TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 : Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 : 황수정
sue100012@naver.com
63 일차 (2018. 05. 29)

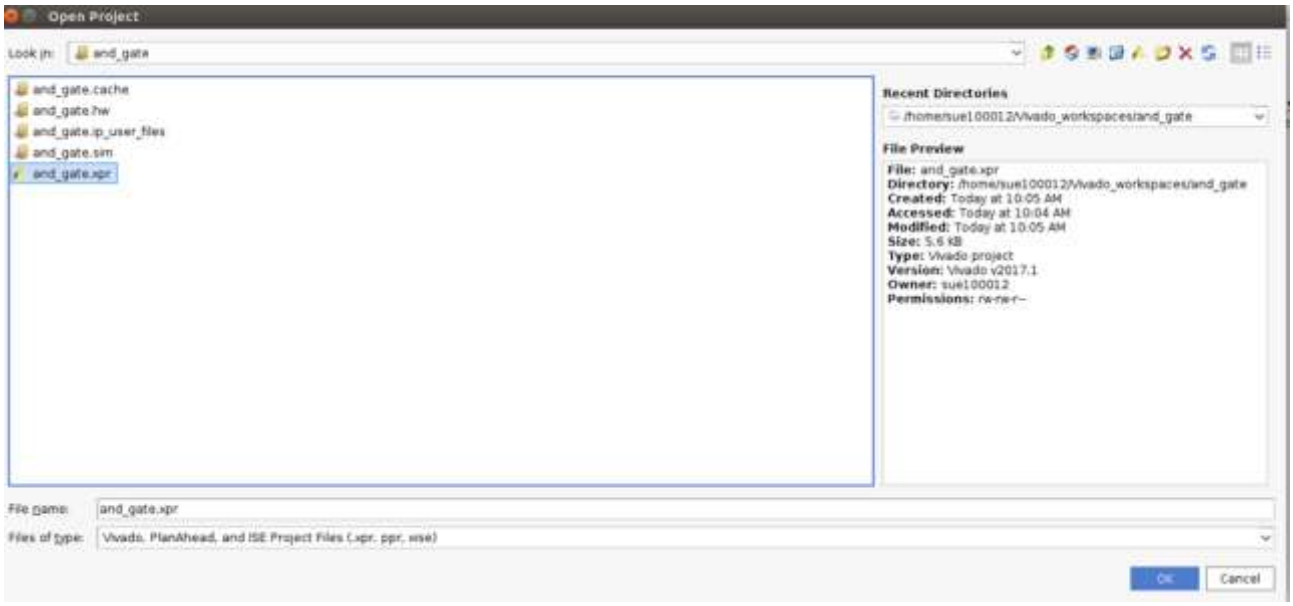
※ 프로젝트 생성

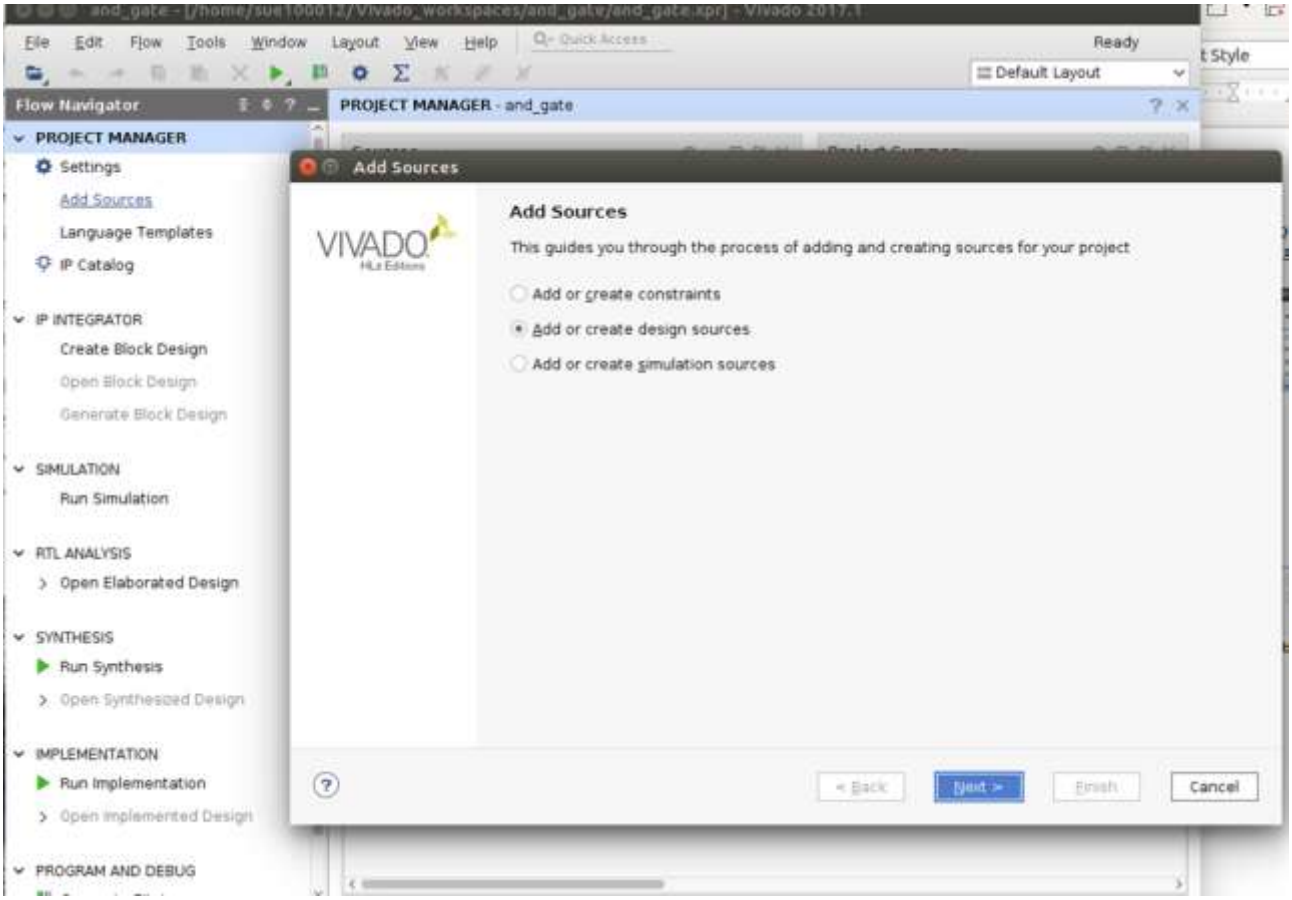
터미널 창에서 Vivado 폴더에 2017.1(버전)→ bin 으로 들어가 vivado 를 실행한다.

