

Creating a Custom IP core using the IP Integrator

Innova Lee(이상훈)
gcccompil3r@gmail.com



Prerequisites

- Zybo Getting Started Guide 완료: SDK 설치

<https://reference.digilentinc.com/learn/programmable-logic/tutorials/zybo-getting-started-with-zynq/start?redirect=1>

Tutorial

이 데모에서 Zynq Processor 의 Processing System 을 사용하여 보드상의 LED 를 조작하기 위한 기본 PWM 제어기를 구축하는 방법을 보여준다. IP Graphic Interface 에서 PWM 윈도우 크기를 변경한 다음 Processor 용으로 작성된 Duty Cycle 을 제어할 수 있다.

1. Open Vivado and create a new project

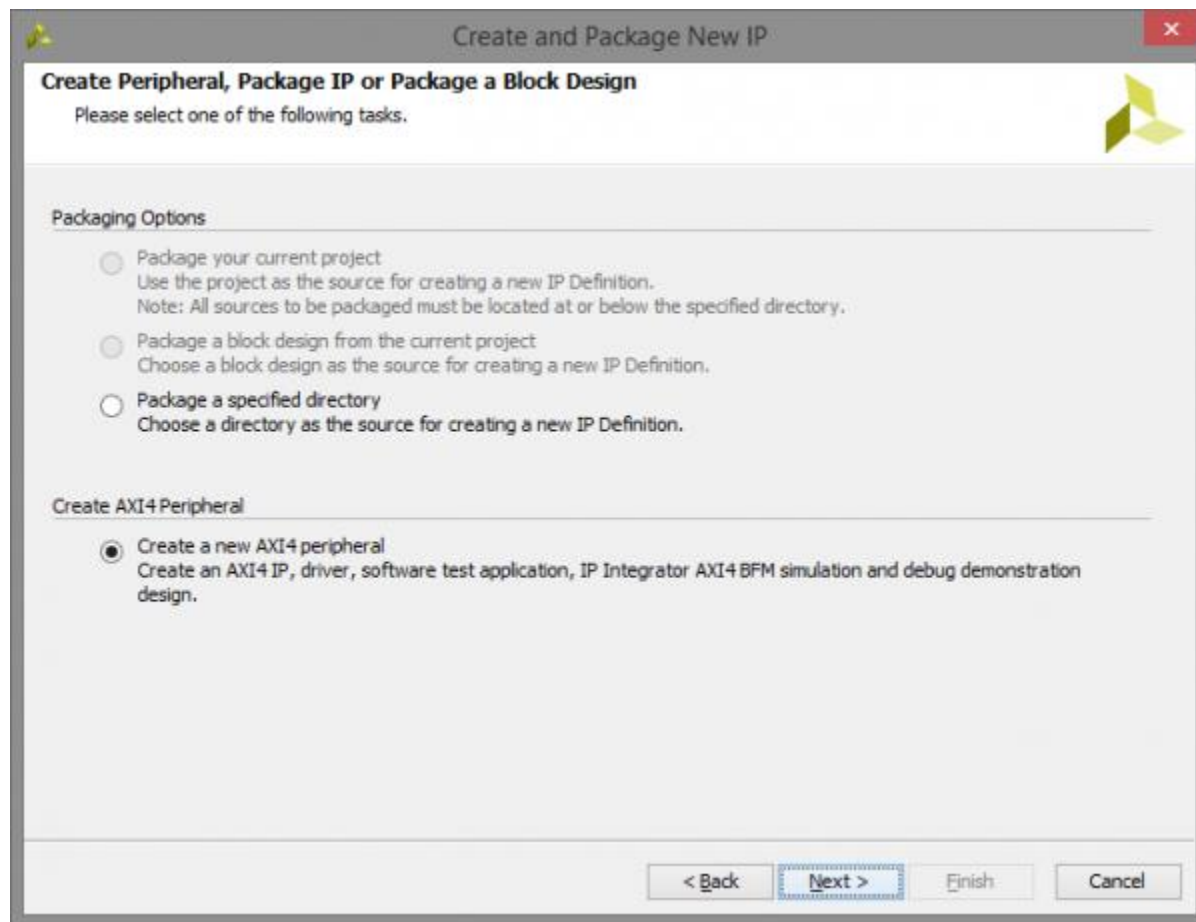
Zybo Getting Started Guide 에 표시된대로 New Project 를 연다.

Tools -> Create and package IP 로 이동한다.

