

# How to make IP with Vivado HLS

Innova Lee(이상훈)  
[gcccompil3r@gmail.com](mailto:gcccompil3r@gmail.com)

## Introduction

이 Lab 에서 Vivado HLS 를 사용하는 설계에서 IP-XACT 어댑터를 생성하고  
Vivado 에서 IP Integrator 를 사용하는 Processor System 에서 생성 된 IP-XACT 어댑터를 사용하는 설계 흐름을 소개한다.

## Objectives

이 Lab 을 완료하면 아래르 수행 할 수 있다:

- \* Vivado HLS 의 Synthesized Design 에서 IP-XACT 어댑터를 만드는 단계와 지침을 이해한다.
- \* Vivado 에서 IP Integrator 를 사용하여 Processor System 생성이 가능함
- \* 생성 된 Processor System 에 생성한 IP-XACT 어댑터를 통합한다.

## The Design

이 설계는 CD 품질(48 KHz) 음악에 추가 된 4 KHz 톤을 필터링 하기 위한 FIR 필터로 구성된다.  
필터의 특성은 아래와 같다.

FS = 48000 Hz  
FPASS1=2000 Hz  
FSTOP1=3800 Hz  
FSTOP2=4200 Hz  
FPASS2=6000 Hz  
APASS1=APASS2=1 dB  
ASTOP=60 dB

이 Lab 에서 Processor System 에서 Instance 화 할 수 있는 설계된 필터의 Peripheral Core 를 개발해야 한다.  
Processor System 은 On-Board CODEC 칩 및 I2C Controller 를 사용하여 Stereo 음악 스트림을 수집하고  
설계된 필터(Bandstop Filter)를 통해 처리하며 Headphone 으로 다시 출력한다.

## Procedure

이 Lab 은 세부적인 지침에 대한 정보를 제공하는 일반적인 개요로 구성되어 있다.  
실험을 진행하려면 아래 세부 지침을 따르라.

이 실습은 9 가지 기본 단계로 구성된다:

Vivado HLS 에서 새 프로젝트를 만들고,

시뮬레이션을 실행하고,

디자인을 합성하고,

RTL/C 공동 시뮬레이션을 실행하고,

Project Navigator 에서 프로젝트를 만들고,

ISIM 을 사용하여 시뮬레이션을 실행하고,

Vivado HLS 에서 IP-XACT 어댑터를 설정하고,

디자인을 구현한다.

Vivado HLS 에서 IP Integrator 를 사용하여 Vivado 에 Processor System 을 만들고  
SDK 에 SW 응용 프로그램을 만들고 HW 에서 Design 을 확인한다.

## General Flow for this Lab

1. New Project 생성
2. C Simulation 실행
3. 설계 합성
4. RTL/C 동시 시뮬레이션 실행
5. IP-XACT 어댑터 설정
6. IP-XACT 어댑터 생성
7. Vivado Project 생성
8. SDK 로 내보내기 및 응용 프로그램 만들기
9. HW 에서 Design 확인
10. Tcl Script 를 사용하여 초기 Design 만들기

## Create a New Project

1-1. Vivado HLS 에서 XC7Z020CLG484-1(ZedBoard) 또는 XC7Z010CLG400-1(Zybo)을 Target 으로 New Project 를 만든다.

1-1-1. Vivado HLS 띄우기: Start > All Programs > Xilinx Design Tools > Vivado 2017.4 > Vivado HLS > Vivado HLS 2017.4  
Getting Started GUI 가 나타남(리눅스라면 설치한 디렉토리에 HLS 쪽에 있음)

1-1-2. Getting Started 섹션에서 Create New Project 를 클릭한다.  
New Vivado HLS Project 마법사가 열린다.

1-1-3. Browse ... 버튼을 클릭하고 c:\Wxup\Whls\Wlabs\Wlab4 로 이동한 다음 OK 를 누른다.

1-1-4. 프로젝트 이름에 fir.prj 를 입력한다.

1-1-5. Next 를 클릭한다.

1-1-6. 원본 파일의 Add/Remove Files 에서 fir 을 함수 이름으로 입력한다.  
제공된 원본 파일에는 fir 이라는 합성 할 함수가 들어 있다.

1-1-7. Add Files ... 버튼을 클릭하고 c:\Wxup\Whls\Wlabs\Wlab4 폴더에서 fir.c 및 fir\_coef.dat 파일을 선택한 다음 Open 을 클릭한다.

1-1-8. Next 를 클릭한다.

1-1-9. testbench 파일의 Add/Remove Files 에서 Add Files ... 버튼을 클릭하고 c:\Wxup\Whls\Wlabs\Wlab4 폴더에서 fir\_test.c 파일을 선택한 다음 Open 을 클릭한다.

1-1-10. Next 를 클릭한다.

1-1-11. Solution Configuration 페이지에서 Solution Name 필드를 solution1 로 두고 Clock Period 를 10(ZedBoard 의 경우) 또는 8(Zybo 의 경우)로 설정한다.  
Uncertainty 필드를 비워두면 ZedBoard 의 기본값으로 1.25, Zybo 의 기본값으로 1.25 가 된다.

1-1-12. Part's Browse 버튼을 클릭하고 Parts Specify 옵션을 사용하여  
xc7z020clg484-1(ZedBoard) 또는 xc7z010clg400-1(Zybo)을 선택하고 OK 를 클릭한 후 아래와 같이 선택한다.  
Family: Zynq  
Sub-Family: Zynq  
Package: clg484(ZedBoard) or clg400(Zybo)  
Speed Grade: -1

1-1-13. Finish 를 클릭한다.

Explorer View 에서 생성 된 Project 를 볼 수 있다.

다양한 하위 폴더를 확장하여 각 하위 폴더 아래의 항목을 확인한다.

1-1-14. 소스 폴더 아래의 fir.c 를 두 번 클릭하여 Information Pane 에서 내용을 연다.

```
1 #include "fir.h"
2
3 void fir (
4     data_t *y,
5     data_t x
6 ) {
7     const coef_t c[N+1]={
8 #include "fir_coef.dat"
9     };
10
11
12     static data_t shift_reg[N];
13     acc_t acc;
14     int i;
15
16     acc=(acc_t)shift_reg[N-1]*(acc_t)c[N];
17     loop: for (i=N-1;i!=0;i--) {
18         acc+=(acc_t)shift_reg[i-1]*(acc_t)c[i];
19         shift_reg[i]=shift_reg[i-1];
20     }
21     acc+=(acc_t)x*(acc_t)c[0];
22     shift_reg[0]=x;
23     *y = acc >> 15;
24 }
```

FIR 필터는 x 를 샘플 입력으로 계산된 샘플을 가리키는 포인터를 출력한다.

둘 다 데이터 유형 data\_t 로 정의된다.

계수는 현재 디렉토리에 있는 fir\_coef.dat 라는 파일에서 coef\_t 유형의 배열 c 에 로드 된다.

순차 알고리즘이 적용되고 축적 된 값(샘플 아웃)이 acc\_t 유형의 변수 acc 에서 계산된다.

1-1-15. Outline Tab 에서 fir.h 를 더블 클릭하여 Information Pane 에 내용을 연다.

```
1 #ifndef _FIR_H_
2 #define _FIR_H_
3 #include "ap_cint.h"
4 #define N 58
5 #define SAMPLES N+10 // just few more samples then number of taps
6 typedef short coef_t;
7 typedef short data_t;
8 typedef int38 acc_t;
9 #endif
10
```

헤더 파일에는 ap\_cint.h 가 포함되어 있어 사용자 정의 데이터 너비(임의 정밀도)를 사용할 수 있다.  
또한 탭 수(N), 생성 될 샘플 수(testbench 에서) 및 데이터 유형 coef\_t, data\_t 및 acc\_t 를 정의한다.  
coef\_t 와 data\_t 는 짧다(16 비트)  
알고리즘은 59 개 이상의 탭을 반복(누적 및 곱셈)하므로 6 비트의 비트 증가 가능성이 있어 acc\_t 는 int38 로 정의된다.  
acc\_t 는 샘플 및 계수 너비보다 크기 때문에 사용하기 전에 casting 해야 한다(fir.c 의 16, 18 및 21 행과 같다)

1-1-16. testbench 폴더 아래에 있는 fir\_test.c 를 두 번 클릭하여 Information Pane 에 내용을 연다.

testbench write 모드에서 fir\_impulse.dat 를 열고 Impulse 를 보낸다.  
(첫 번째 샘플은 0x8000 이다)

## Run C Simulation

2-1. C Simulation 을 실행하여 예상 출력을 관찰한다.

2-1-1. Project > Run C Simulation 을 선택하거나 Tool bar 버튼을 클릭하고 C Simulation Dialog 창에서 OK 를 클릭한다.  
testbench 는 apcc 컴파일러를 사용하여 컴파일되고 csim.exe 파일이 생성된다.  
그러면 csim.exe 가 실행되고 출력이 Console View 에 표시된다.

```
Starting C simulation ...
C:/Xilinx/Vivado/2017.4/bin/vivado_hls.bat C:/xup/hls/labs/lab4/fir.prj/solution1/csim.tcl
INFO: [HLS 200-10] Running 'C:/Xilinx/Vivado/2017.4/bin/unwrapped/win64.o/vivado_hls.exe'
INFO: [HLS 200-10] For user 'parimalp' on host 'xsjparimalp31' (Windows NT_amd64 version 6
INFO: [HLS 200-10] In directory 'C:/xup/hls/labs/lab4'
INFO: [HLS 200-10] Opening project 'C:/xup/hls/labs/lab4/fir.prj'.
INFO: [HLS 200-10] Opening solution 'C:/xup/hls/labs/lab4/fir.prj/solution1'.
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [HLS 200-10] Setting target device to 'xc7z020clg484-1'
INFO: [SIM 211-2] ***** CSIM start *****
INFO: [SIM 211-4] CSIM will launch GCC as the compiler.
    Compiling(apcc) ../../../../fir_test.c in debug mode
INFO: [HLS 200-10] Running 'c:/Xilinx/Vivado/2017.4/bin/unwrapped/win64.o/apcc.exe'

INFO: [HLS 200-10] In directory 'C:/xup/hls/labs/lab4/fir.prj/solution1/csim/build'
INFO: [APCC 202-3] Tmp directory is apcc_db
INFO: [APCC 202-1] APCC is done.
    Generating csim.exe
0 -32768 378
1 0 73
2 0 -27
3 0 -170
4 0 -298
5 0 -352
6 0 -302
7 0 -168
8 0 -14
9 0 80
10 0 64
11 0 -53
12 0 -186
13 0 -216
```

계산중인 Filter 계수를 확인해야 한다.

## Synthesize the Design

3-1. 기본값을 사용하여 Design 을 Sythesize 한다.

Synthesis 결과를 보고 이 단계의 세부 절차에 나열된 질문에 답하라.

3-1-1. Solution > C Run Synthesis > Active Solution 을 선택하여 Synthesis 절차를 시작하라.

3-1-2. Synthesis 가 완료되면 여러 개의 Report 파일에 접근 할 수 있고 Synthesis 결과가 Information Pane 에 표시된다.

3-1-3. Synthesis Report 는 Design 의 Estimated Latency 뿐만 아니라 Performance 및 Resource 추정도 표시한다.

3-1-4. 오른쪽에 Scroll 막대를 사용하여 아래로 Scroll 하여 Report 에 들어가서 아래 물음에 답하라.

### Question 1

Estimated clock period:

---

Worst case latency:

---

Number of DSP48E used:

---

Number of BRAMs used:

---

Number of FFs used:

---

Number of LUTs used:

---



3-1-5. 이 보고서에는 Tool 로 생성 된 최상위 인터페이스 신호도 표시된다.