새로운 프로젝트를 생성하기를 하고 파일명과 주소(/home/계정명/vivado_workspace)를 지정해주고 next 를 하면 소스를 추가하는 창이 뜬다. next 로 넘어가고 그 다음 옵션 창도 next 한다.

※ 프로젝트 불러오기

생성한 프로젝트를 불러오는 방법은 만들어 놓은 프로젝트로 들어가서 확장자 버전 *.xpr 선택한다.





FPGA (Field-Programmable Gate Array)

칩안에 수많은 논리 게이트(NAND, NOR, Flip-Flop 등) 등이 연결이 되지 않은 채로 들어가 있다. 자신이 만들고자 하는 회로를 설계하여 해당 FPGA 의 CAD 툴(Synthesizer)을 돌리면 게이트들의 결선 정보(Xilinx 에서는 bit file 이라고 하는듯..)로 변환되어 나온다. 이것을 FPGA 에 써주면 원하는 칩으로서 동작하고 이러한 과정은 한번이 아니고 반영구적으로 무한히 가능하므로 여러 회로 얼마든지 만들어 볼 수가 있다.

이미 설계된 하드웨어를 반도체로 생산하기 직전 최종적으로 하드웨어의 동작 및 성능을 검증하기 위해 제작하는 중간 개발물 형태의 집적 회로(IC)다. 반도체 제조업자 측에서 보면 양산되어 일반적 용도로 사용되므로 범용 IC 의 범주에 속하고, 사용자 측에서 보면 사용자 요구에 맞게 프로그래밍하여 사용할 수 있으므로 주문형 반도체(ASIC) 범주에 속한다.

비메모리 반도체의 일종으로, 회로 변경이 불가능한 일반 반도체와 달리 여러 번 회로를 다시 새겨 넣을 수 있는 반도체다. 오류 발생시 수정이 가능하고 개발시간이 짧으며 초기 개발비용이 적게 든다. 그러나 일반적으로 속도가 느리고 복잡한 설계에 적용이 불가하며 소비전력이 크다는 단점이 있다.

즉, **FPGA 는 각종 트랜지스터를 논리적으로 프로그래밍 한 것**을 말한다. 우리가 사용하는 디지털 칩도 결국 내부는 아날로그다. 이 것이 복잡해지면 사람이 구성을 못한다. 그래서 논리를 만들어서 단순화 시킨 것이다. 즉, FPGA 는 트렌지스터 수십억개를 반도체에 집어 넣을 수 있도록 해준다.

```
Port (
    a, b : in std_logic;
    result : out std_logic;
    );
Result <= a and b;
```

예시로 위의 코드를 보자면 이 코드를 이용해 FPGA 로 트랜지스터를 제어한다. 이 코드처럼 c 언어 프로그래밍 하는 것과 비슷하게 만들어주는 것이다. 부품 배치가 암호화 해서 들어간다.

RTL(Registor Transistor Logic. 컴퓨터에 사용되고 있는 기본적인 회로의 하나이며, 저항과 트랜지스터로 구성하는 회로이다. 이 회로는 DCTL 의 베이스회로에 직렬로 저항을 넣어 동작의 안정도를 증가시킨 것이다. 동작속도는 그렇게 빠르지 않다.). 이 처리는 팸리스 업체에서 한다. 그곳에서 기계를 통해 실제 배치를 하게 된다. 반도체 업체는 이 곳을 통할 수 밖에 없다.

FPGA 를 사용하기 위한 환경 설정 중에 보드 설정해주는 것이 있다. 오늘 한 코드 중에 LED 를 켜는 코드는 zybo 보드를 사용해야 하는데, 현재 다운 받은 board 파일에 존재하지 않아 새로 다운 받아주어야 한다.

https://reference.digilentinc.com/reference/software/vivado/board-files

로 가서 압축 파일을 다운 받는다. 다운 받은 파일을 압축을 풀고 vivado 의 하위 폴더 중에 board files 로 들어가야 한다. 하지만 락이 걸려 제한이 있는 파일이 이기 때문에 옮겨지지 않아, 터미널 창에서 sudo 를 이용해서 넣어주어야 한다. 명령어는 다음과 같다.

sudo cp -apr /home/계정주소/파일위치/ /home/~옮길 파일 위치/board_files/

를 해주면 된다.

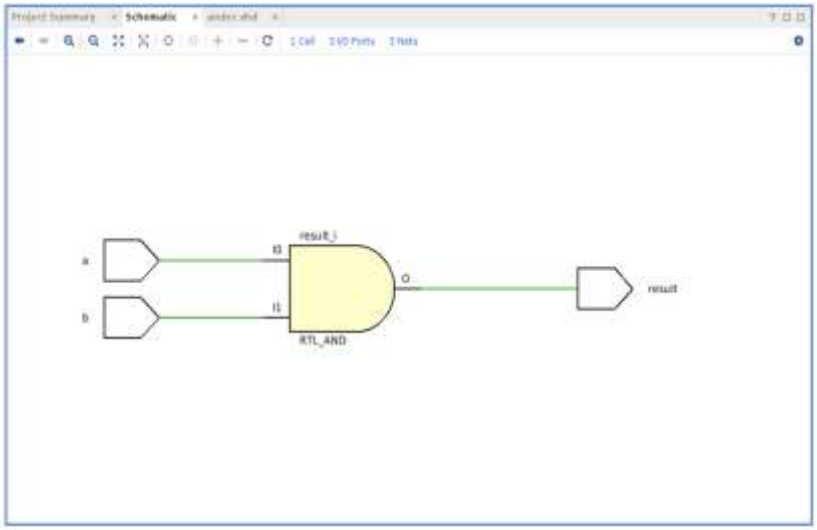
그럼 그 해당파일로 옮겨진다.

Zynq : PL(programmable Logic, Verilog, VHDL) + PS → 리눅스가 올라갈 수 있다.

이러한 예시로 위의 코드를 이용하여 AND gate 를 설정해본다. 다음과 같이 코드를 작성하고 open Elaborated Design 을 해준다.