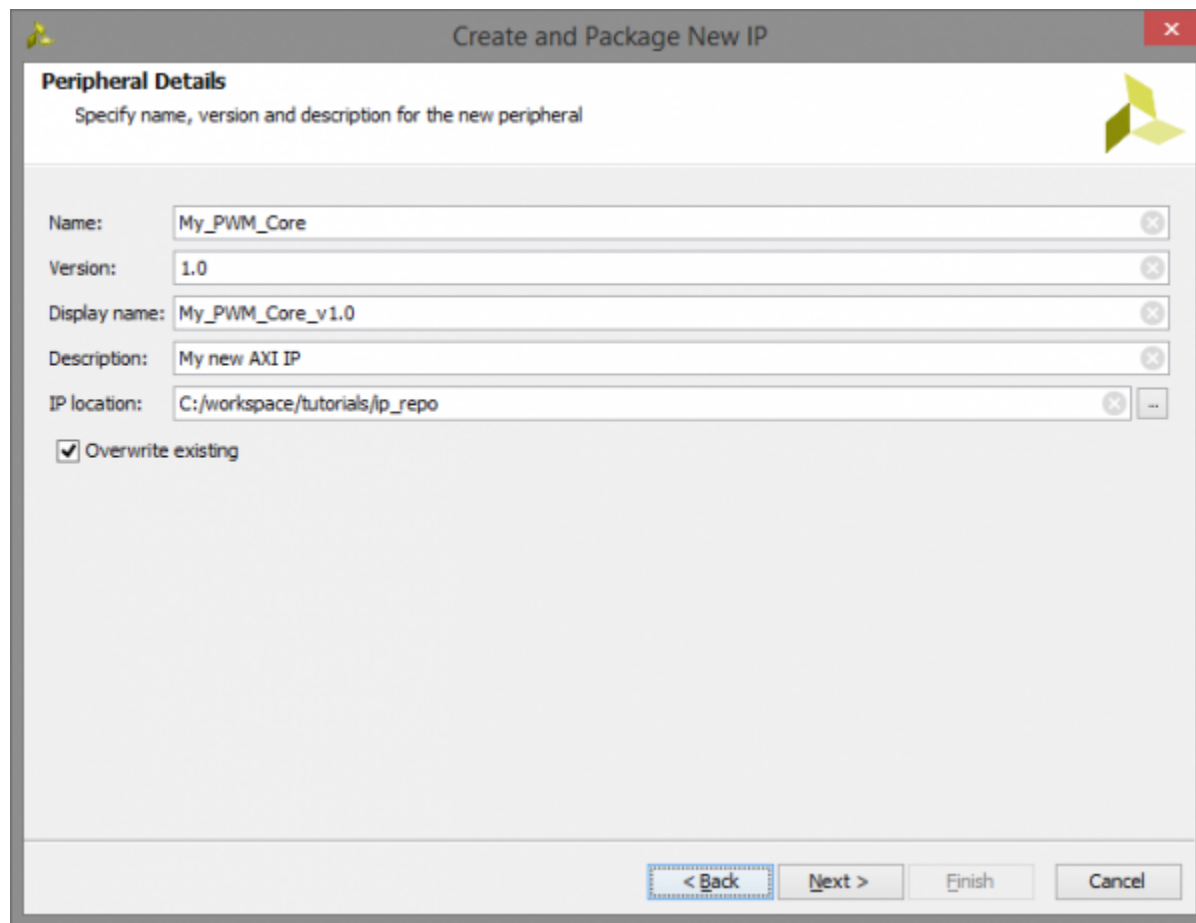


2. Create your custom IP project

2.1) 새로운 AXI4 Peripheral 을 선택하고 Next 를 클릭한다.



2.2) 이름 필드에 "My_PWM_Core" 를 입력하고 Next 를 클릭한다.

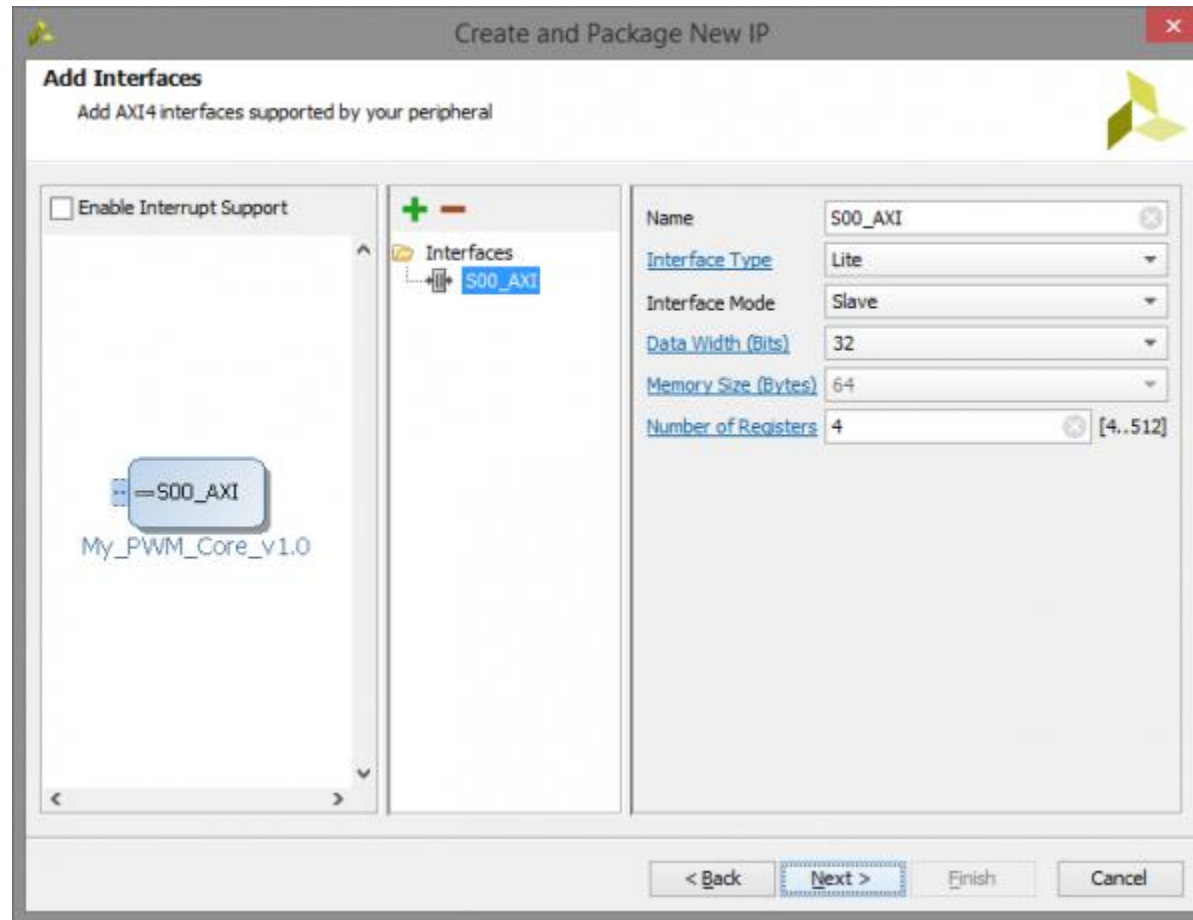


The screenshot shows a dialog box titled "Create and Package New IP" with a close button (X) in the top right corner. The dialog has a tab labeled "Peripheral Details" and a subtitle "Specify name, version and description for the new peripheral". The dialog contains several input fields and a checkbox:

- Name:** My_PWM_Core
- Version:** 1.0
- Display name:** My_PWM_Core_v1.0
- Description:** My new AXI IP
- IP location:** C:/workspace/tutorials/ip_repo
- ☒ Overwrite existing

At the bottom of the dialog, there are four buttons: "< Back" (highlighted with a dashed border), "Next >", "Finish", and "Cancel".

2.3) AXI 인터페이스를 변경할 필요가 없으므로 Next 를 클릭한다.



2.4) Edit IP 를 선택하고 Finish 를 클릭한다.



3. Designing the IP core

- 3.1) Vivado 의 새로운 Instance 가 새로운 IP 코어를 열어준다.
최상위 레벨 파일 My_PWM_Core_v1_0 을 확장한다.
My_PWM_Core_v1_0_S00_AXI 를 더블 클릭하여 편집기에서 연다.