Interface					
Summary					
RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	fir	return value
ap_rst	in	1	ap_ctrl_hs	fir	return value
ap_start	in	1	ap_ctrl_hs	fir	return value
ap_done	out	1	ap_ctrl_hs	fir	return value
ap_idle	out	1	ap_ctrl_hs	fir	return value
ap_ready	out	1	ap_ctrl_hs	fir	return value
y	out	16	ap_vld	y	pointer
y_ap_vld	out	1	ap_vld	y	pointer
x	in	16	ap_none	x	scalar

Design 에서 x 입력을 16 비트 Scalar 로 예상하고 16 비트 데이터의 포인터를 통해 y 를 출력하는 것을 볼 수 있다.  
또한 결과가 유효할 때를 나타내는 ap\_vld 신호가 있다.

3-2. PIPELINE 지시어를 추가하여 Loop 를 만들고 Design 을 다시 Synthesize 하고 합성 결과를 본다.

3-2-1. Information View 에서 fir.c 가 열려 있는지 확인한다.

3-2-2. Directive Tab 을 선택하고 PIPELINE 지시어를 Loop 에 적용한다.

3-2-3. Solution > C Run Synthesis > Active Solution 을 선택하여 Synthesis 과정을 시작한다.

3-2-4. Synthesis 가 완료되면 Synthesis 결과가 Information Pane 에 표시된다.

3-2-5. Latency 는 62 또는 63 Clock Cycle 로 감소되었다.  
DSP48 및 BRAM 소비는 동일하게 유지되지만 LUT 및 FF 소모량은 약간 증가했다.

## Run RTL/C CoSimulation

4-1. Verilog 를 선택하여 RTL/C Simulation 을 실행하라.  
Simulation 에 대한 Verify 를 통과한다.

4-1-1. Solution -> C/RTL Co-Sim 실행을 선택하거나 버튼을 클릭하여 Dialog Box 를 열면 원하는 Simulation 을 실행할 수 있다.

C/RTL Co-Simulation Dialog Box 가 열린다.

4-1-2. Verilog 옵션을 선택하고 OK 를 클릭한다.

Co-Simulation 이 실행되어 여러 파일을 생성 및 컴파일 한 다음 Design 을 시뮬레이션 한다.  
Console Window 에서 진행 상황을 볼 수 있다.  
완료되면 RTL Simulation 보고서는 성공했으며 보고된 Latency 는 62 엀음을 보여준다.

## Setup IP-XACT Adapter

5-1. INTERFACE 지시어를 추가하여 AXI4LiteS 어댑터를 생성하면  
RTL Export 단계에서 IP-XACT 어댑터를 생성 할 수 있다.

5-1-1. Information View 에서 fir.c 파일이 열려 있고 Focus 가 있는지 확인하라.

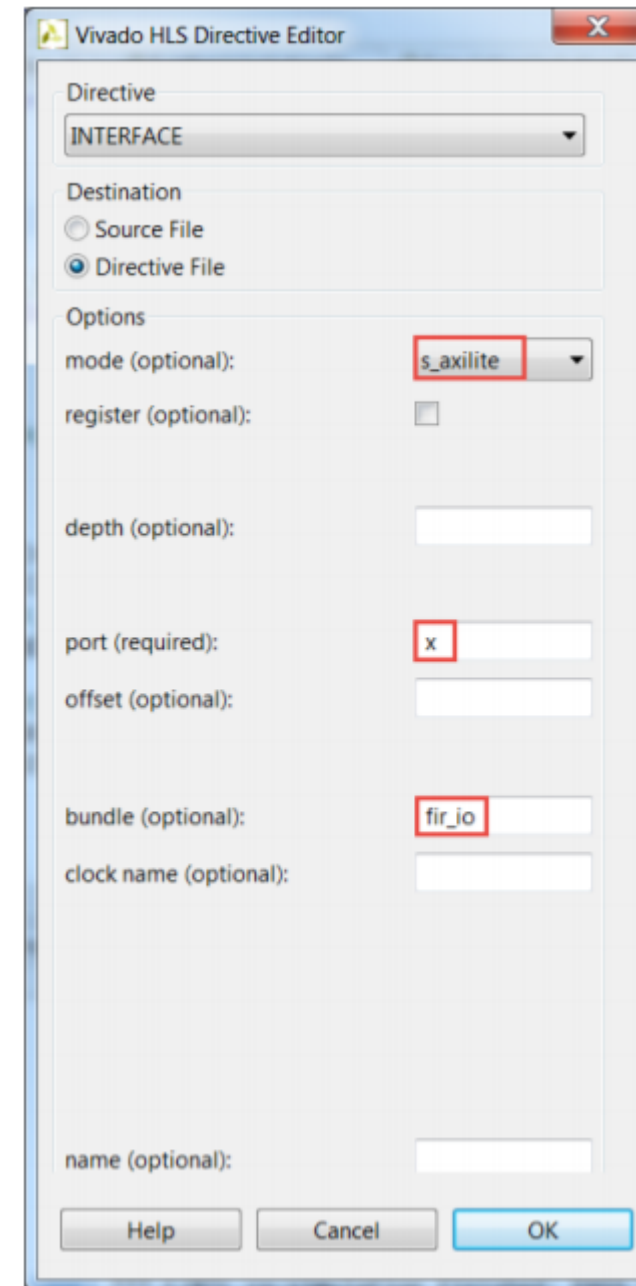
5-1-2. Directive Tab 을 선택하라.

5-1-3. x 를 마우스 우클릭하고 Insert Directive ... 를 클릭한다.

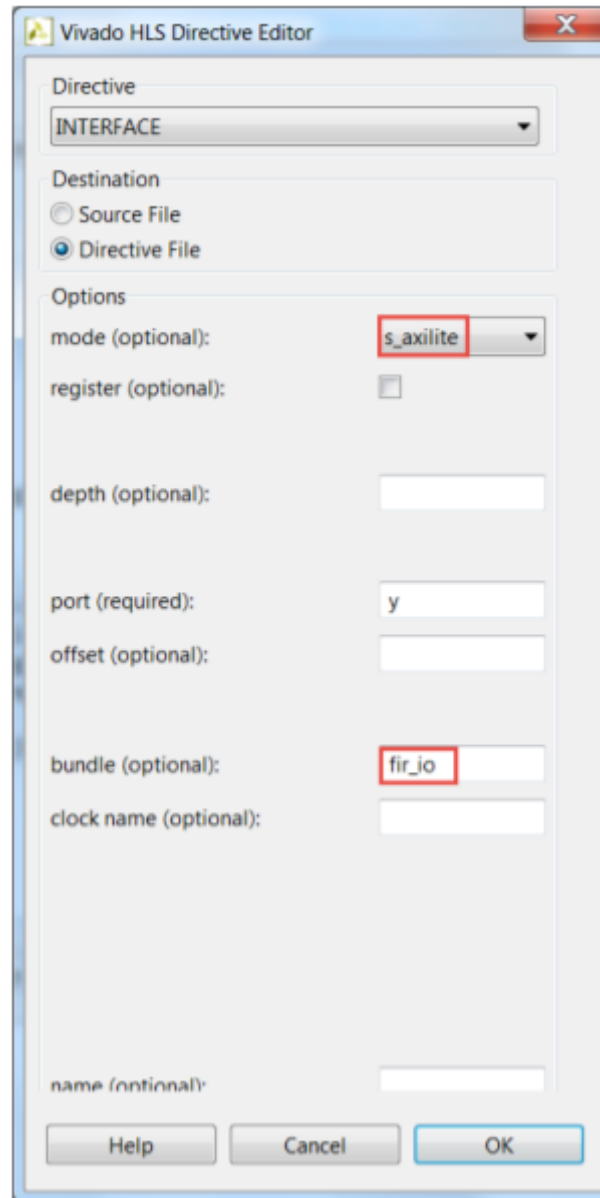
5-1-4. Vivado HLS Directive Editor Dialog Box 의 Drop-Down 버튼을 사용하여 INTERFACE 를 선택한다.

5-1-5. Mode 옆에 있는 버튼을 클릭한다(옵션).  
s\_axilite 를 선택하라.

5-1-6. Bundle(선택 사항) 필드에 fir\_io 를 입력하고 OK 를 클릭한다.

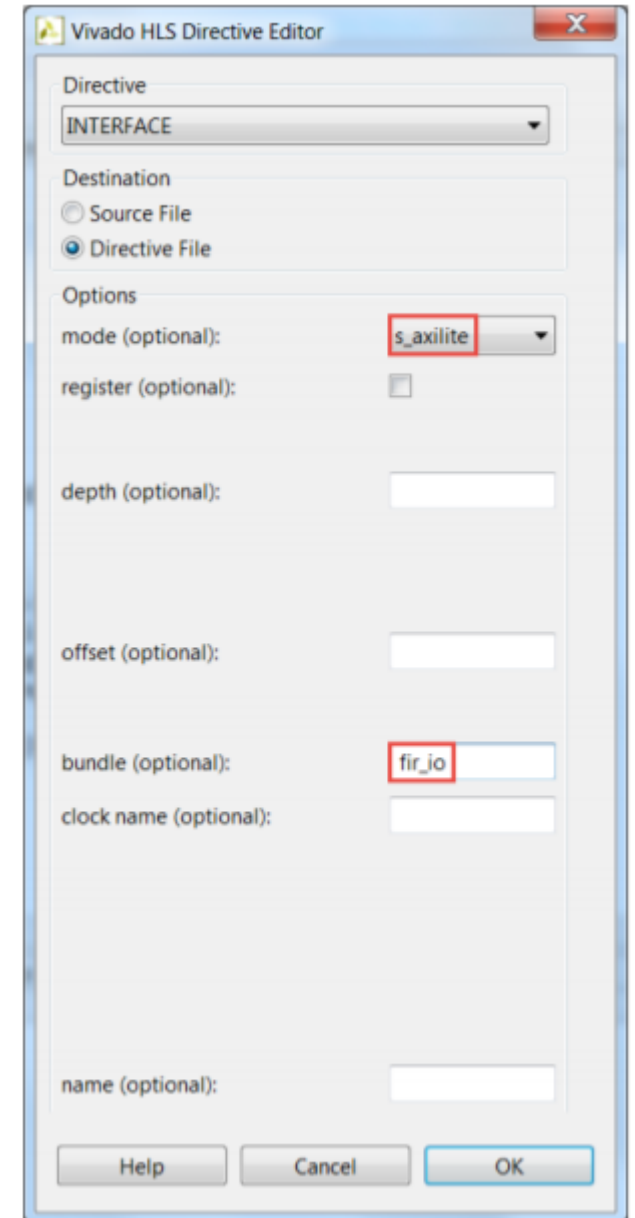


5-1-7. 마찬가지로 y 출력에 INTERFACE 지시어(Bundle 포함)을 적용한다.



5-1-8. Bus 어댑터의 일부로 ap\_start, ap\_done 및 ap\_idle 신호를 포함하도록 INTERFACE 지시어를 최상위 모듈 fir 에 적용하라.  
(표시된 변수 이름은 리턴 됨)  
Bundle 정보도 포함하라.

위의 단계 5-1-3 에서 5-1-8 까지가 SW 를 통해 접근 할 수 있는  
x, y, ap\_start, ap\_valid, ap\_done 및 ap\_idle 에 대한 주소 맵을 작성한다.  
또는 ap\_start, ap\_valid, ap\_done, ap\_idle 신호는 최상위 모듈 fir 에  
RESOURCE 지시어를 적용하지 않음으로써 Core 에서 별도의 Port 로 생성 될 수 있다.  
그러면 이러한 Port 는 사용 가능한 GPIO IP 를 사용하는 Processor System 에 연결되어야 한다.

