.tcl** 스크립트를 이용할 수 있다.

run_bft.tcl 스크립트의 첫 번째 라인은 디자인 프로젝트를 생성하고, 타겟 자일링스 부품을 명기하고, 소스 RTL 및 XDC 파일을 추가하며, VHDL 라이브러리를 정의한다. 이 스크립트는 합성 및 구현 실행, **synth_1** 및 **impl_1** 을 생성하며, Constraints 설정, **constrs_1** 을 정의한다.

다음 몇 라인들은 RTL 디자인을 정교하게 만들고, 시뮬레이션한다. 일괄 플로우에서는 이를 수행할 필요가 없다.

1. 필요시 스크립트를 여러 번 실행할 수 있도록 기존 프로젝트에 덮어쓰기 위해 **-force** 옵션을 **create_project** 명령어에 추가한다.
2. Tcl 스크립트에서 라인 코멘트를 만들기 위해 '#' 심볼을 삽입하거나 라인을 삭제하여 스크립트에서 아래의 라인들을 제거한다.

```
#synth_design -rtl -name rtl_1
#close_design
#launch_xsim -simset sim_1 -mode behavioral
#close_sim
```

이 지침서의 이전 단계에서, 합성 및 구현을 실행했고, 합성된 디자인을 분석하면서 구현을 실행했다. 일괄(Batch) 플로우에서는, 합성을 실행한 다음, 타이밍 및 전력 리포트를 실행하고, 구현을 실행하게 될 것이다. 이를 수행하기 위해서는 스크립트에서 Tcl 명령어를 재배열해야 한다. **open_run** 명령어와 다음의 3개 라인을 잘라서 **wait_on_run synth_1** 명령어 다음으로 이동시킨다.

3. 다음 라인들을 잘라낸다:

```
open_run synth_1...
report_timing_summary...
report_power...
```

4. **wait_on_run synth_1...** 라인 뒤에 이를 붙여넣기 한다.

이제 스크립트는 합성을 시작하고, 합성이 완료되기를 기다린다. 그런 다음 구현을 시작하고, 구현이 완료되기를 기다린다. 마지막으로 스크립트는 **write_bitstream**을 시작하고, 완료되기를 기다린다.

5. 파일에 **close_design** 라인이 있다면, 이를 제거한다.
6. 파일을 저장한다.

이 파일은 [그림 30](#)처럼 보일 것이다.

```

create_project project_bft_batch C:/Data/Vivado_Tutorial/Tutorial_Created_Data/project_bft_batch -part xc7k70tfbg484-2 -force
set_property simulator_language Verilog [current_project]
add_files {C:/Data/Vivado_Tutorial/Sources/hdl/bft.vhdl C:/Data/Vivado_Tutorial/Sources/hdl/async_fifo.v
C:/Data/Vivado_Tutorial/Sources/hdl/FifoBuffer.v}
add_files -fileset sim_1 C:/Data/Vivado_Tutorial/Sources/hdl/bft_tb.v
add_files C:/Data/Vivado_Tutorial/Sources/hdl/bftLib
set_property library bftLib [get_files {C:/Data/Vivado_Tutorial/Sources/hdl/bftLib/round_4.vhdl
C:/Data/Vivado_Tutorial/Sources/hdl/bftLib/round_3.vhdl C:/Data/Vivado_Tutorial/Sources/hdl/bftLib/round_2.vhdl
C:/Data/Vivado_Tutorial/Sources/hdl/bftLib/round_1.vhdl C:/Data/Vivado_Tutorial/Sources/hdl/bftLib/core_transform.vhdl
C:/Data/Vivado_Tutorial/Sources/hdl/bftLib/bft_package.vhdl}]
import_files -force
import_files -fileset constrs_1 -force -norecurse C:/Data/Vivado_Tutorial/Sources/bft_full.xdc
update_compile_order -fileset sources_1
update_compile_order -fileset sources_1
update_compile_order -fileset sim_1
launch_runs synth_1
wait_on_run synth_1
open_run synth_1 -name netlist_1
report_timing_summary -delay_type min_max -report_unconstrained -check_timing_verbose -max_paths 10 -input_pins -name timing_1
report_power -results {power_1}
launch_runs impl_1
wait_on_run impl_1
open_run impl_1
report_timing_summary -delay_type min_max -report_unconstrained -check_timing_verbose -max_paths 10 -input_pins -name timing_1
launch_runs impl_1 -to_step write_bitstream
wait_on_run impl_1