4. Modifying My_PWM_Core_v1_0_S00_AXI

- 4.1) 사용자가 최상위 레벨에서 PWM 창의 최대 값을 변경할 수 있게 하려면 아래 부분에 내용을 추가한다.
(코드 상에서 변경해야 함)

```
// Users to add parameters here
```

```
// User parameters ends
```

```
// Do not modify the parameters beyond this line
```

아래와 같이 변경한다.

```
// Users to add parameters here
```

```
parameter integer PWM_COUNTER_MAX = 1024,
```

```
// User parameters ends
```

```
// Do not modify the parameters beyond this line
```


4.2) PWM 신호에 사용자 정의 IP 코어 외부 포트를 제공하려면 아래와 같이 변경한다:

```
// Users to add ports here
```

```
// User ports ends
```

```
// Do not modify the ports beyond this line
```

이 부분을 아래와 같이 변경한다.

```
// Users to add ports here
```

```
output wire PWM0,
```

```
output wire PWM1,
```

```
output wire PWM2,
```

```
output wire PWM3,
```

```
// User ports ends
```

```
// Do not modify the ports beyond this line
```

4.3) 아래 변경은 16 비트 폭이며 최대(2^{16}) - 1 의 카운터를 생성하며 대부분의 응용 프로그램에서 사용하기에 충분하다.
파일의 맨 아래로 스크롤하여 아래를 변경하라:

```
// Add user logic here
// User logic ends
```

이 부분을 아래와 같이 변경한다.

```
// Add user logic here
reg [15:0] counter = 0;
```

```
//simple counter
always @(posedge S_AXI_ACLK) begin
    if(counter < PWM_COUNTER_MAX-1)
        counter <= counter + 1;
    else
        counter <= 0;
end
```

```
//comparator statements that drive the PWM signal
assign PWM0 = slv_reg0 < counter ? 1'b0 : 1'b1;
assign PWM1 = slv_reg1 < counter ? 1'b0 : 1'b1;
assign PWM2 = slv_reg2 < counter ? 1'b0 : 1'b1;
assign PWM3 = slv_reg3 < counter ? 1'b0 : 1'b1;
// User logic ends
```

전반적으로 이 모듈은 프로세서의 슬레이브 레지스터에 데이터를 쓴다.

간단한 카운터는 최대 값까지 카운팅하고 영원히 리셋된다.

그런 다음 각 비교기 명령문은 현재 카운터 값이 슬레이브 레지스터에 저장된 값 보다 큰지 확인한 다음 비교 값이 현재 카운터보다 작으면 PWM 을 HIGH 로 설정한다.

이를 통해 달성한 것:

- PWM_COUNTER_MAX 로 PWM 창 크기를 매개 변수화
- 상위 레벨 파일이 PWM 신호를 얻을 수 있도록 포트를 추가했다.
- 0 에서 PWM_COUNTER_MAX - 1 까지 카운트하는 간단한 카운터 추가
- PWM 신호를 생성하는 4 개의 비동기 비교기 신호 추가

5. Modifying My_PWM_Core_v1_0

5.1) My_PWM_Core_v1_0 을 더블 클릭하여 편집기에서 연다.

다음 수정 사항을 PWM 신호 포트와 매개 변수를 최상위 HDL 파일에 추가한다.
이렇게 하면 GUI 가 IP 코어를 변경, 연결 및 수정할 수 있다.

5.2) 아래 부분을 변경한다:

```
module My_PWM_Core_v1_0 #  
(  
    // Users to add parameters here  
  
    // User parameters ends  
    // Do not modify the parameters beyond this line
```

아래와 같이 변경한다.

```
module My_PWM_Core_v1_0 #  
(  
    // Users to add parameters here  
    parameter integer PWM_COUNTER_MAX = 128,  
  
    // User parameters ends  
    // Do not modify the parameters beyond this line
```

5.2) 그리고 아래 부분을 수정한다:

```
module My_PWM_Core_v1_0 #  
(  
    // Users to add parameters here  
  
    // User parameters ends  
    // Do not modify the parameters beyond this line
```

아래와 같이 수정한다:

```
module My_PWM_Core_v1_0 #  
(  
    // Users to add parameters here  
    parameter integer PWM_COUNTER_MAX = 128,  
  
    // User parameters ends  
    // Do not modify the parameters beyond this line
```

5.3) 이 부분도 수정한다:

```
// Users to add ports here  
  
// User ports ends
```

아래와 같이 수정한다:

```
// Users to add ports here  
output wire PWM0,  
output wire PWM1,  
output wire PWM2,  
output wire PWM3,  
// User ports ends
```

5.4) 여기도 수정한다:

```
// Instantiation of Axi Bus Interface S00_AXI
My_PWM_Core_v1_0_S00_AXI # (
    .C_S_AXI_DATA_WIDTH(C_S00_AXI_DATA_WIDTH),
    .C_S_AXI_ADDR_WIDTH(C_S00_AXI_ADDR_WIDTH)
) My_PWM_Core_v1_0_S00_AXI_inst (
    .S_AXI_ACLK(s00_axi_aclk),
    .S_AXI_ARESETN(s00_axi_aresetn),
    .S_AXI_AWADDR(s00_axi_awaddr),
    .S_AXI_AWPROT(s00_axi_awprot),
    .S_AXI_AWVALID(s00_axi_awvalid),
    .S_AXI_AWREADY(s00_axi_awready),
    .S_AXI_WDATA(s00_axi_wdata),
    .S_AXI_WSTRB(s00_axi_wstrb),
    .S_AXI_WVALID(s00_axi_wvalid),
    .S_AXI_WREADY(s00_axi_wready),
    .S_AXI_BRESP(s00_axi_bresp),
    .S_AXI_BVALID(s00_axi_bvalid),
    .S_AXI_BREADY(s00_axi_bready),
    .S_AXI_ARADDR(s00_axi_araddr),
    .S_AXI_ARPROT(s00_axi_arprot),
    .S_AXI_ARVALID(s00_axi_arvalid),
    .S_AXI_ARREADY(s00_axi_arready),
    .S_AXI_RDATA(s00_axi_rdata),
    .S_AXI_RRESP(s00_axi_rresp),
    .S_AXI_RVALID(s00_axi_rvalid),
    .S_AXI_RREADY(s00_axi_rready)
);
```