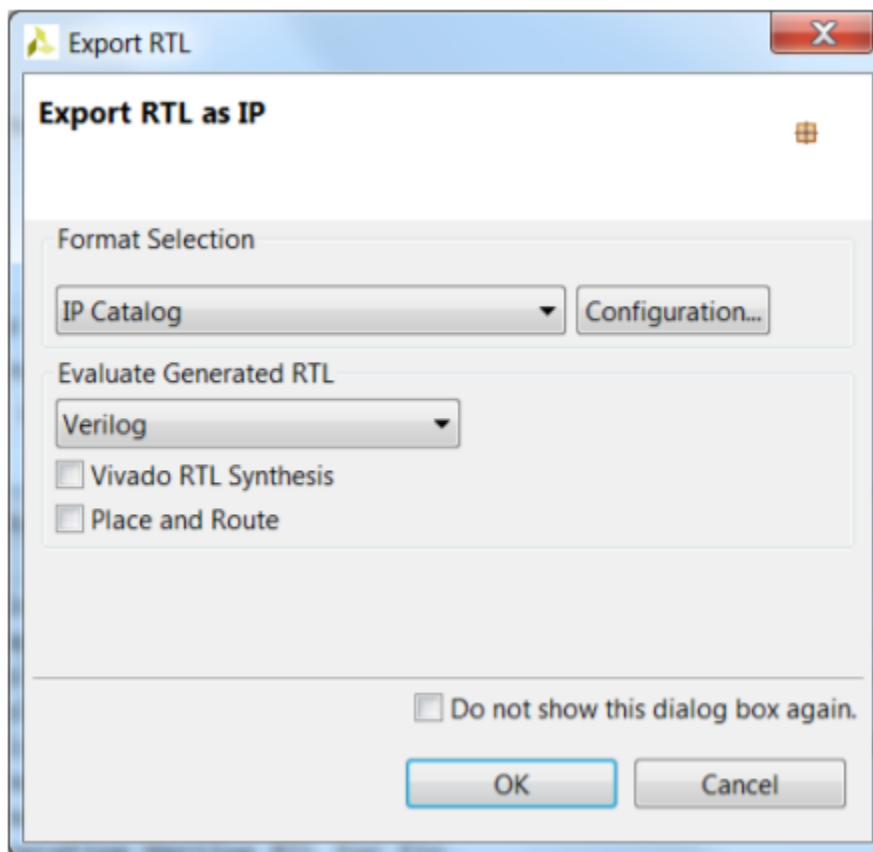


## Generate IP-XACT Adapter

6-1. 지시어가 추가되면서 Design 을 다시 합성한다.  
RTL Export 를 실행하여 IP-XACT 어댑터를 생성하라.

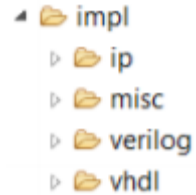
6-1-1. 지시어가 추가되었으므로 Design 을 다시 합성하는 것이 안전하다.  
Solution > Run C Synthesis > Active Solution 을 선택하라.  
Synthesis Report 하단의 인터페이스 요약을 확인하여 작성된 인터페이스를 확인하라.

6-1-2. Design 이 합성되면 Solution > Export RTL 을 선택하여 원하는 IP 가 생성 될 수 있도록 Dialog Box 를 연다.  
Export RTL Dialog Box 가 열린다.



6-1-3. OK 를 클릭하여 IP-XACT 어댑터를 생성하라.

6-1-4. 실행이 완료되면 Explorer View 에서 impl 폴더를 확장하고 생성 된 여러 디렉토리를 관찰한다.  
IP, 기타, Verilog 및 VHDL 등을 살펴보면 된다.



ip 디렉토리를 확장하고 여러 파일과 하위 디렉토리를 관찰한다.

관심있는 하위 디렉토리 중 하나는 header, c, tcl, mdd 및 makefile 파일로 구성된 drivers 디렉토리다.

관심있는 또 다른 파일은 zip 파일로 zip 파일은 IP Integrator Design 에서 가져올 수 있는 IP 저장소에 해당한다.

6-1-5. File > Exit 를 선택하여 Vivado HLS 를 닫는다.



## Create a Vivado Project

- 7-1. Vivado Tcl Shell 을 실행하고 제공된 tcl Script 를 실행하여 ZedBoard(xc7z020clg484-1 Device 가 있는 경우) 또는 Zybo(xc7z010clg400-1 Device 가 있는 경우)를 대상으로 초기 시스템을 만든다.  
시스템을 처음부터 새로 작성하려면 부록에 제공된 단계를 따르고 아래 7-2 단계부터 계속하라.

7-1-1. Start > All Programs > Xilinx Design Tools > Vivado 2017.4 > Vivado 2017.4 Tcl Shell 을 연다(리눅스 용 별도)

7-1-2. Shell 창에서 cd 를 사용하여 디렉토리를 c:\wxup\whlswlabs\wlab4 로 변경한다.

- 7-1-3. 제공된 Script 파일을 실행하여 다음 명령을 입력하고  
zed\_audio\_ctrl 및 GPIO Peripherals 가 있는 초기 시스템을 만든다.

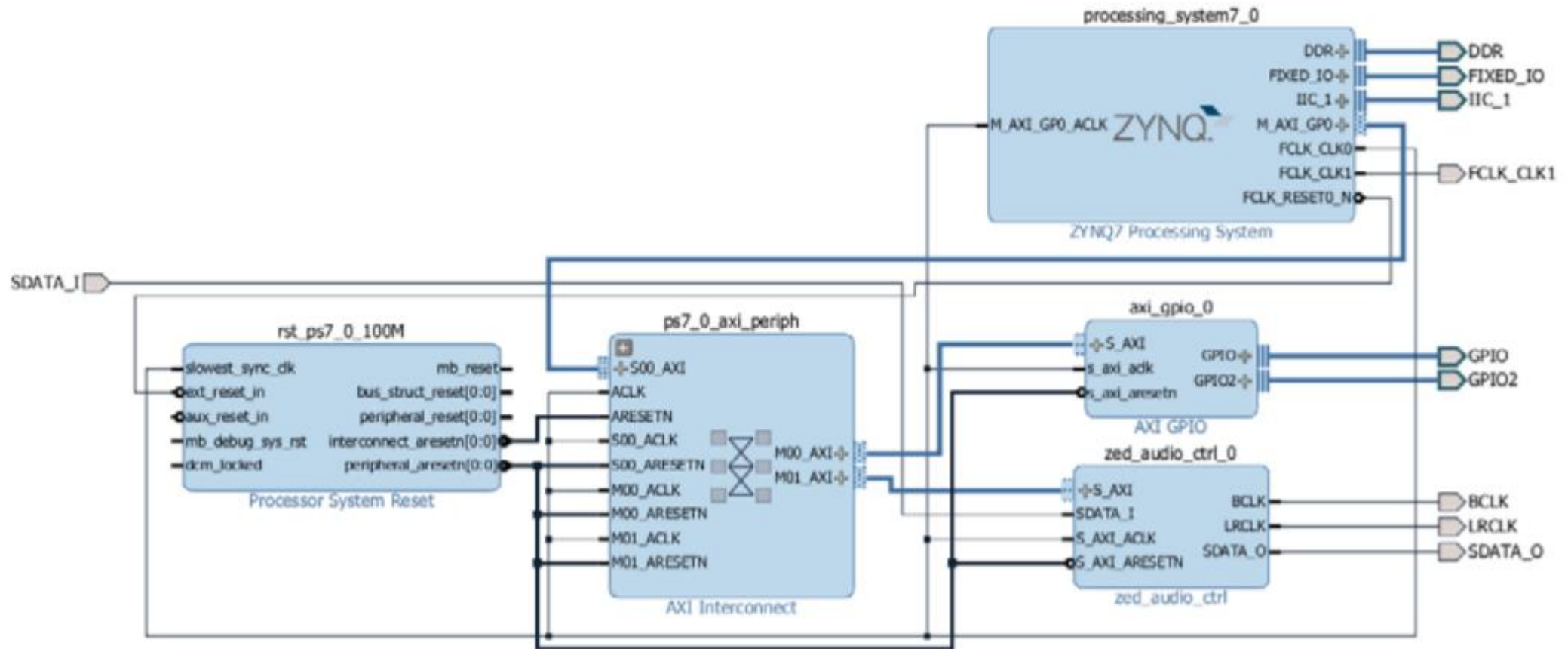
ZedBoard 용 소스 zed\_audio\_project\_create.tcl 또는

Zybo 용 소스 zybo\_audio\_project\_create.tcl 이다.

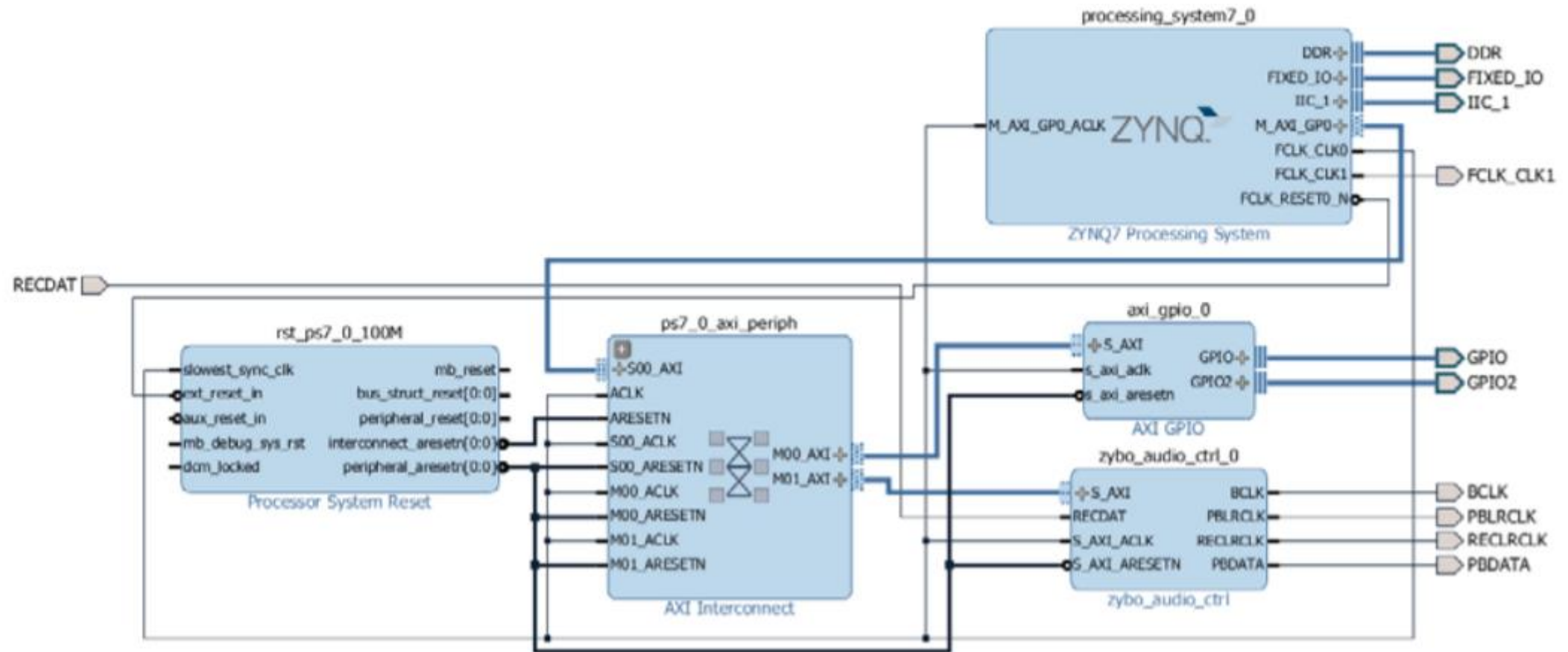
Script 가 실행되고 아래에 표시된 초기 시스템이 생성된다.