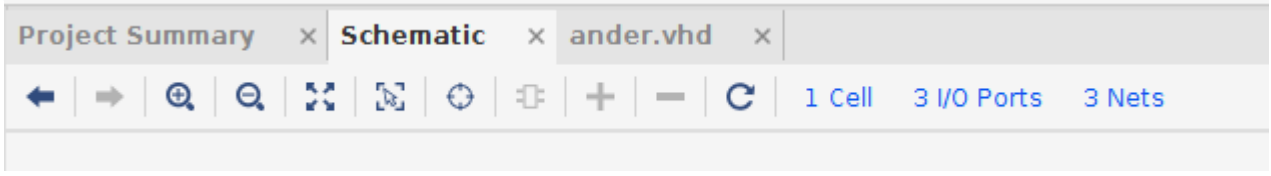
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ander is
--
    Port (
    a, b : in std_logic;
    result : out std_logic
    );


end ander;

architecture Behavioral of ander is

begin

    result <= a and b;

end Behavioral;
```

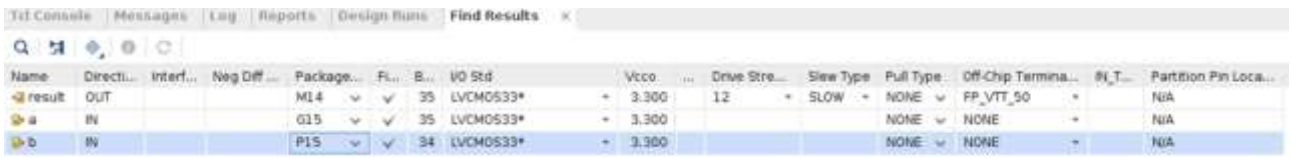
그럼 다음과 같이 and gate 가 뜨게 된다.

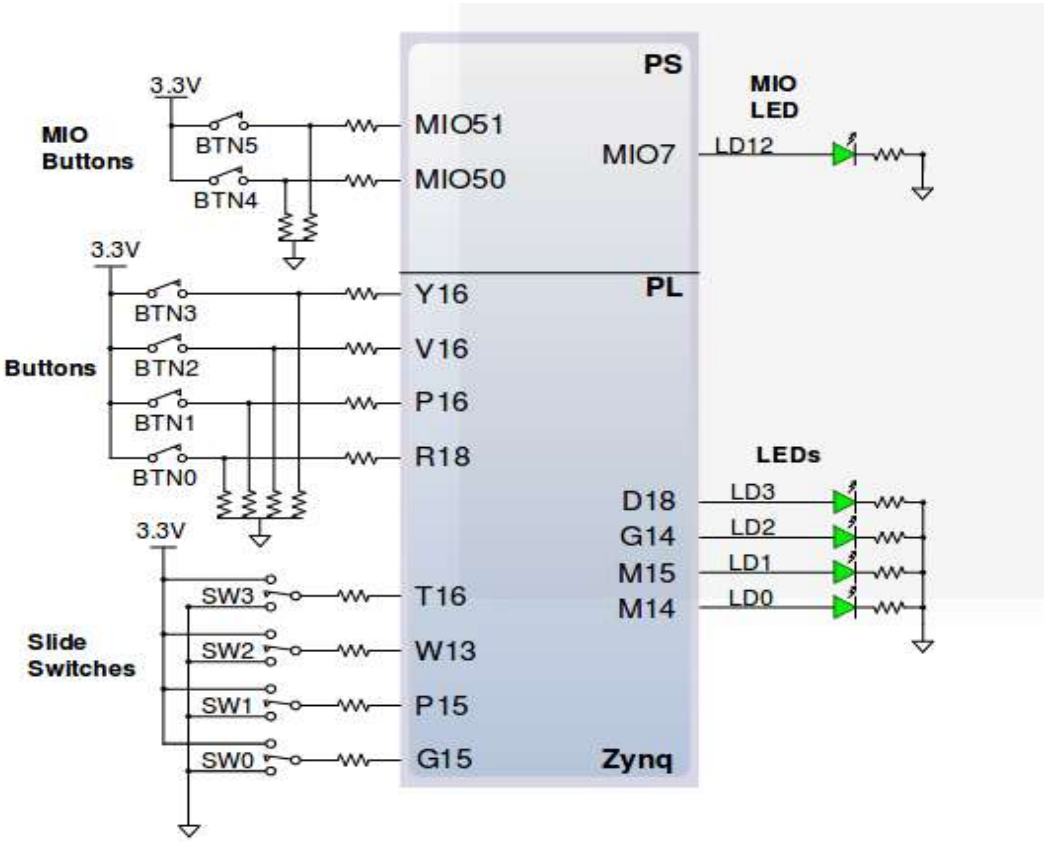