해당 부분을 아래와 같이 변경한다:

```
// Instantiation of Axi Bus Interface S00_AXI
My_PWM_Core_v1_0_S00_AXI # (
    .C_S_AXI_DATA_WIDTH(C_S00_AXI_DATA_WIDTH),
    .C_S_AXI_ADDR_WIDTH(C_S00_AXI_ADDR_WIDTH),
    .PWM_COUNTER_MAX(PWM_COUNTER_MAX)
) My_PWM_Core_v1_0_S00_AXI_inst (
    .PWM0(PWM0),
    .PWM1(PWM1),
    .PWM2(PWM2),
    .PWM3(PWM3),
    .S_AXI_ACLK(s00_axi_aclk),
    .S_AXI_ARESETN(s00_axi_aresetn),
    .S_AXI_AWADDR(s00_axi_awaddr),
    .S_AXI_AWPROT(s00_axi_awprot),
    .S_AXI_AWVALID(s00_axi_awvalid),
    .S_AXI_AWREADY(s00_axi_awready),
    .S_AXI_WDATA(s00_axi_wdata),
    .S_AXI_WSTRB(s00_axi_wstrb),
    .S_AXI_WVALID(s00_axi_wvalid),
    .S_AXI_WREADY(s00_axi_wready),
    .S_AXI_BRESP(s00_axi_bresp),
    .S_AXI_BVALID(s00_axi_bvalid),
    .S_AXI_BREADY(s00_axi_bready),
    .S_AXI_ARADDR(s00_axi_araddr),
    .S_AXI_ARPROT(s00_axi_arprot),
    .S_AXI_ARVALID(s00_axi_arvalid),
    .S_AXI_ARREADY(s00_axi_arready),
    .S_AXI_RDATA(s00_axi_rdata),
    .S_AXI_RRESP(s00_axi_rresp),
    .S_AXI_RVALID(s00_axi_rvalid),
    .S_AXI_RREADY(s00_axi_rready)
);
```

6. Packaging the IP core

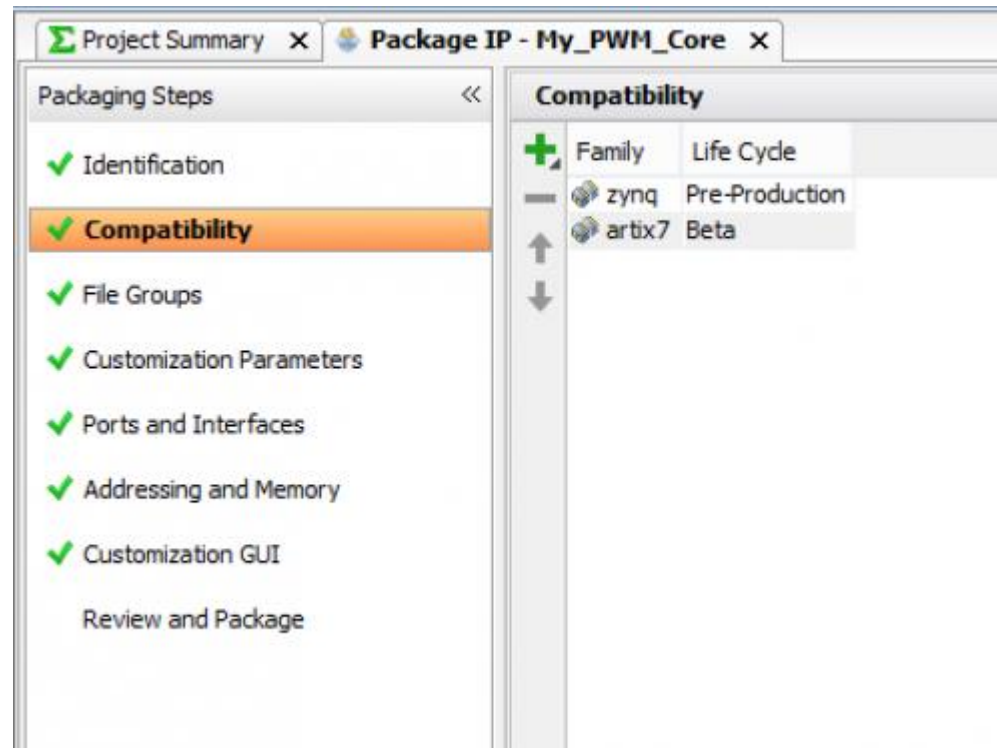
이제 코어를 작성 했으므로 완벽한 IP 패키지를 만들기 위해 HDL 을 패키지화해야 한다.

6.1) Flow Navigator 에서 Package IP 를 클릭하면 Package IP 탭이 보일 것이다.

호환성을 선택하고 "Artix7" 및 "Zynq" 가 있는지 확인한다.

해당 항목이 없으면 더하기 단추를 클릭하여 추가 할 수 있다.

Life Cycle 은 이 시점에서 중요하지 않다.



6.2) Customization Parameters 를 선택하고 Customization Parameters Wizard 에서 Merge Changes 를 선택한다.
Hidden Parameter 폴더 아래의 최상위 파일에 PWM_COUNTER_MAX 매개 변수가 추가된다.

Project Summary X Package IP - My_PWM_Core X

Packaging Steps<<

✓ Identification

✓ Compatibility

✓ File Groups

✓ Customization Parameters

✓ Ports and Interfaces

✓ Addressing and Memory

✓ Customization GUI

Review and Package

Customization Parameters

Name	Description	Display Name	Value	Value Bit String Length	Value Format	Value Source
Customization Parameters						
C_S00_AXI_DATA_WIDTH	Width of S_AXI data bus	C S00 AXI DATA WIDTH	32	0	long	default
C_S00_AXI_ADDR_WIDTH	Width of S_AXI address bus	C S00 AXI ADDR WIDTH	4	0	long	default
C_S00_AXI_BASEADDR		C S00 AXI BASEADDR	0xFFFFFFFF	32	bitString	default
C_S00_AXI_HIGHADDR		C S00 AXI HIGHADDR	0x00000000	32	bitString	default
PWM_COUNTER_MAX		Pwm Counter Max	128	0	long	default

6.3) Customization GUI 를 선택한다.

이것이 우리의 그래픽 인터페이스를 바꾸는 것이다.

한 가지 문제는 매개 변수에 대한 창이 없다.