# ZedBoard



# Zybo



## 7-2. IP Catalog 에 HLS IP 추가

7-2-1. Flow Navigator Pane 에 Project Manager 에서 설정을 클릭하라.

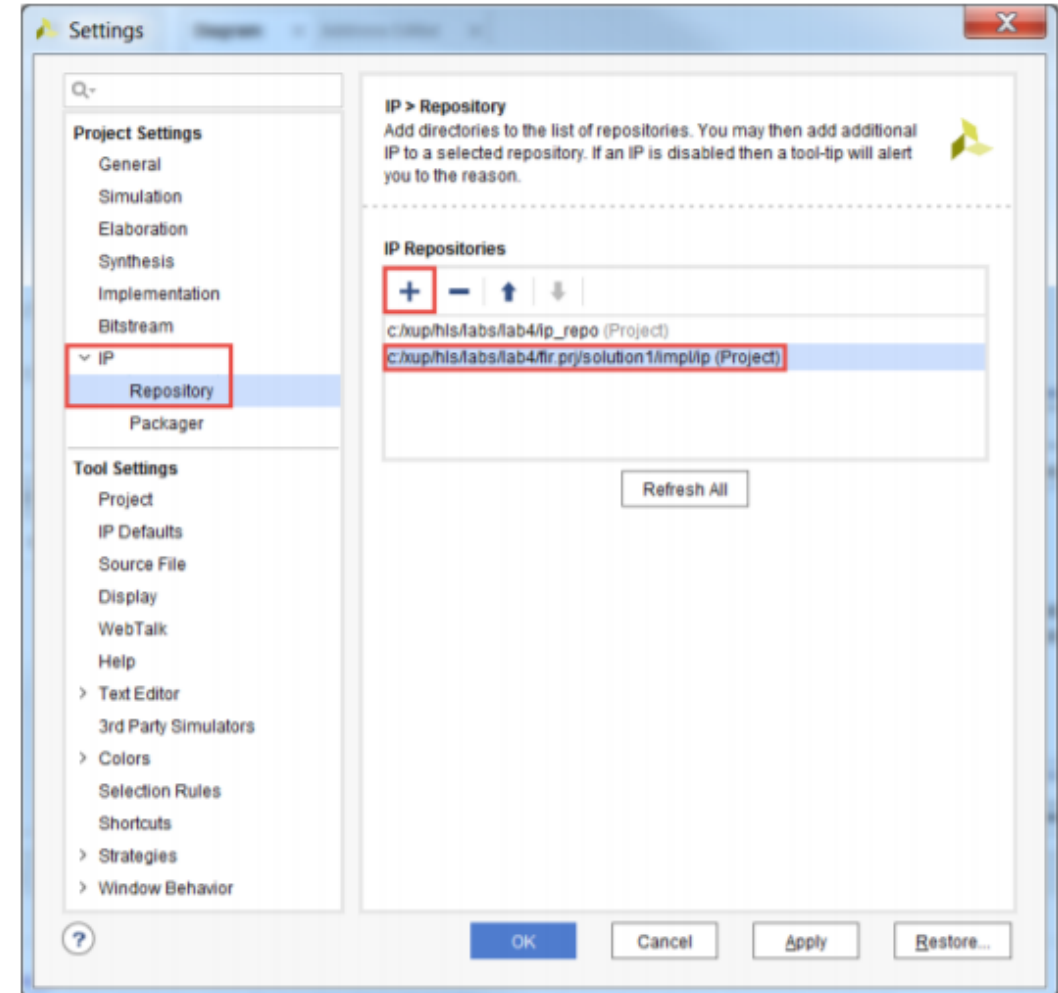
7-2-2. IP > Repository 를 확장하라.

7-2-3. + 버튼을 클릭하라(lab4/ip\_repo 디렉토리가 이미 추가되었다)  
c:\xup\whls\labs\lab4\wfir.prj\wfsolution1\wimpl\wip 로 이동하고 Select 를 클릭한다.

디렉토리가 스캔되어 IP 저장소에 추가되고 하나의 IP 항목이 감지된다.

7-2-4. OK 를 클릭한다.

7-2-5. OK 를 클릭하여 설정을 적용한다.



7-3. Instance 를 fir\_left 및 fir\_right 로 명명하는 처리 시스템에  
fir\_top Core 를 각 측면 채널에 대해 두 번 Instance 화 한다.

7-3-1. IP 추가 아이콘을 클릭하고 Fir 을 입력하여  
Catalog 에서 Fir 을 검색하고 Fir 항목을 더블 클릭하여 Instance 에 추가한다.  
추가 된 IP 에는 Vivado HLS 에서 만든 HLS 로고가 있음을 알 수 있다.

7-3-2. Diagram 에서 추가 된 Instance 를 선택하고  
왼쪽의 Block 속성 양식의 이름 필드에 Instance 이름을 입력하여 Instance 이름을 fir\_left 로 변경한다.

7-3-3. 마찬가지로 HLS IP 의 다른 Instance 를 추가하고 이름을 fir\_right 로 지정한다.

7-3-4. Run Connection Automation 을 클릭하고 All Automation 을 선택한다.

7-3-5. /fir\_left/s\_axi\_fir\_io 및 /fir\_right/s\_axi\_fir\_io 를 클릭하고  
둘 다 M\_AXI\_GP0 에 연결되는지 Verify 하고 OK 를 클릭하라.

7-4. PS-PL Interrupt Ports > IRQ\_F2P 포트를 활성화하라.

두 개의 단일 비트 입력 Port 를 가진 concat IP 의 Instance 를 추가하라.

입력 Port 를 두 개의 FIR Instance 의 Interrupt Port 에 연결하고

출력 Port 를 processing\_system7\_0 Instance 의 IRQ\_F2P Port 에 연결한다.

7-4-1. processing\_system7\_0 Instance 를 두 번 클릭하여 Re-Customization 양식을 Open 한다.

7-4-2. 왼쪽 창에서 Interrupt 를 선택하고 오른쪽의 Fabric Interrupt 확인란을 클릭하라.

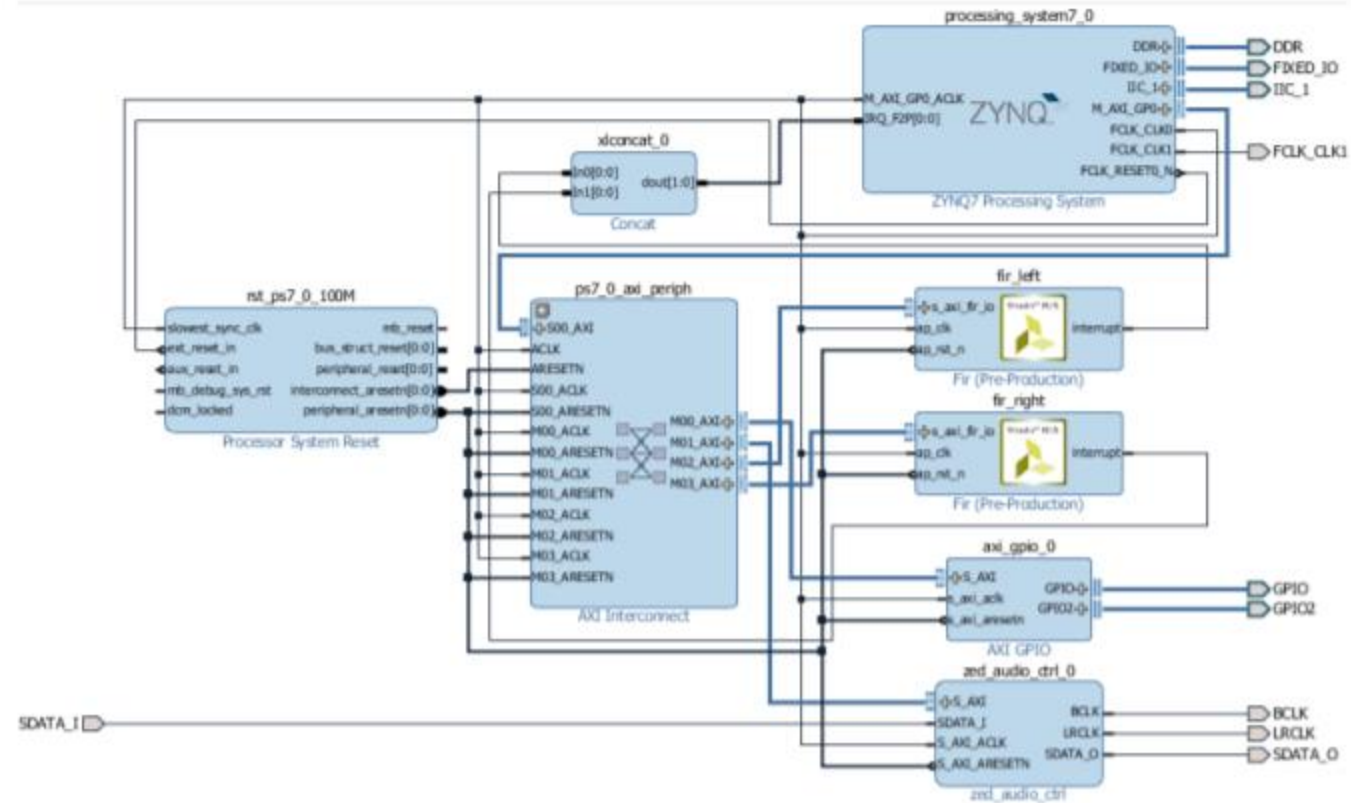
7-4-3. 오른쪽에 있는 Fabric Interrupt > PL-PS Interrupt Port > IRQ\_F2P 항목을 확장하고  
IRQ\_F2P[15:0] 의 Check-Box 를 클릭하라.

7-4-4. OK 를 클릭한다.

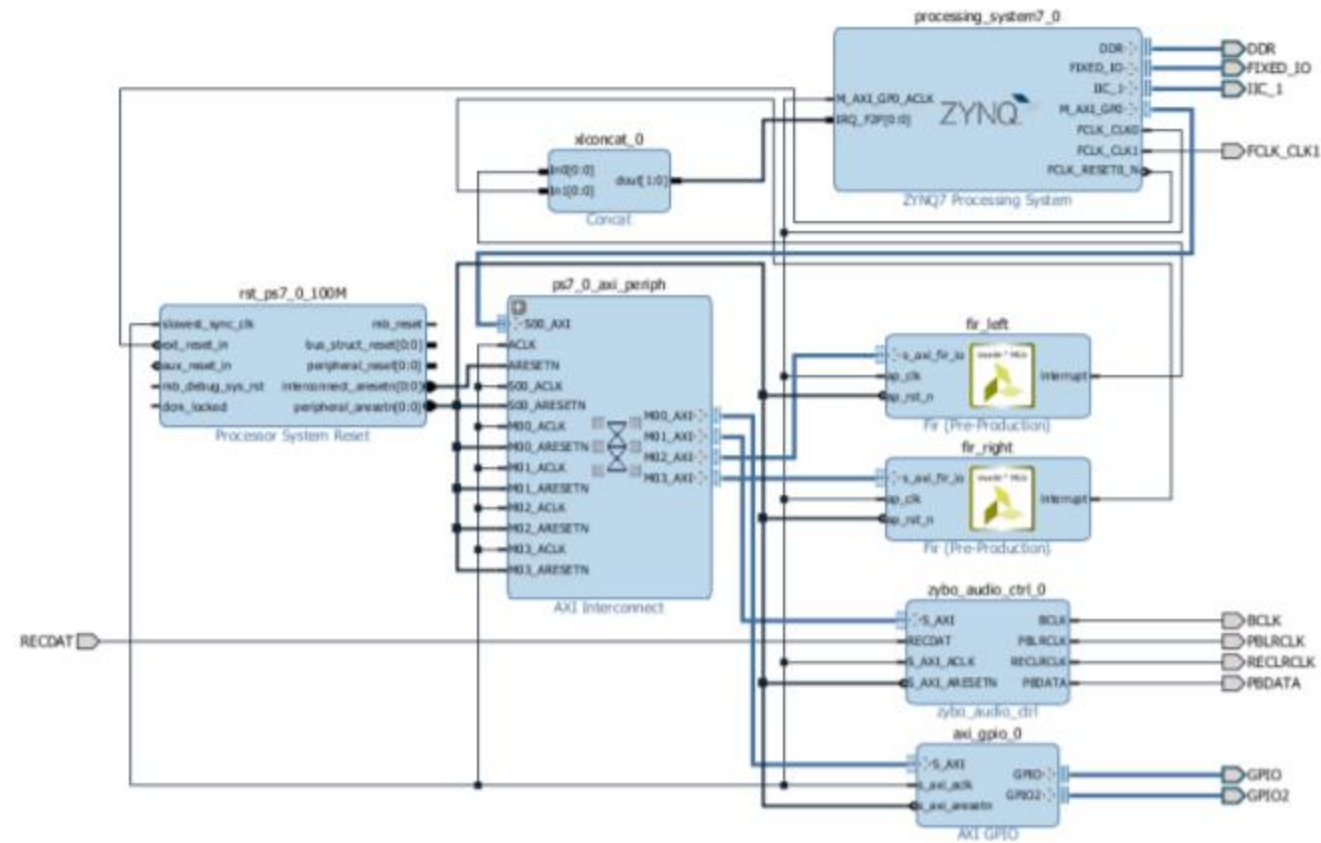
7-4-5. concat IP 의 Instance 를 추가하라.