이 후에 밑에 있는 그림에서처럼 I/O Ports 를 누르면 I/O std 를 LVCMOS33*을 설정하고 밑에 선생님이 주신 pdf 로 가서 그림을 보면서 맞는 핀을 설정한다.

https://reference.digilentinc.com/_media/zybo:zybo_rm.pdf

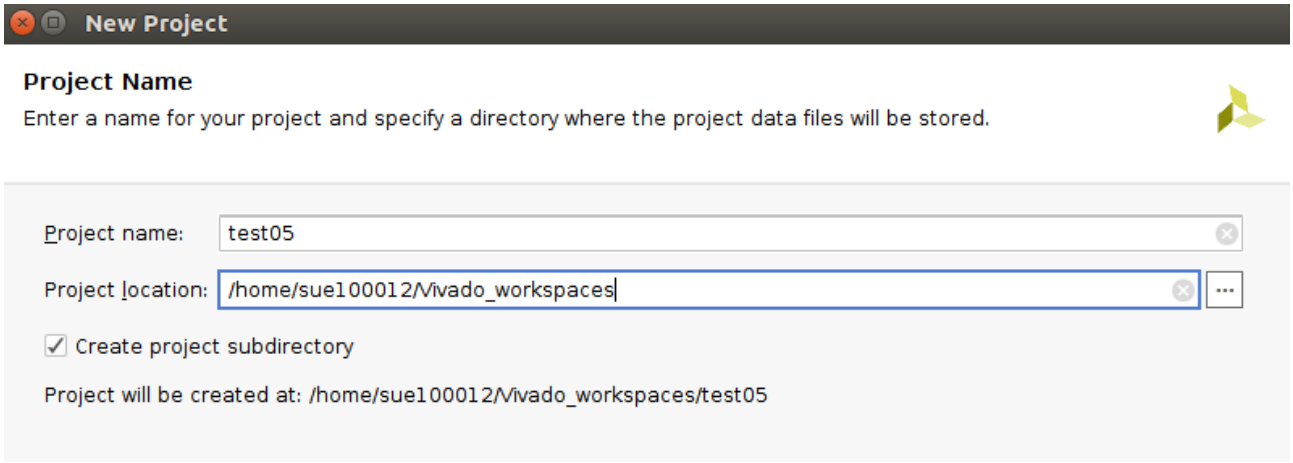






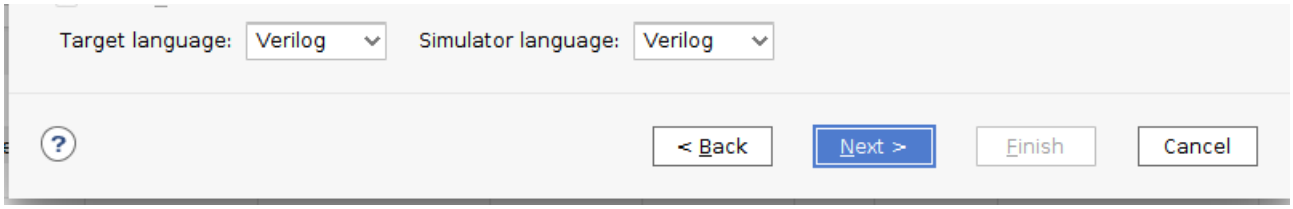
※ FPGA 로 LED 켜기

프로젝트를 생성한다.

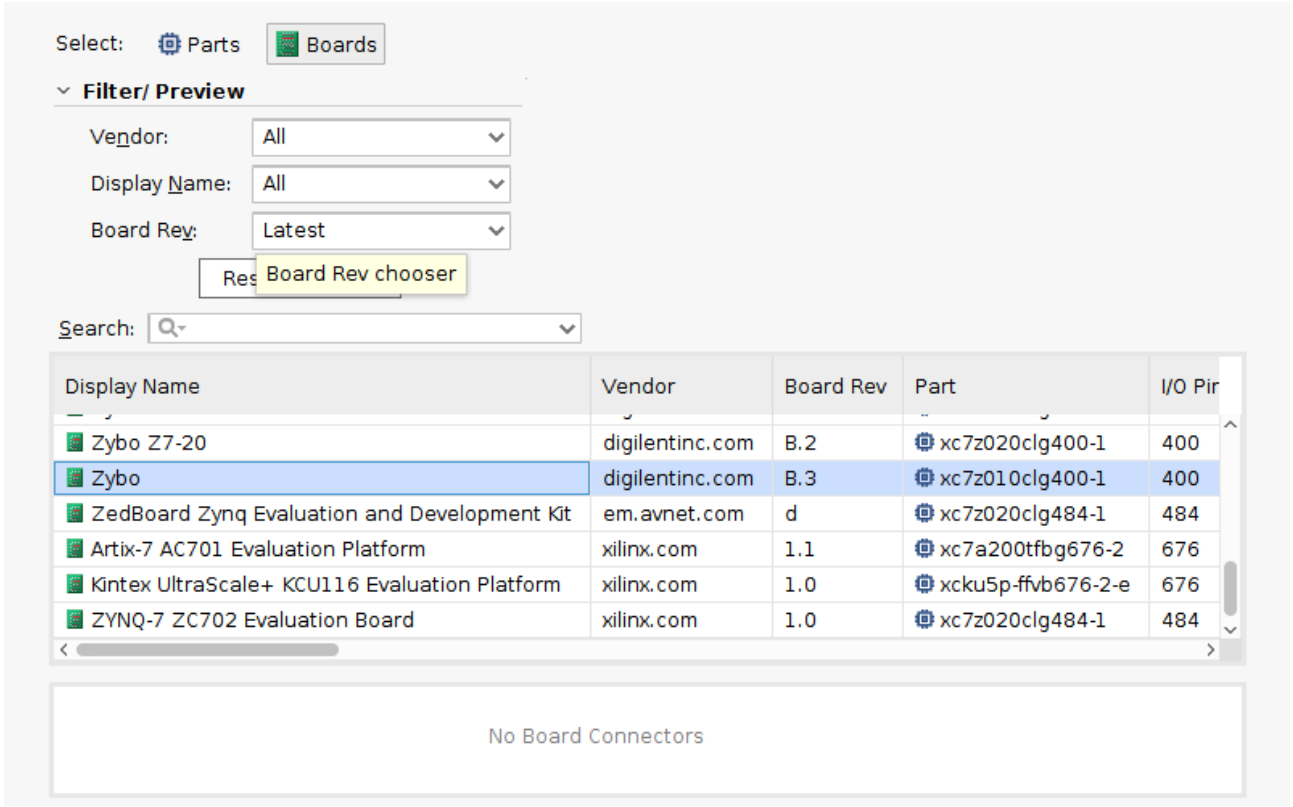


로 프로젝트 위치를 정해준다. 그 다음에 next 를 한다.

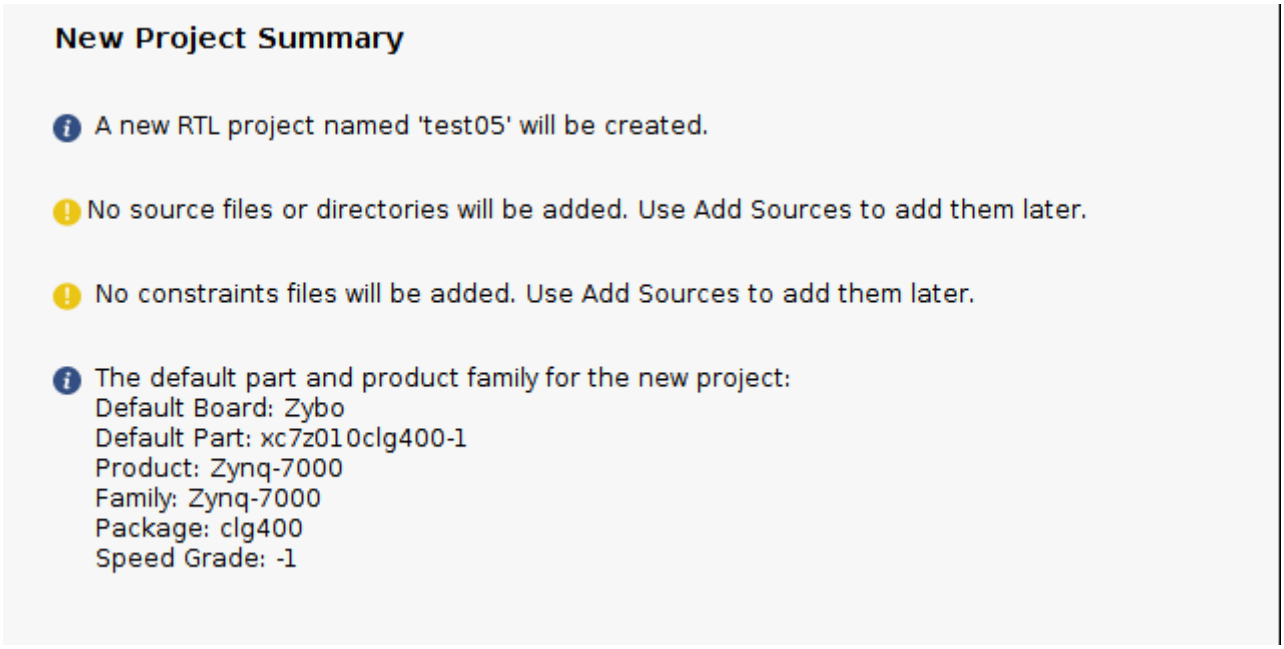
로 들어가서 위의 프로젝트를 선택해주고 next 를 한다.