PWM Counter Max 를 마우스로 우클릭하고 'Edit Parameter ...' 을 선택하라.

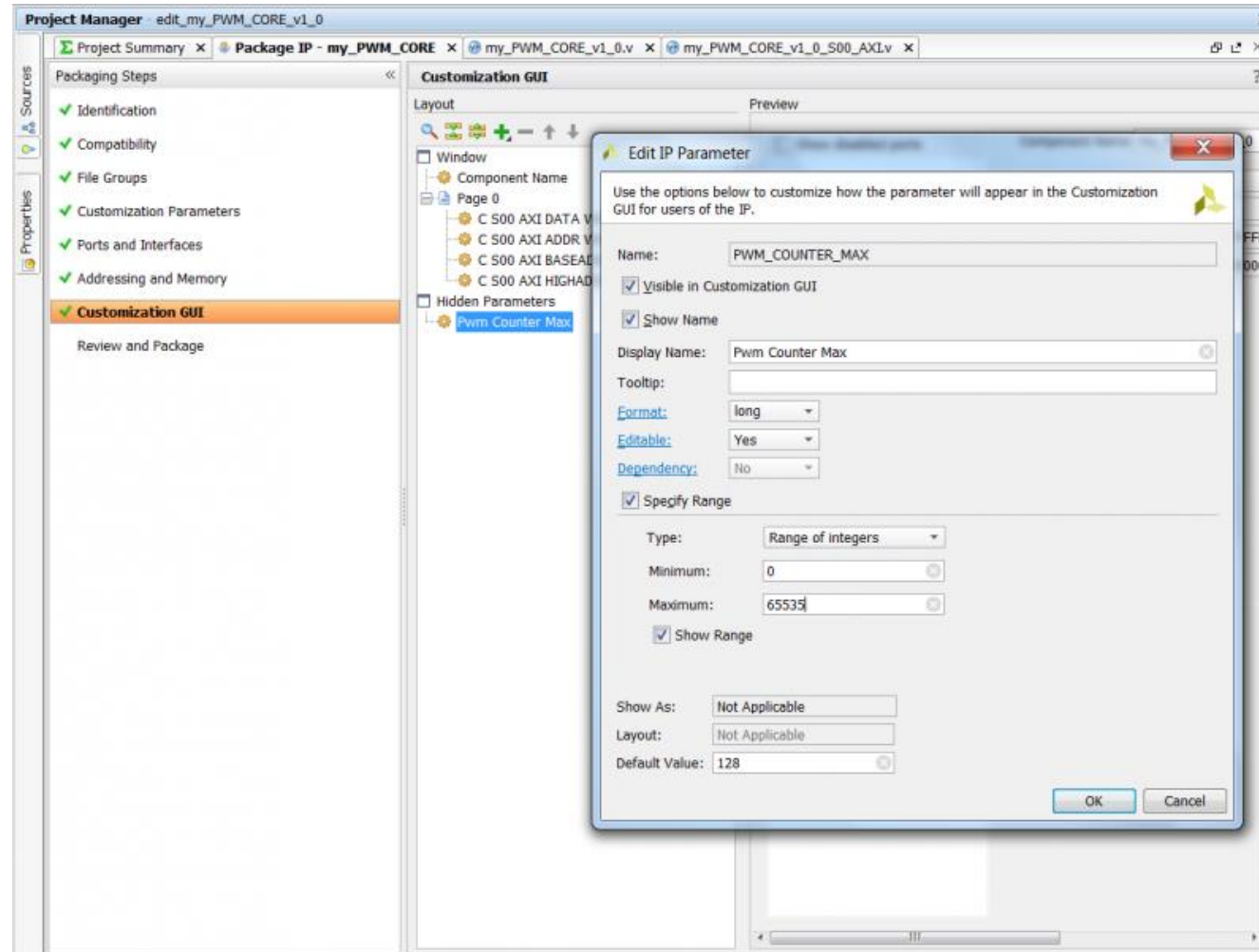
Visible in Customization GUI 상자를 체크한다.

Specify Range 박스를 체크한다.

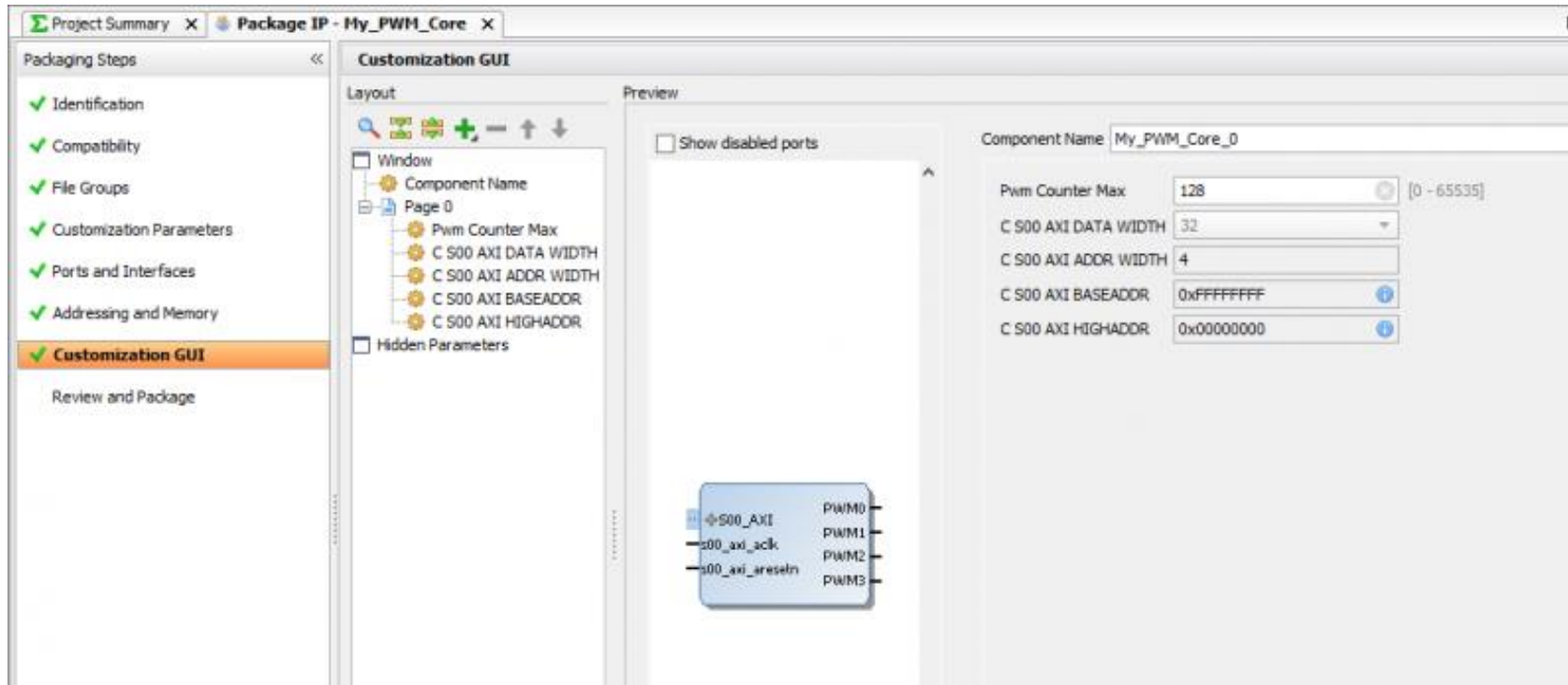
Type Dropdown 메뉴에서 정수 범위를 선택하라.