7-4-6. 각 FIR Instance 의 Interrupt Port 를 xlconcat\_0 Instance 의 두 입력 Port 에 연결하라.

7-4-7. xlconcat\_0 Instance 의 출력 Port 를 processing\_system7\_0 Instance 의 IRQ\_F2P Port 에 연결한다.  
이 단계에서 Design 은 아래 그림과 같아야 한다(재생성 버튼을 클릭해야 할 수도 있음)



(a) ZedBoard



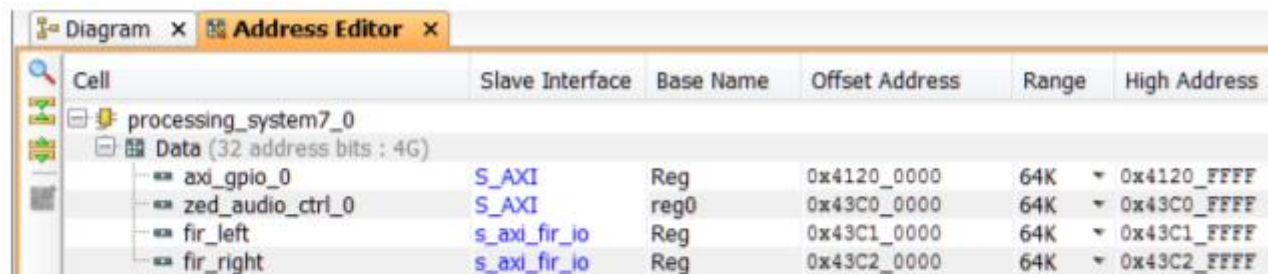
(b) Zybo

7-5. 주소를 확인하고 설계를 Verify 한다.

system\_wrapper 파일을 생성하고 제공된 Xilinx Design Constraints(XDC)를 추가한다.

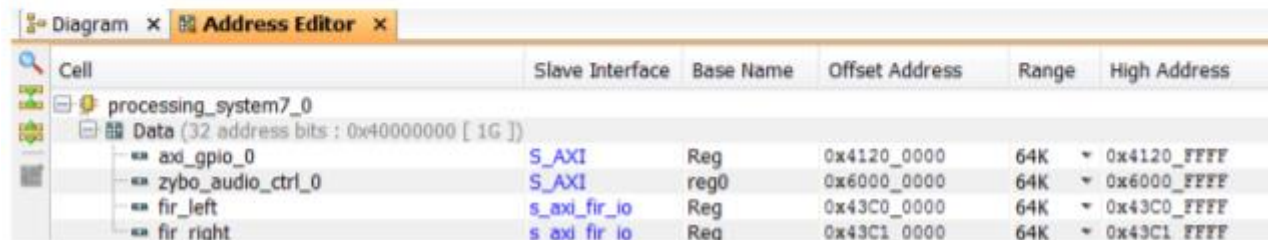
7-5-1. 주소 편집기를 클릭하고 필요한 경우 processing\_system7\_0 > Data 를 확장하라.

생성된 주소 맵은 아래와 같다.



Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 4G)					
axi_gpio_0	S_AXI	Reg	0x4120_0000	64K	0x4120_FFFF
zed_audio_ctrl_0	S_AXI	reg0	0x43C0_0000	64K	0x43C0_FFFF
fir_left	s_axi_fir_io	Reg	0x43C1_0000	64K	0x43C1_FFFF
fir_right	s_axi_fir_io	Reg	0x43C2_0000	64K	0x43C2_FFFF

(a) ZedBoard



Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 0x40000000 [ 1G ])					
axi_gpio_0	S_AXI	Reg	0x4120_0000	64K	0x4120_FFFF
zybo_audio_ctrl_0	S_AXI	reg0	0x6000_0000	64K	0x6000_FFFF
fir_left	s_axi_fir_io	Reg	0x43C0_0000	64K	0x43C0_FFFF
fir_right	s_axi_fir_io	Reg	0x43C1_0000	64K	0x43C1_FFFF

(b) Zybo



7-5-2. Design Validation 을 실행하고(Tools > Validate Design) 오류가 없는지 확인하라.

7-5-3. Source View 에서 Block Diagram 파일 system.bd 를 우클릭하고  
HDL Reference 생성을 선택하여 HDL Wrapper 파일을 업데이트한다.  
메시지가 나타나면 Vivado 가 Wrapper 및 자동 업데이트 관리 옵션을 선택하고 OK 를 클릭한다.

7-5-4. Flow Navigator 창에서 Add Sources 를 클릭하고  
Add or Create Constraints 를 선택한 후 Next 를 클릭한다.

7-5-5. Add Files 버튼을 클릭하고 c:\Wxup\Whls\Wlabs\Wlab4 폴더로 이동하여  
zed\_audio\_constraints.xdc 또는 zybo\_audio\_constraints.xdc 를 선택한다.

7-5-6. Copy constraints files into project 를 클릭한 다음 Finish 를 클릭하여 파일을 추가한다.

7-5-7. Flow Navigator 에서 Generate Bitstream 을 클릭하여  
Synthesis, Implementation, 그리고 bitstream generation 절차를 실행한다.

7-5-8. 절차를 시작한다는 메시지가 표시되면 Save, Yes, 그리고 OK 를 클릭한다.

7-5-9. 비트 생성이 완료되면 Open Implemented Design 옵션이 선택된 선택 상자가 표시되는데 Cancel 을 클릭한다.

## Export to SDK and create Application Project

8-1. 생성된 bitstream 과 함께 HW 를 SDK 로 export 한다.

8-1-1. File > Export > Export Hardware ... 을 선택한다.

8-1-2. Include Bitstream 옵션이 선택되어 있는지 확인하고 Target 디렉토리로 설정한 상태에서 확인을 클릭하라.

8-1-3. File > Launch SDK 를 선택한다.

8-1-4. OK 를 클릭한다.

8-1-5. SDK 에서 File > New > Board Support Package 를 선택한다.

8-1-6. 기본 설정으로 Finish 를 클릭한다(독립형 운영 체제 사용)  
그러면 OS 및 Library 선택 항목을 보여주는 Software Platform Setting 양식이 열린다.

8-1-7. 추가적인 라이브러리를 지원하지 않고 standalone\_bsp\_0 소프트웨어 플랫폼 프로젝트를 만들려고하므로 OK 를 클릭하여 기본 설정을 적용하라.  
라이브러리 생성기는 백그라운드에서 실행되며  
C:\xup\whlslabs\lab4\audio\aduio.sdk\standalone\_bsp\_0\ps7\_cortexa9\_0\include 디렉토리에 xparameters.h 파일을 만든다.

8-1-8. File > New > Application Project 를 선택한다.

8-1-9. 프로젝트 이름으로 TestApp 을 입력하고 Board Support Package 의 경우 기존 사용(standalone\_bsp 가 유일한 옵션이어야 함)을 선택한다.

8-1-10. Next 를 클릭하고 Empty Application 을 선택하고 Finish 를 클릭한다.

8-1-11. Project View 에서 TestApp 을 선택하고 src 폴더를 우클릭 한 다음 Import 를 선택한다.

8-1-12. General Category 를 확장하고 File System 을 더블 클릭한다.

8-1-13. C:\xup\whlslabs\lab4 폴더로 이동하고 OK 를 클릭한다.

8-1-14. ZedBoard 의 경우 zed\_testapp.c 및 zed\_audio.h 를 선택하고 Zybo 의 경우  
zybo\_testapp.c 및 zybo\_aduio.h 를 선택하고 Finish 를 클릭하여 File 을 Project 에 추가한다.  
프로그램이 성공적으로 컴파일 되어야 한다.

## Verify the Design in Hardware

9-1. Zybo: JP7 이 USB 전원을 선택하도록 설정되었는지 확인하라.

PC 와 Board 의 JTAG Port 사이에 micro USB 케이블을 연결하라.

PC 의 라인 입력 잭과 스피커(헤드폰) 출력 잭 사이에 Audio patch 케이블을 연결하라.

헤드폰을 Board 의 Line 출력 잭(ZedBoard) 또는 HPH OUT(Zybo)에 연결하라.

Board 의 전원을 켜라.

9-1-1. Zybo 전용: JP7 이 USB 전원을 선택하도록 설정되어 있는지 확인하라.

9-1-2. PC 와 Board 의 JTAG Port 사이에 micro USB 케이블을 연결하라.

9-1-3. PC 의 Line 입력 잭과 스피커(헤드폰) 출력 잭 사이에 Audio 패치 케이블을 연결하라.

9-1-4. Zybo Board 의 Line 출력 잭 또는 Zybo 보드의 HPH 출력 잭에 헤드폰을 연결하라.  
보드의 전원을 켜다.

9-1-5. Xilinx > Program FPGA 를 선택한다.

9-1-6. system\_wrapper.bit 비트 스트림이 선택되고 BMM 파일 필드가 비어 있는지 확인한다.

9-1-7. Program 을 클릭한다.  
그러면 FPGA 가 구성된다.

9-1-8. corrupted\_music\_4KHz.wav 또는 관심있는 다른 wave 파일을 더블 클릭하여 설치된 media player 를 사용하여 재생하라.  
continuous play mode 로 설정한다.

9-1-9. Project Explorer 창에서 TestApp 을 우클릭하고 Run As > Launch On Hardware(System Debugger) 를 선택한다.  
프로그램이 다운로드 되어 실행된다.  
손상된 신호를 듣고 싶다면 SW0 을 OFF 로 설정한다.  
필터링 된 신호를 들으려면 SW0 을 ON 으로 설정한다.

9-1-10. 완료되면 Board 의 전원을 끈다.

9-1-11. File > Exit 를 사용하여 SDK 및 Vivado 를 종료한다.

## Conclusion

이 Lab 에서 RESOURCE 지시어를 추가하여 IP-XACT 어댑터를 작성했다.  
Implementation 단계에서 IP-XACT 어댑터를 생성했다.  
그런 다음 IP Integrator 를 사용하여 Processor System 을 만들고  
생성된 IP-XACT 어댑터를 통합한 다음 제공된 Application 으로 System 을 테스트했다.