```

그림 30: 편집된 Tcl 스크립트

일괄(Batch) 프로젝트 스크립트 실행

이제 새로운 Tcl 스크립트를 실행할 수 있으며, 비바도 툴을 Tcl 스크립트에 있는 모든 명령어를 실행하게 하는 일괄 모드(Batch Mode)로 실행한 다음, 완료되면 비바도를 종료한다.

1. 윈도우 상에서, **Command Prompt** 창을 엽니다. 리눅스의 경우 앞 3 단계를 간단히 스킵한다.
2. 자일링스 인스톨 공간으로 **디렉토리를 변경**하고, 여러분의 컴퓨터에 자일링스 툴 경로를 설정하고자 한다면, **settings32.bat** 이나 **settings64.bat** 을 실행한다.

```

cd <Vivado_install_area>/Vivado/2014.x
settings64

```

settings64.bat 파일은 여러분의 컴퓨터에 비바도 툴을 실행하는 경로 및 환경을 설정한다.

3. **<Extract_Dir>/Vivado_Tutorial** 로 디렉토리를 변경하고, 일괄 모드(Batch Mode)로 비바도 툴을 시작한다:

```

cd <Extract_Dir>/Vivado_Tutorial
vivado -mode batch -source run_bft.tcl

```

4. Command Prompt 창으로 트랜스크립트되기 때문에, 비바도 로그 출력을 검토한다.

launch_runs 명령어가 사용된 후에는, 보다 적은 정보가 툴 트랜스크립트로 되풀이된다. 또한 리포트 및 실행 상태는 프로젝트에 모이고, 실행이 완료된 후 이용 가능하다.

비바도 툴을 일괄 모드(Batch Mode)로 실행했기 때문에, 소스 스크립트가 실행이 완료된 후 종료된다.

Step 12: 디자인 상태 확인

1. 비바도 IDE 를 시작하고 여러분이 생성한 BFT 일괄 프로젝트(project_bft_batch.xpr)를 오픈한다:

Start > All Programs > Xilinx Design Tools > Vivado 2014.x > Vivado 2014.x³

이 대신 명령어 라인에서 비바도 IDE를 시작할 수도 있다:

```
> cd <Extract_Dir>/Vivado_Tutorial/Tutorial_Created_Data  
> vivado -mode gui
```

비바도 IDE가 시작된다.

2. File > Open Project 로 프로젝트를 오픈하고, project_bft_batch 의 위치를 확인한다.

비바도 IDE의 상단 우측에 있는 프로젝트 상태 바를 확인할 수 있는데, 이 상태는 비트스트림이 생성되었다는 사실을 반영한다.(write_bitstream Complete)

3. Flow Navigator 에서 **Open Implemented Design** 버튼을 클릭하여 구현된 디자인을 확인한다.

4. 완료되면, 비바도 툴을 종료한다. 이 사용 지침서도 마무리되었다.

File > Exit

요약

이 지침서에서 여러분은 다음과 같은 내용을 배우게 되었을 것이다:

- 프로젝트 모드 및 비-프로젝트 모드 사용.
- 비바도 IDE 에서 RTL 프로젝트 생성
- 비바도 합성, 시뮬레이션, 구현 툴 컨피규레이션
- 비바도 시뮬레이터, 합성, 구현 시작.
- 합성된 디자인에 Constraints 적용.
- 타이밍 및 전력 리포트 생성.
- Device 에디터에서 라우팅 결과 검토.
- 비트스트림 파일 생성.
- 프로젝트 기반 Tcl 스크립트를 생성하기 위해 저널 파일 (.jou) 사용.
- 명령어 라인에서 프로젝트 기반 Tcl 스크립트 시작.
- 비바도 디자인 수트 Tcl shell 및 비바도 IDE 간의 전환.

³ 여러분의 비바도 디자인 수트 설치 시 **시작** 메뉴에서 **자일링스 디자인 툴**과 다르게 불러질 수 있다.