우리는 $(2^{16}) - 1 = 65535$ 의 최대 값과 최소 값 0 이므로 이것은 유용하지 않다.

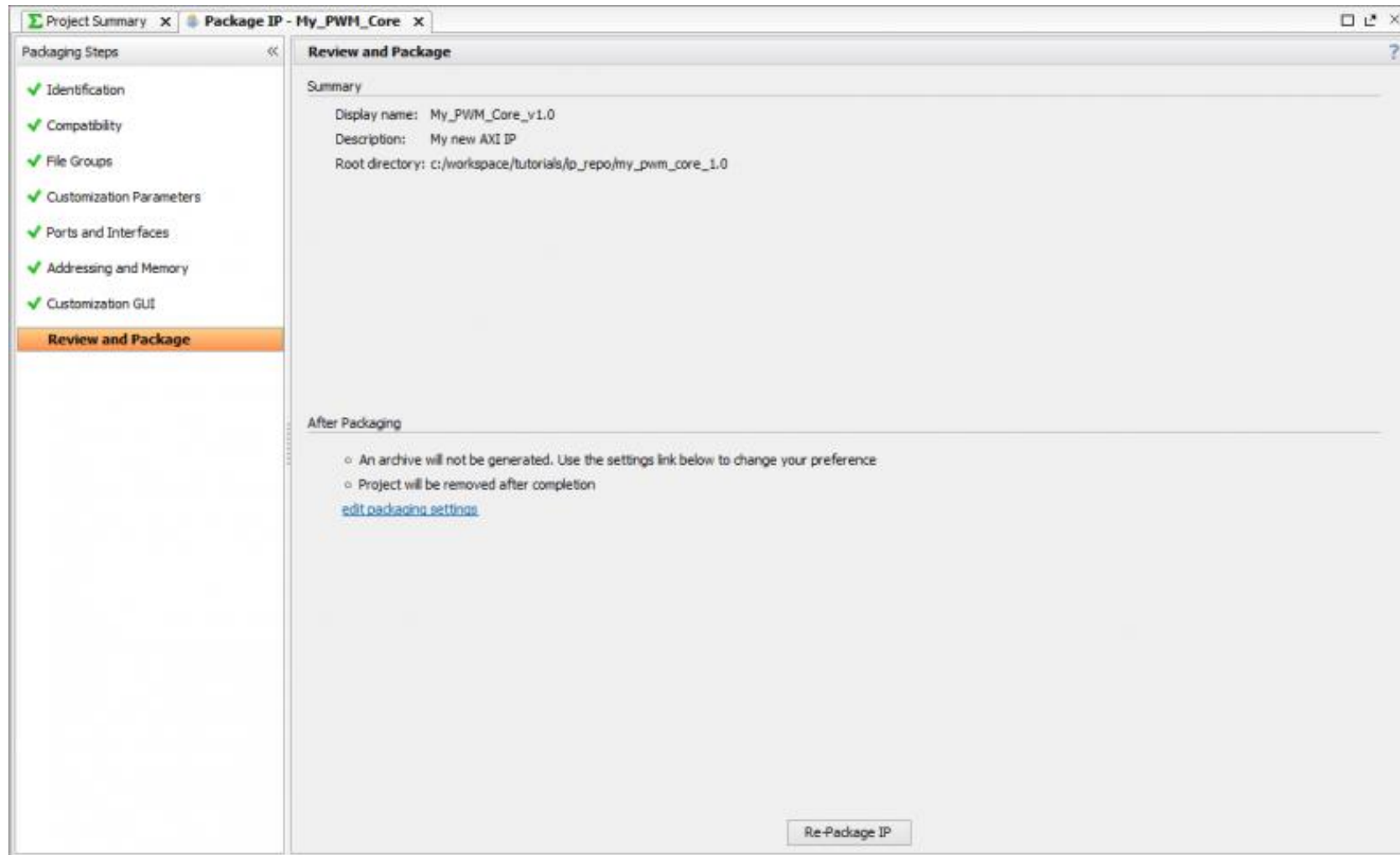
OK 를 클릭한다.



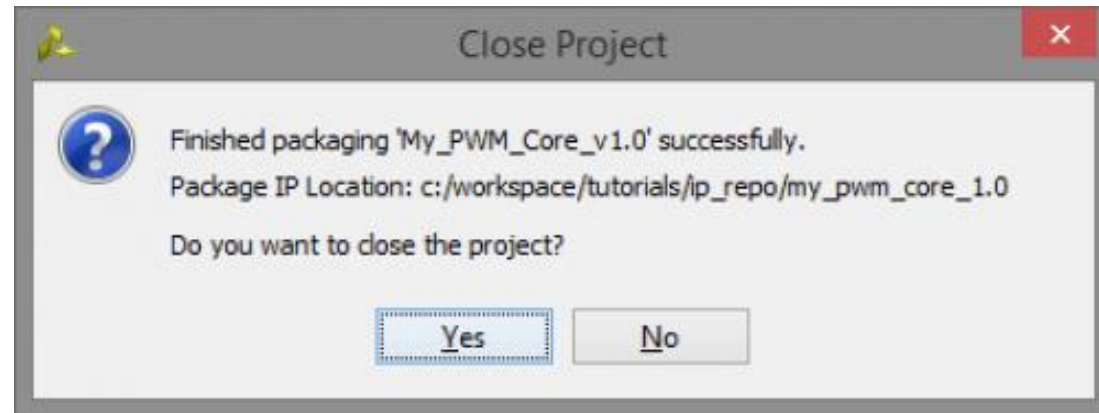
6.4) PWM Counter Max 를 Page 0 으로 드래그하여 메인 페이지에 가져온다.