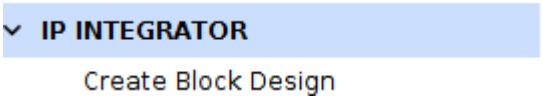
verliog 를 선택해준다. 그 다음 Add Constraints(optional)도 next 를 해준다.



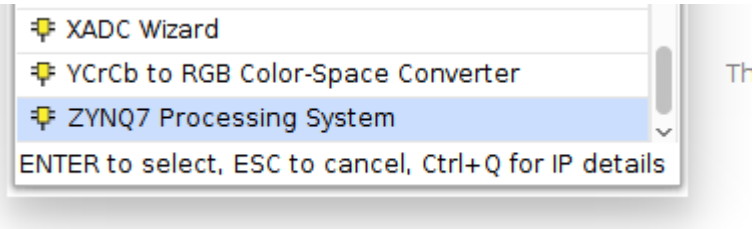
위의 보드를 선택해주고 next 를 한다. 그럼 프로젝트에 대한 정보 관련 창이 뜨고



해당 보드가 맞는지 확인한 후 마침하면 된다.

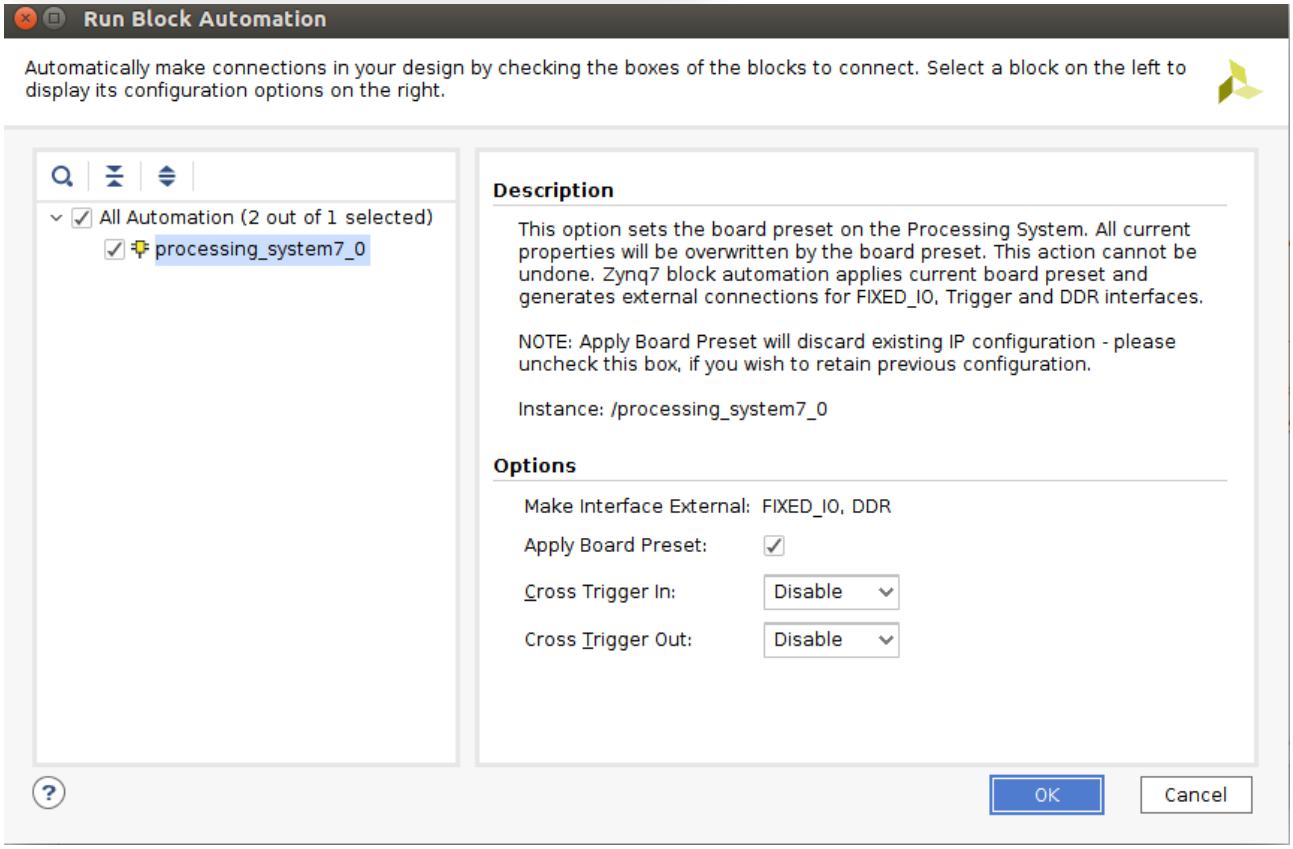


로 회로를 설계해준다. +버튼으로 추가를 해준다. 먼저



옆의 시스템을 추가해주고

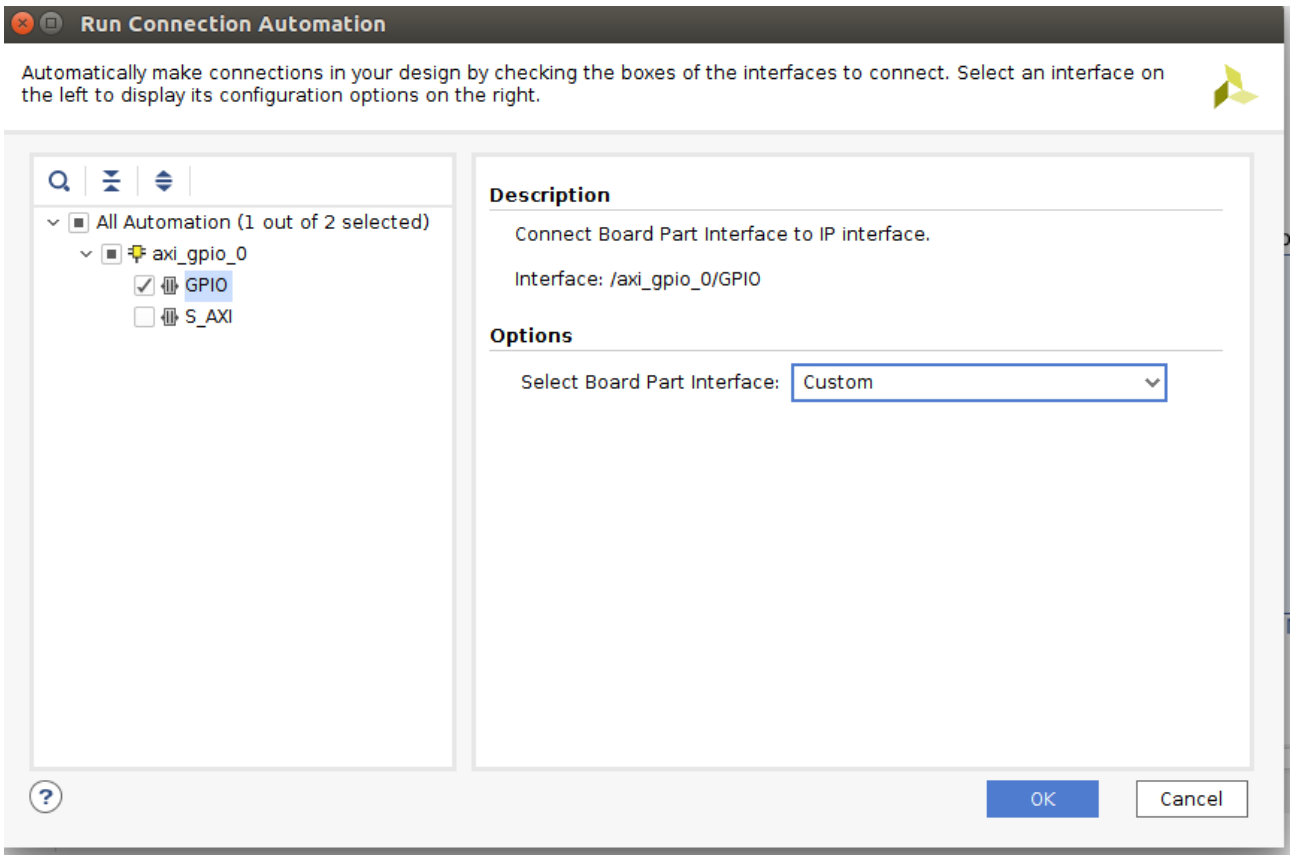
Run Connetion Automation 으로 밑의