## Answers

1. Answer the following questions:

Estimated clock period:	<u>8.70 ns (zedboard) 7.38 ns (zybo)</u>
Worst case latency:	<u>174 (zedboard) 175 (zybo) clock cycles</u>
Number of DSP48E used:	<u>3</u>
Number of BRAMs used:	<u>0</u>
Number of FFs used:	<u>167 (zedboard) 168 (zybo)</u>
Number of LUTs used:	<u>154 (zedboard) 157 (zybo)</u>

# Appendix

## Create a Project using Vivado GUI

10-1. Vivado 를 띄우고 ZedBoard(xc7z020clg484-1 Device) 또는 Zybo(xc7z010clg400-1 Device) 및 Verilog 언어를 사용하여 Empty Project 를 만든다.

10-1-1. Start > All Programs > Xilinx Design Tools > Vivado 2017.4 > Vivado 2017.4(리눅스 별도)

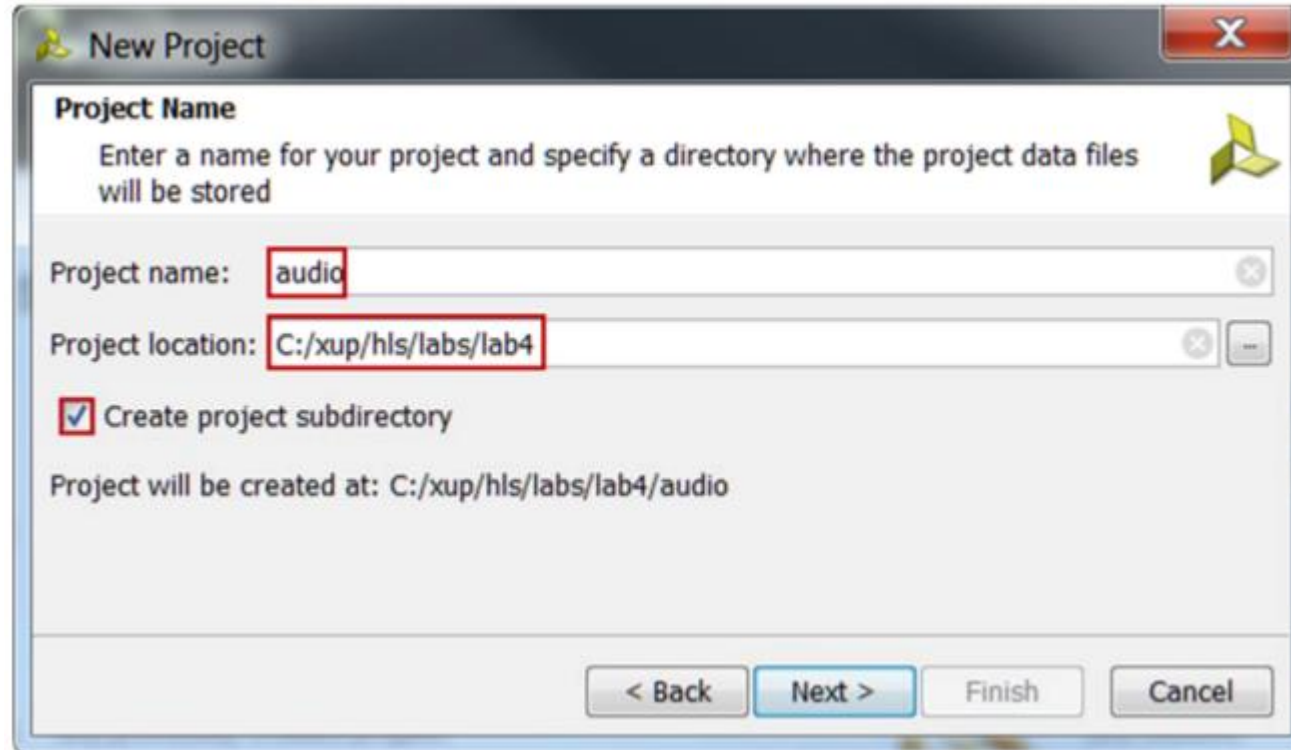
10-1-2. Create New Project 를 클릭하여 마법사를 시작한다.

Create a New Vivado Project Dialog Box 를 볼 수 있을 것이다.  
Next 를 클릭한다.

10-1-3. New Project 양식의 Project Location 필드에서 Browse 버튼을 클릭하고 c\Wxup\Hls\labs\lab4 로 이동한 다음 Select 를 클릭한다.

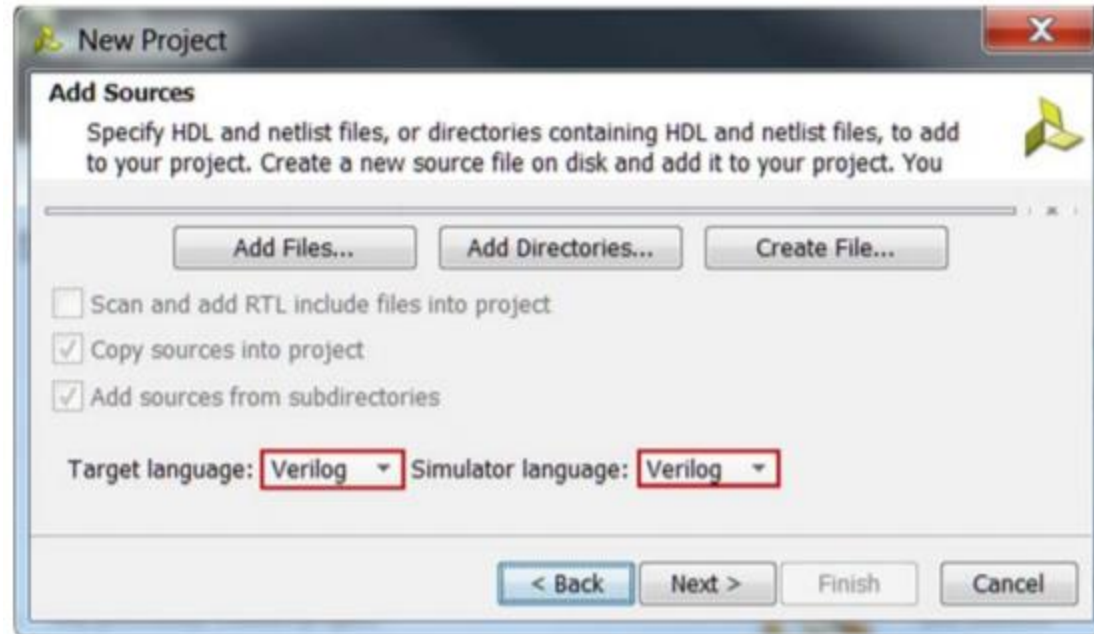
10-1-4. Project Name 필드에 audio 를 입력한다.

Create Project Subdirectory 박스가 체크되었는지 확인한다.



10-1-5. Project Type 양식에서 RTL Project 를 선택하고 Next 를 클릭한다.

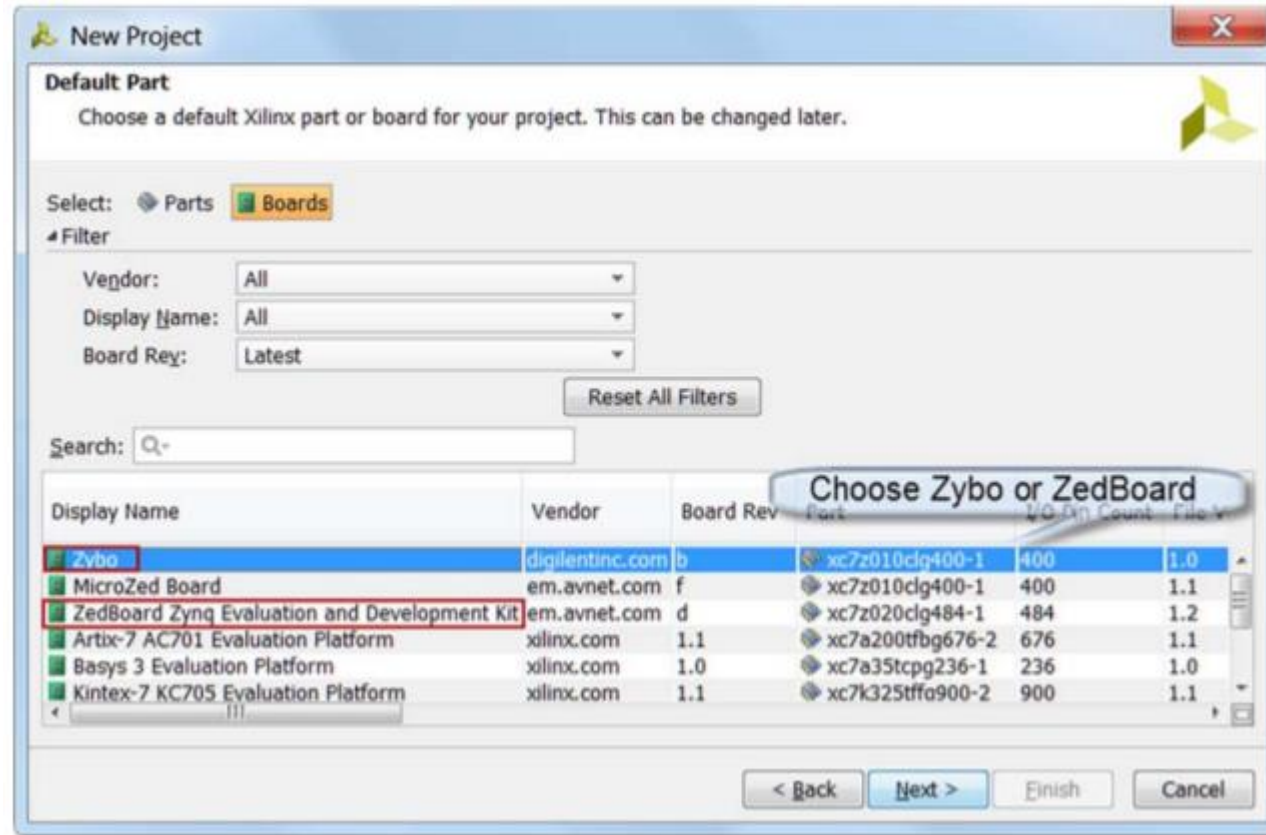
10-1-6. Add Sources 양식에서 Verilog 를 Target 언어로 선택하고 Simulator Language 를 선택하고 Next 를 클릭한다.



10-1-7. Next 를 두 번 클릭하여 Adding Existing IP 및 Add Constraints Dialog Boxes 를 Skip 한다.

10-1-8. Default Part 양식에서 Boards 를 선택하고 ZedBoard Zynq Evaluation and Development Kit 혹은 Zybo 를 선택한다.  
그리고 Next 를 클릭한다.

\* Zybo 항목이 보이지 않고 Zybo 보드를 타겟팅하려면 readme\_zybo.docx 파일을 읽고 zybo 보드 파일을 Vivado 설치 디렉토리에 설치하라.



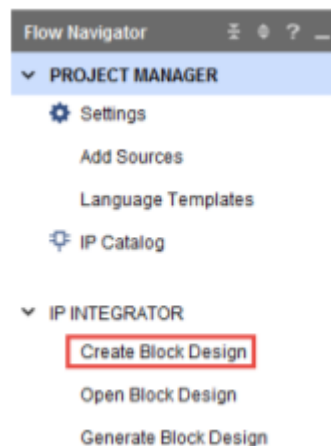
10-1-9. Project Summary 를 체크하고 Finish 를 클릭하여 empty Vivado 프로젝트를 만든다.



## Creating the System Using the IP Integrator

11-1. IP Integrator 를 사용하여 새로운 Block Design 을 생성하고 ARM Cortex-A9 Processor 기반 Hardware System 을 생성한다.