6.5) 이제 코어가 완성되어야 하므로 Review and Package 를 선택하고 Re-package IP 버튼을 클릭하라.



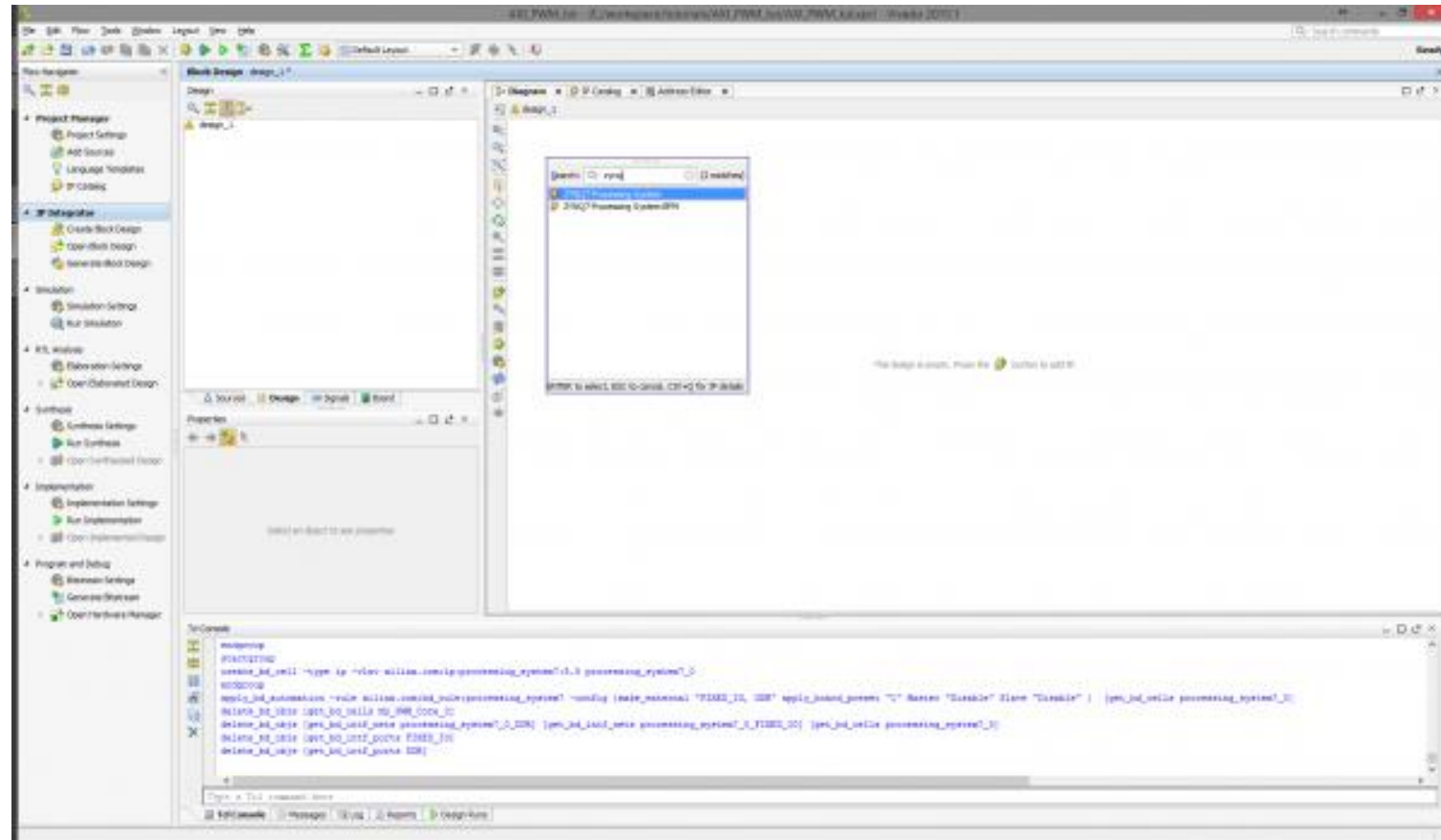
6.6) 프로젝트를 닫을 것인지 묻는 팝업이 나타나면 Yes 를 선택한다.