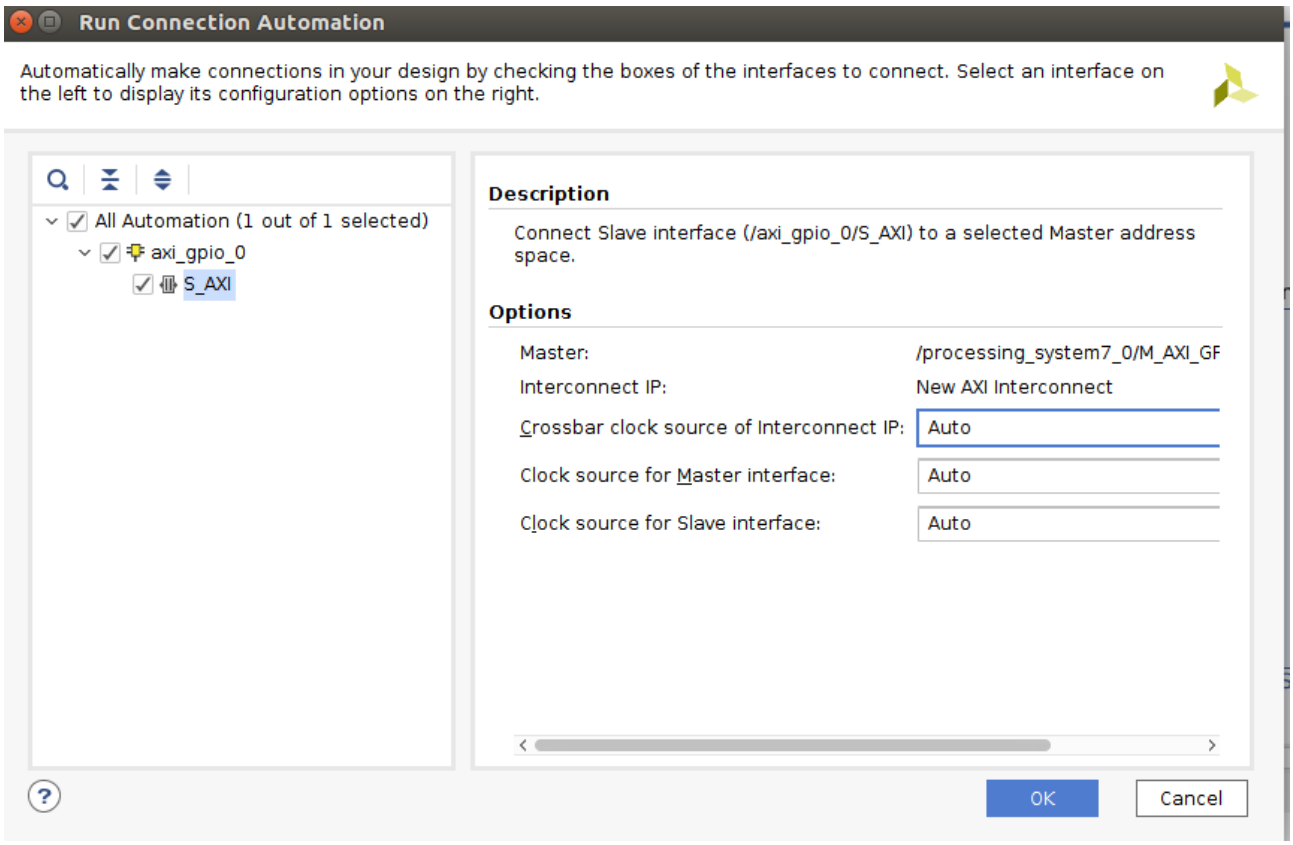
정보를 확인해준다.

그 후에 GPIO 를 추가해준다.

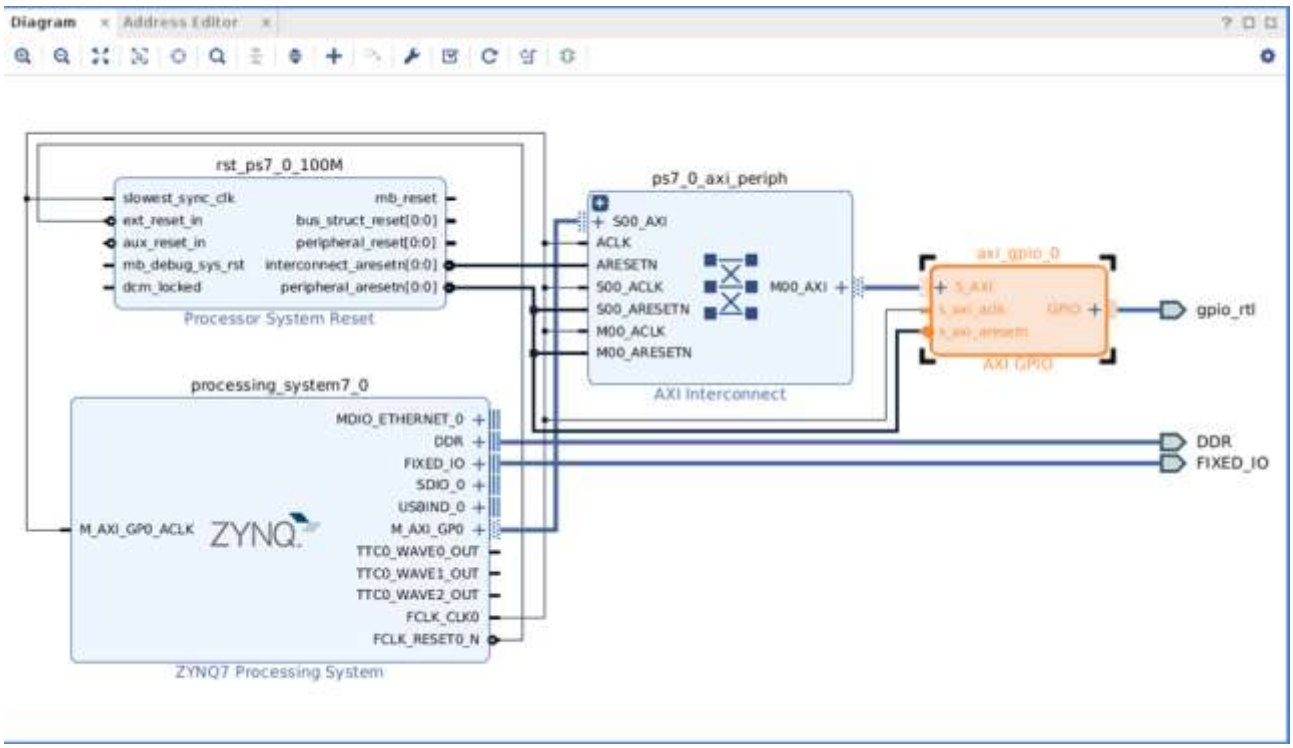
밑과 같이 설정해준다,



그 후 Run Connetion Automation 로 확인한다.

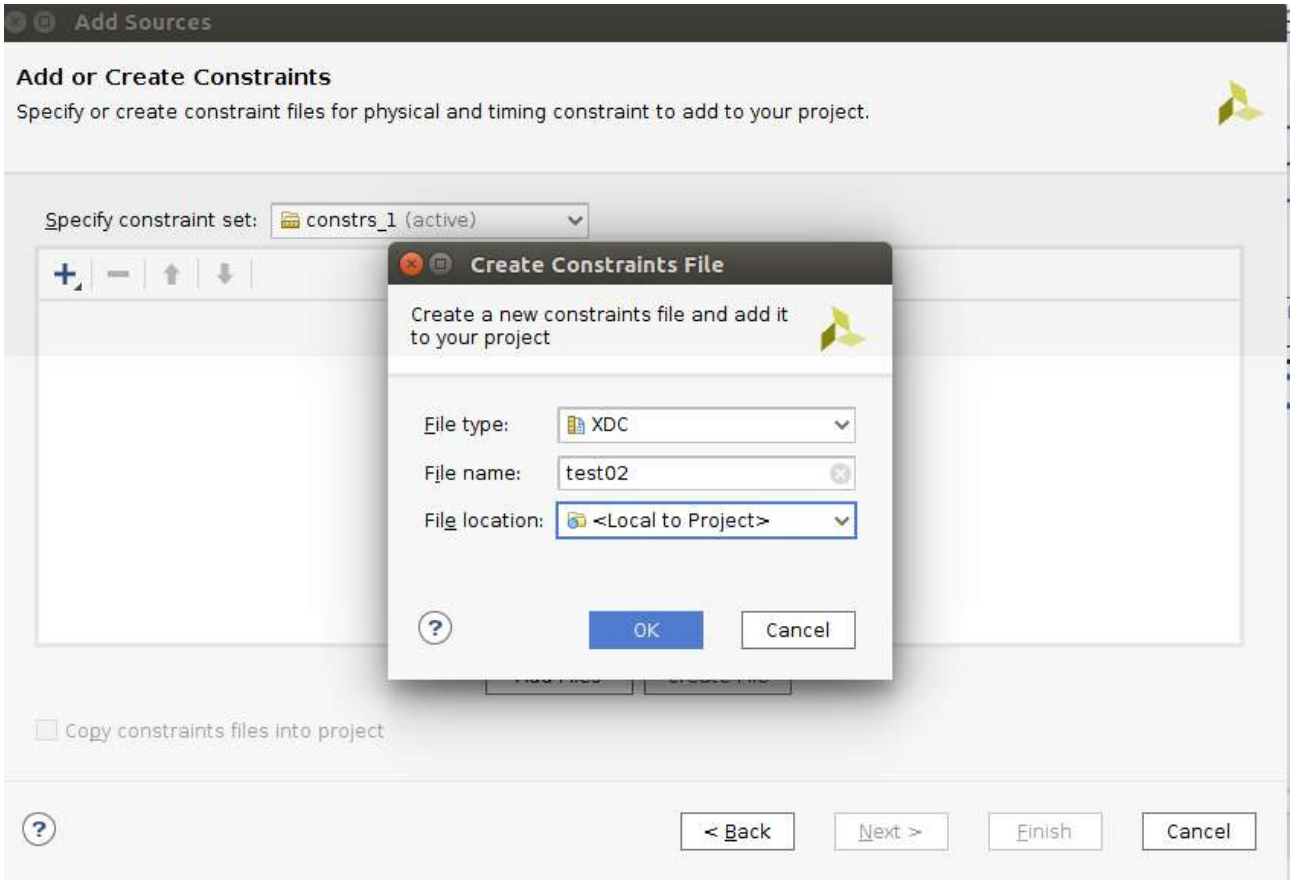


그 후 생성된 회로를 Regenerate Layout 을 눌러주면 다음과 같이 정렬 된다.



정렬 된 회로와 이제 어떤 핀을 쓸 지 정해주어야 한다.

Add sources 로 더해준다.

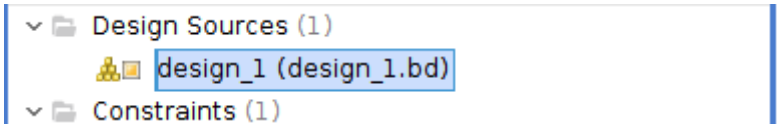


파일 이름을 설정해준다.

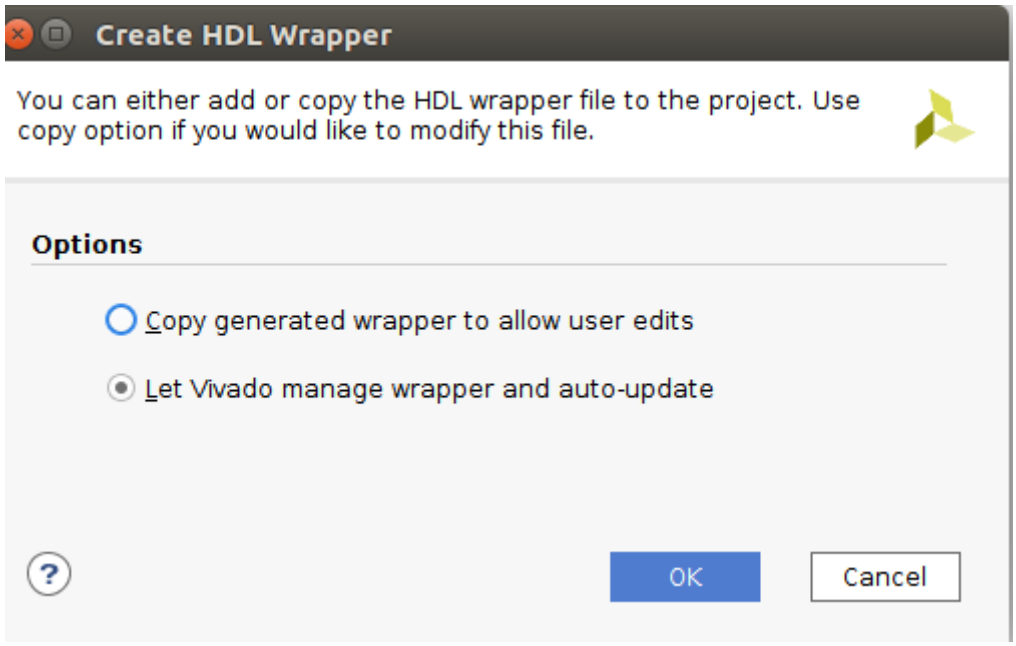