11-1-1. Flow Navigator 에서 IP Integrator 아래의 Create Block Design 을 클릭한다.

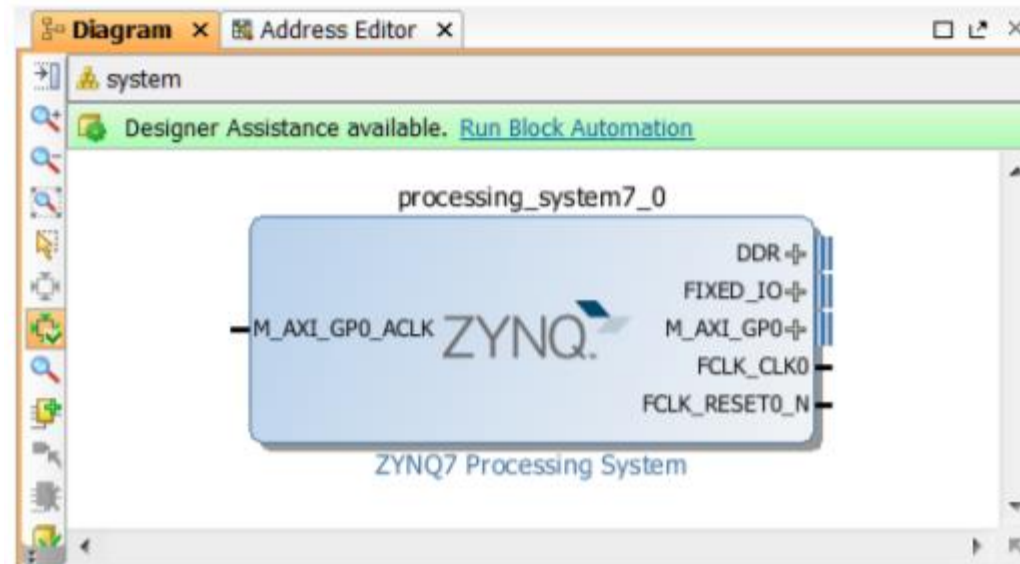


11-1-2. Design 이름에 대한 System 을 입력하고 OK 를 클릭한다.

11-1-3. Catalog 에서 IP 를 다른 방법으로 추가 할 수 있다.

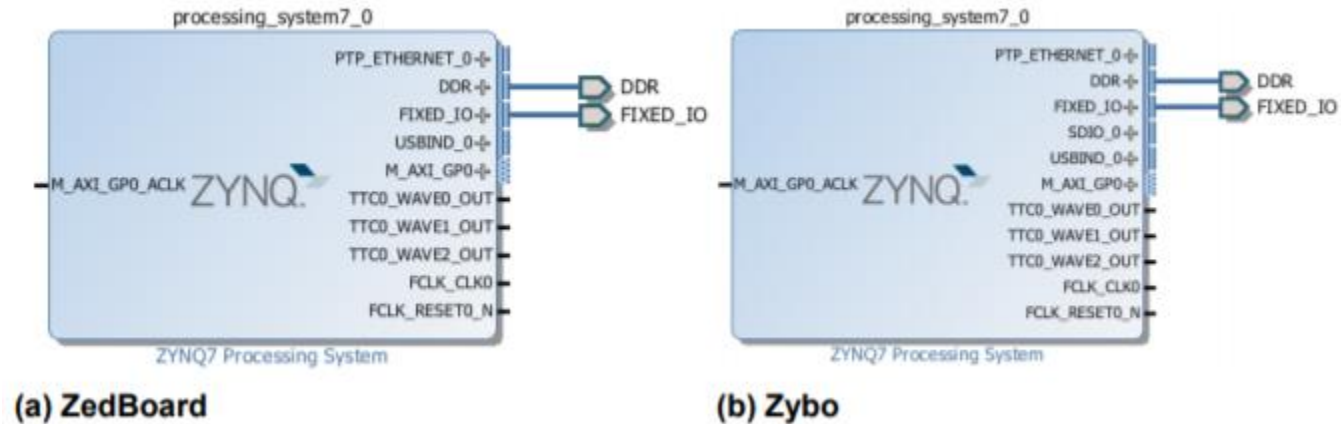
Diagram 패널 상단의 메시지에서 Add IP 를 클릭하거나  
Block Diagram Side Bar 에서 Add IP 아이콘을 클릭하거나 Ctrl + I 를 누르거나,  
Diagram Workspace 을 우클릭하고 Add IP 를 선택한다.

- 11-1-4. IP Catalog 가 열리면 검색 창에 "zy" 를 입력하고  
ZYNQ7 Processing System 항목을 찾아 더블 클릭하거나 Entry 를 클릭하고 Enter 키를 눌러 Design 에 추가하라.  
Zynq Block 이 추가 된다.



- 11-1-5. Designer Assistance 를 사용할 수 있는 Diagram 창의 맨 위에 있는 Message 를 확인하라.  
Run Block Automation 을 클릭하고 processing\_system7\_0 을 선택하라.

- 11-1-6. Automation 을 실행할지 묻는 메시지가 나타나면 OK 를 클릭한다.  
Block Automation 이 완료되면 외부 Port 가 DDR 및 Fixed IO 에 자동으로 추가된다.  
다른 기본 Port 중 일부가 Block 에도 추가된다.



- 11-1-7. Block Diagram 에서 Zynq Block 을 더블 클릭하여 Zynq Processing System 의 Customization Window 를 Open 한다.  
이제는 Zynq 의 Block Diagram 이 열려 Processing System 의 구성 가능한 다양한 Block 이 표시된다.  
이 단계에서 Designer 는 구성 가능한 다양한 Block(녹색으로 강조 표시됨)을 클릭하고 System 구성을 변경할 수 있다.

- 11-2. I/O Peripheral Block 을 구성하여 UART1 및 I2C1 Peripheral 을 사용하고  
다른 원하지 않는 Peripheral 을 비활성화 한다.  
Timer0 을 선택 해제 한다.  
PL Fabric Clock 인 FCLK\_CLK1 을 활성화하고 ZedBoard 의 경우  
10.000 MHz 또는 Zybo 의 경우 12.288 MHz 로 해당 주파수를 설정한다.
- 11-2-1. 왼쪽에 있는 MIO Configuration(MIO 구성) 탭을 선택하여  
Configuration 창을 열고 오른쪽 창에서 I/O Peripheral(I/O 주변 장치)을 확장한다.
- 11-2-2. I2C1 Peripheral 의 체크 박스를 클릭한다.  
필요하지 않은 USB0, SD 0, ENET 0, GPIO > GPIO MIO 를 필요로 하지 않는다.
- 11-2-3. MIO Configuration 선택 탭에서  
Application Processing Unit 그룹을 확장하고 Timer 0 을 선택 해제한다.
- 11-2-4. 왼쪽 창에서 Clock Configuration 을 선택하고  
오른쪽의 PL Fabric Clock 항목을 확장 한 다음 FCLK\_CLK1 의 체크 박스를 클릭한다.
- 11-2-5. ZedBoard 의 경우 FCLK\_CLK1 의 요청 빈도 값을  
10.000 MHz 로 변경하고 Zybo 의 경우 12.288 MHz 로 변경한다.

Peripheral I/O Pins	Component	Clock Source	Requested Frequ...
MIO Configuration	+	Processor/Memory Clocks	
Clock Configuration	+	IO Peripheral Clocks	
DDR Configuration	-	PL Fabric Clocks	
SMC Timing Calculation	<input checked="" type="checkbox"/>	FCLK_CLK0	IO PLL 100.000000
Interrupts	<input checked="" type="checkbox"/>	FCLK_CLK1	IO PLL 10.000000
	<input type="checkbox"/>	FCLK_CLK2	IO PLL 50.000000
	<input type="checkbox"/>	FCLK_CLK3	IO PLL 50

(a) ZedBoard

Component	Clock Source	Requested Frequ...	Actual Frequency...	Range(MHz)
Timers				
System Debug Clocks				
Processor/Memory Clocks				
PL Fabric Clocks				
<input type="checkbox"/> FCLK_CLK3	IO PLL	50	50.000000	0.100000 : 250.000000
<input type="checkbox"/> FCLK_CLK2	IO PLL	50	50.000000	0.100000 : 250.000000
<input checked="" type="checkbox"/> FCLK_CLK1	IO PLL	12.288	12.280702	0.100000 : 250.000000
<input checked="" type="checkbox"/> FCLK_CLK0	IO PLL	100	100.000000	0.100000 : 250.000000

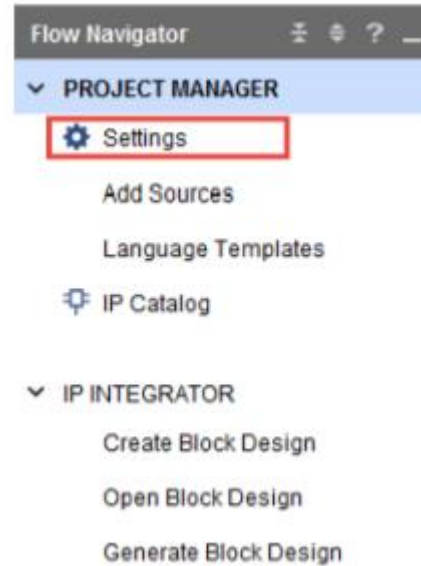
(b) Zybo

11-2-6. OK 를 클릭한다.

Zynq Block 은 필요한 Port 만 보여준다.

11-3. 제공된 I2C 기반 ZedBoard 용 zed\_audio\_ctrl IP  
또는 Zybo 용 zybo\_audio\_ctrl IP 를 IP Catalog 에 추가한다.

11-3-1. Flow Navigator 창에서 Project Manager 아래의 Settings 를 클릭한다.  
IP Catalog 가 열린다.

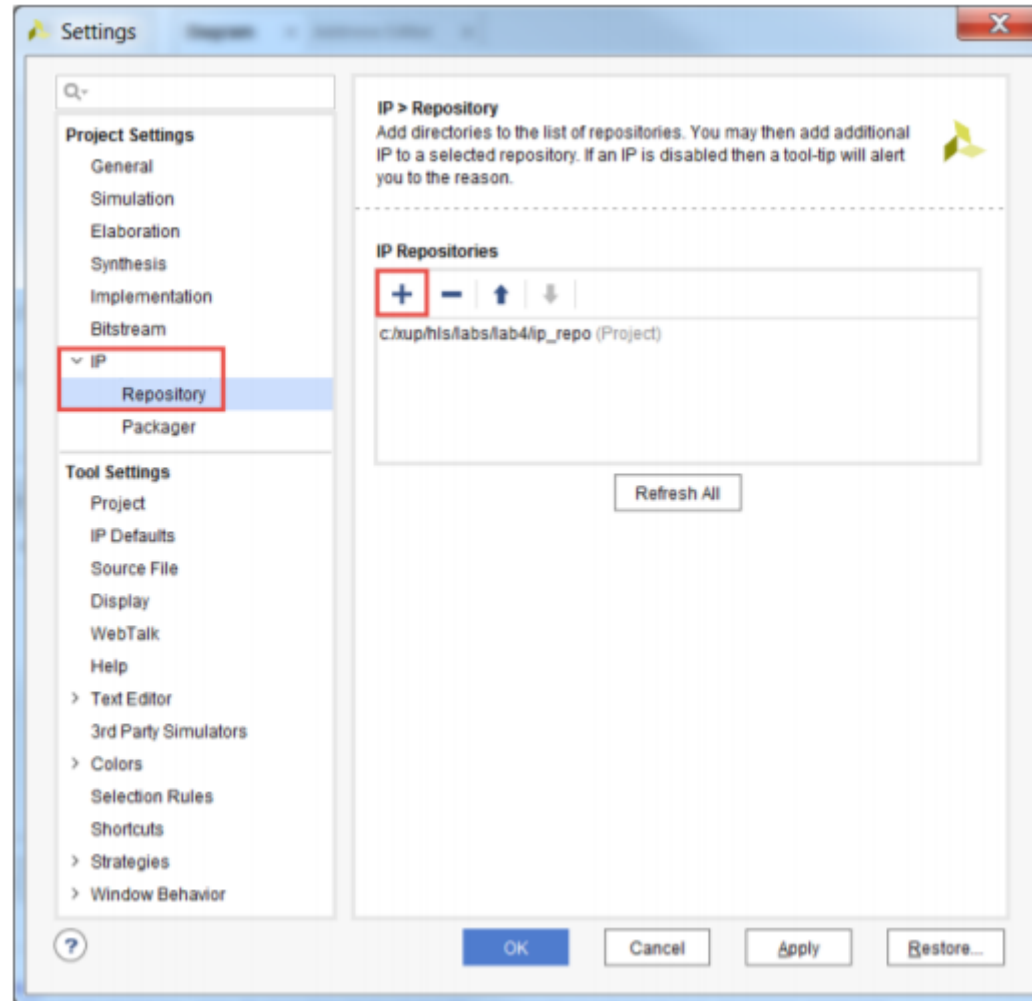


11-3-2. 왼쪽 창에서 IP > Repository 를 확장하라.