해당 소스로 가서 set_property -dict { PACKAGE_PIN V15 IOSTANDARD LVCMOS33 } [get_ports {gpio_rtl_tri_io}]; 를 써준다.

이는 V15 핀을 사용하겠다는 뜻이다.

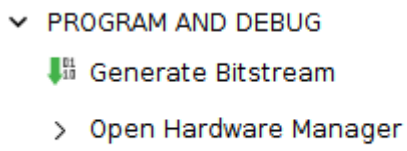
처럼 해주고 저장을 해준다.



해당디자인 소스에서 HDL wrapper 생성해준다.



다음과 같이 선택하고 생성해준다.



로 bitstream 을 해준다. 그 후 제대로 컴파일이 되면 export 하드웨어를 해준 후에 sdk 를 연결해서 작업해주면 된다. SDK 에서 제대로 프로젝트를 생성하면서 제대로 연결되었는지 확인해 준 후, 다음의 소스코드를 작성해주고 컴파일 한 후, 보드로 구워주면 된다.

```
#include <stdio.h>
#include <xgpio.h>
#include "xparameters.h"
#include "sleep.h"

int main(void)
{
    XGpio in, out;
```

```
    XGpio_Initialize(&out, XPAR_AXI_GPIO_0_DEVICE_ID);

    XGpio_SetDataDirection(&out, 1, 0x0);

    while(1)
     {
     Xil_Out32(0x41200000, 0x0);
     sleep(1);
     Xil_Out32(0x41200000, 0xFFFFFFFF);
     sleep(1);
     }

    return 0;
}
```

로 결과는 LED 가 깜박 거린다.