11-3-3. + 버튼을 클릭한다.

c:\WxupWhlsWlabsWlab4Wip\_repo 를 찾아 Select 를 클릭한다.

디렉토리가 스캔되어 IP 저장소 창에 추가되고 두 개의 IP 항목이 감지된다.



11-3-4. OK 를 클릭하여 설정을 적용한다.

- 11-4. ZedBoard: 채널 1 의 2 비트 폭과 채널 2 의 1 비트 입력 폭을 사용하여 zed\_audio\_ctrl 및 GPIO 를 Instance 화 한다.  
Zybo: 채널 1 에서만 1 비트 출력의 너비와 채널 2 에서만 1 비트 너비의 입력으로 zybo\_audio\_ctrl 및 GPIO 를 Instance 화 한다.  
Connection Automation 을 실행하여 상호간 연결하도록 한다.
- 11-4-1. IP Catalog 가 열려 있지 않으면 Add IP 버튼을 클릭하고  
gpio 를 입력하여 Catalog 에서 AXI GPIO 를 검색하고 AXI GPIO 항목을 더블 클릭하여 Instance 를 추가한다.
- 11-4-2. Add IP to Block Design 버튼을 클릭한다.
- 11-4-3. 추가된 Instance 를 더블 클릭하면 Re-Customized IP GUI 가 표시된다.
- 11-4-4. 채널 1 의 너비를 ZedBoard 의 경우 2 로 변경하거나 Zybo 의 경우에만 1 의 너비로 변경한다.
- 11-4-5. Enable Dual Channel 상자를 체크하고 너비를 입력 1 로 설정한 다음 OK 를 클릭한다.
- 11-4-6. 마찬가지로 ZedBoard 용 zed\_audio\_ctrl 또는 Zybo 용 zybo\_audio\_ctrl 중 하나의 Instance 를 추가한다.
- 11-4-7. Design 지원을 사용할 수 있다.  
Run Connection Automation 을 클릭하고 axi\_gpio\_0/S\_AXI 를 선택한다.
- 11-4-8. OK 를 클릭하여 M\_AXI\_GP0 인터페이스에 연결한다.  
두 개의 추가 Block 인 Proc Sys Reset 과 AXI Interconnect 가 자동으로 설계에 추가되었다.
- 11-4-9. 마찬가지로 Run Connection Automation 을 클릭하고  
ZedBoard 의 경우 zed\_audio\_ctrl\_0/S\_AXI 를 선택하고  
Zybo 의 경우 zybo\_audio\_ctrl\_0/S\_AXI 를 선택하고 OK 를 클릭한다.



11-5. IIC\_1, GPIO, FCLK\_CLK1 및 zed\_audio\_ctrl 또는 zybo\_audio\_ctrl Port 를 외부로 만든다.

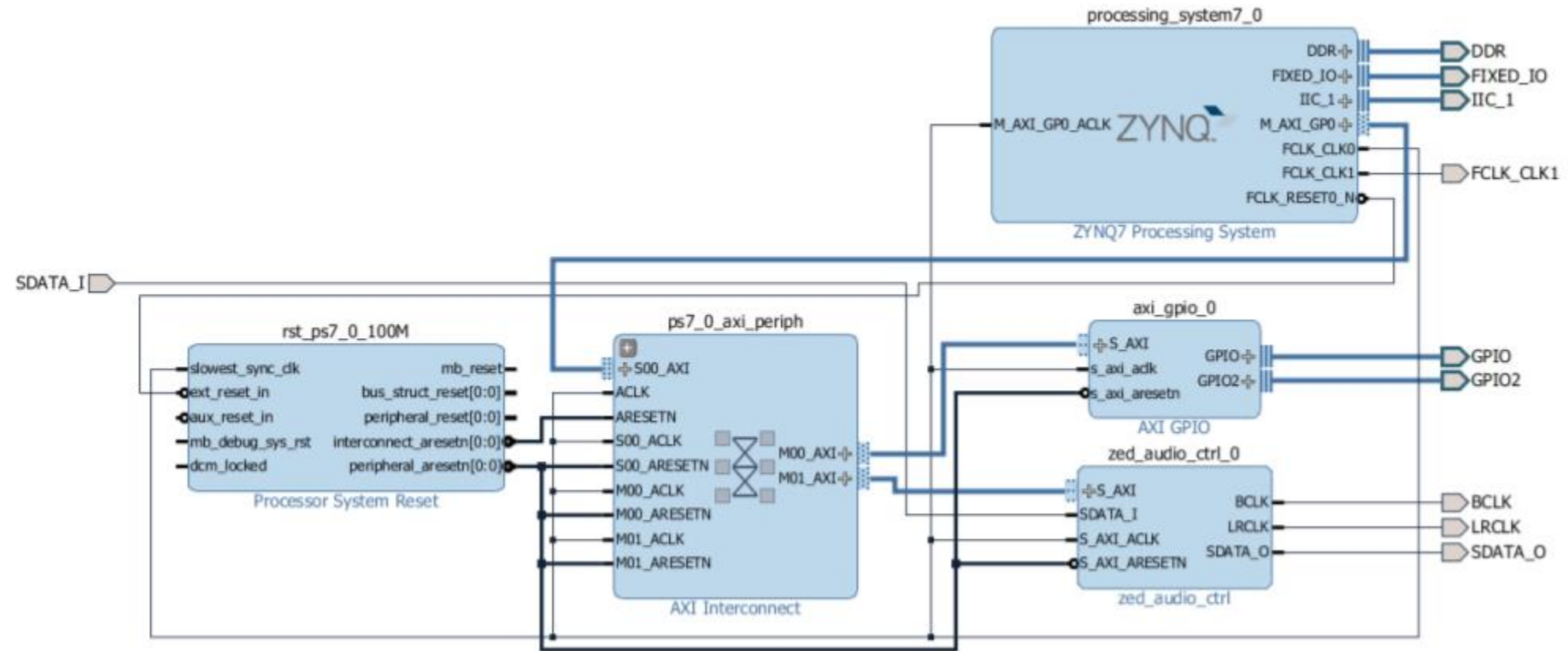
11-5-1. axi\_gpio\_0 Instance 의 GPIO Interface 를 우클릭하고  
Make External 을 선택하여 External Port 를 만든다.  
이렇게 하면 GPIO 라는 External Port 가 만들어지고 Peripheral 에 연결된다.

11-5-2. axi\_gpio\_0 Instance 의 GPIO2 Interface 를 우클릭하여  
Make External 을 선택하여 External Port 를 만든다.

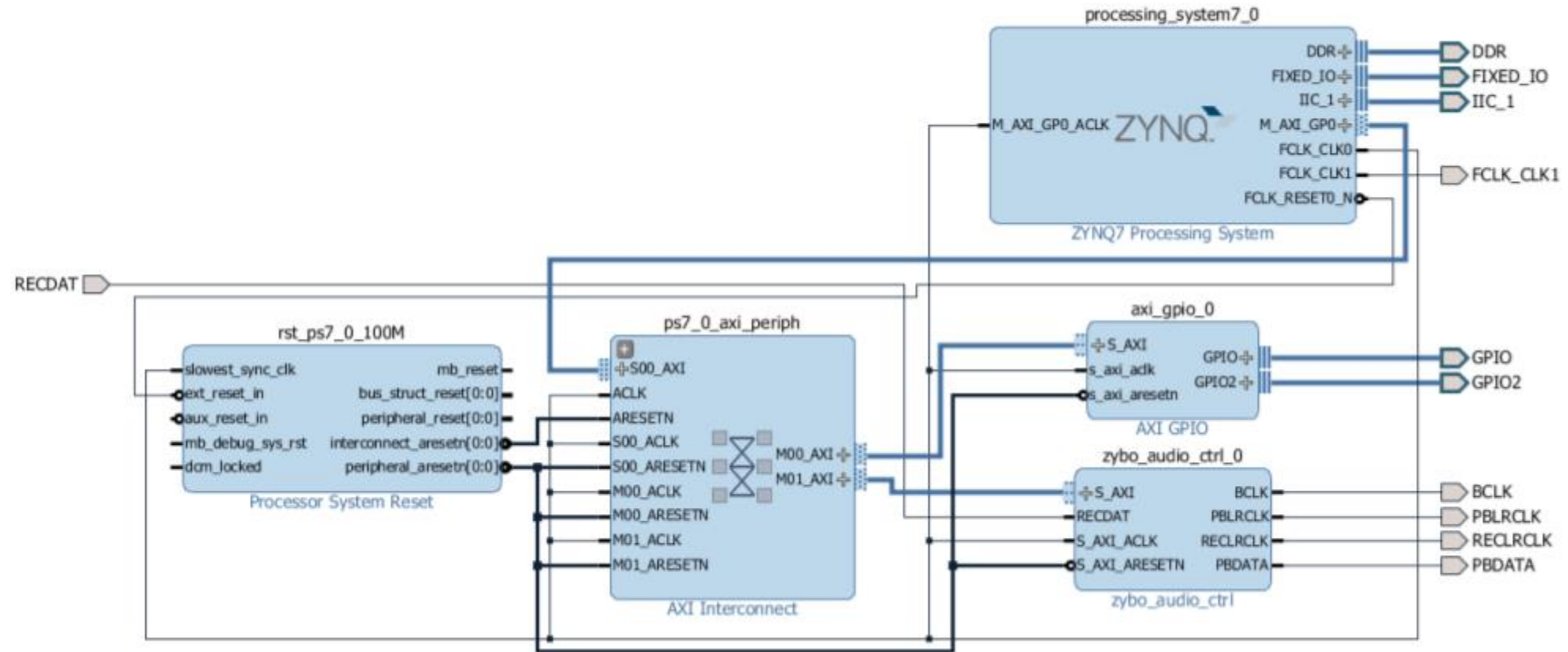
11-5-3. 마찬가지로 zed\_audio\_ctrl\_0 Instance 또는 zybo\_audio\_ctrl\_0 Instance 중  
한 번에 하나의 Port 를 선택하여 External Port 로 만든다.

11-5-4. 마찬가지로 processing\_system7\_0 Instance 의 IIC\_1 Interface 및 FCLK\_CLK1 Port 를 외부로 설정한다.  
이 단계에서 Design 은 아래 그림과 같아야 한다.  
(regenerate 버튼을 클릭해야 할 수도 있다)

# ZedBoard



# Zybo



# References

1. <https://www.xilinx.com/support/university/vivado/vivado-workshops/Vivado-high-level-synthesis-flow-zynq.html>