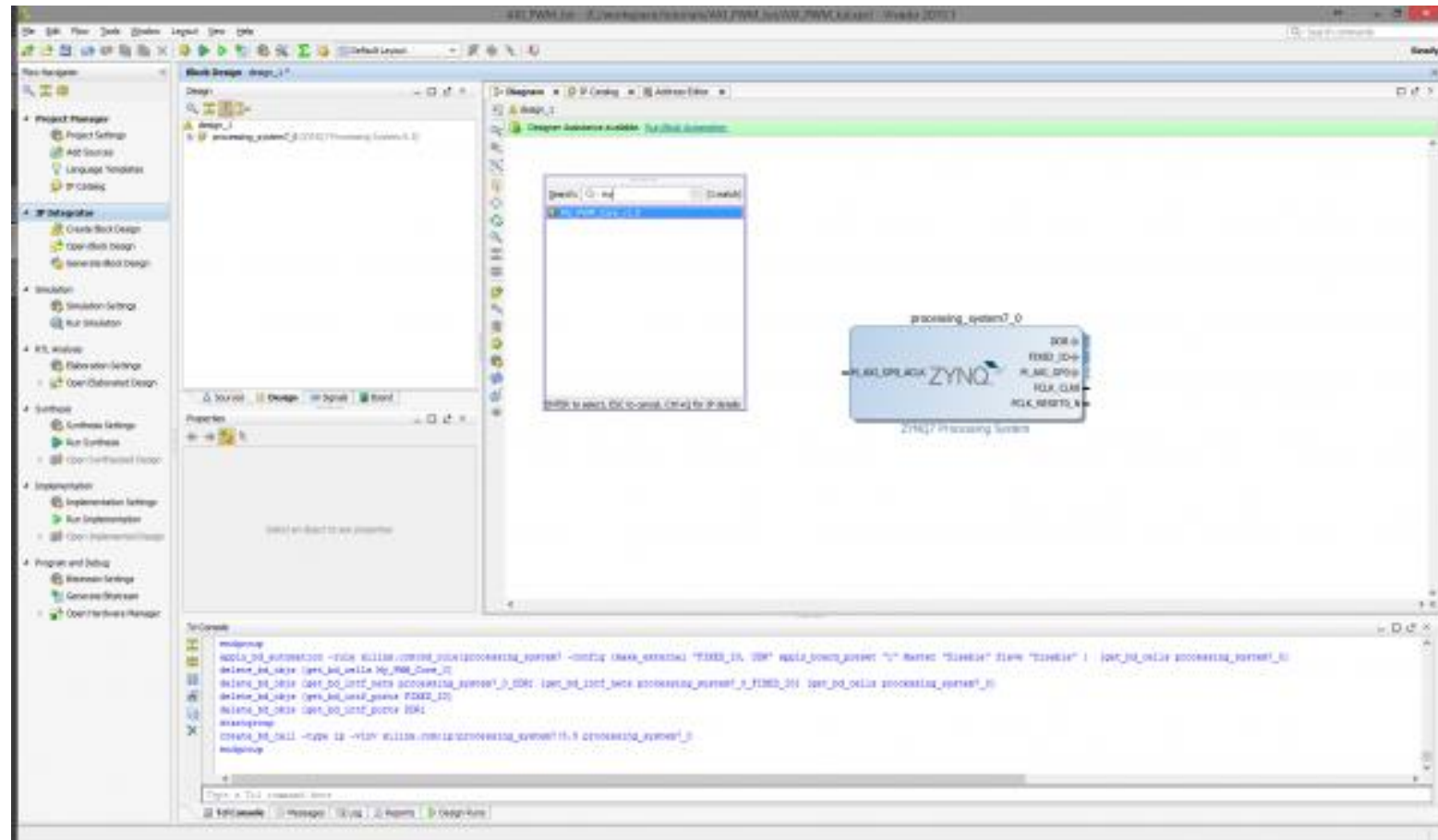
7. Create Zynq design

7.1) 원본 창의 프로젝트 관리자 페이지에서 Create Block Design 을 클릭한다.
이렇게 하면 Block Design 이 프로젝트에 추가된다.

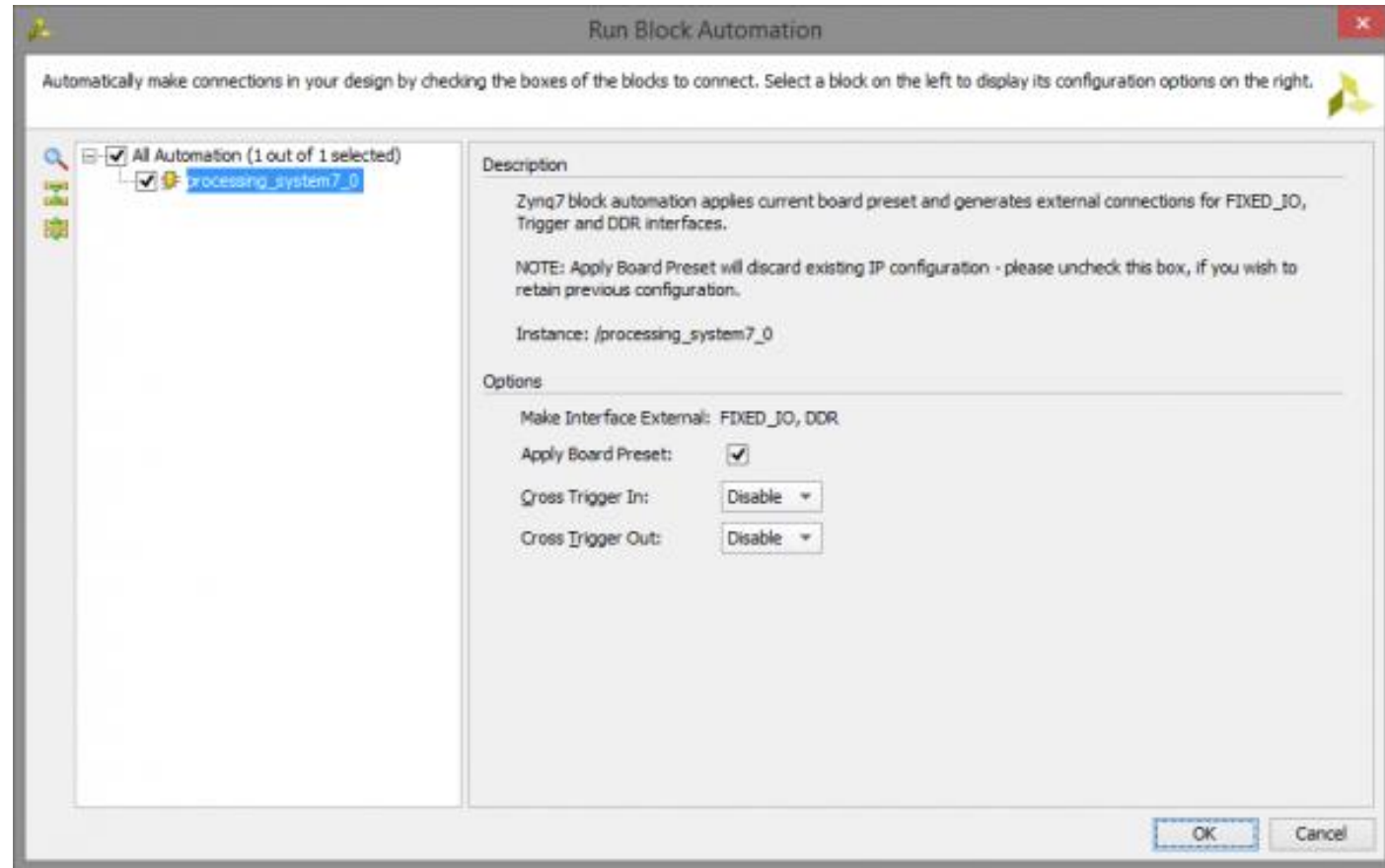
7.2) Add IP 버튼을 사용하여 Zynq Processing System 을 추가한다.



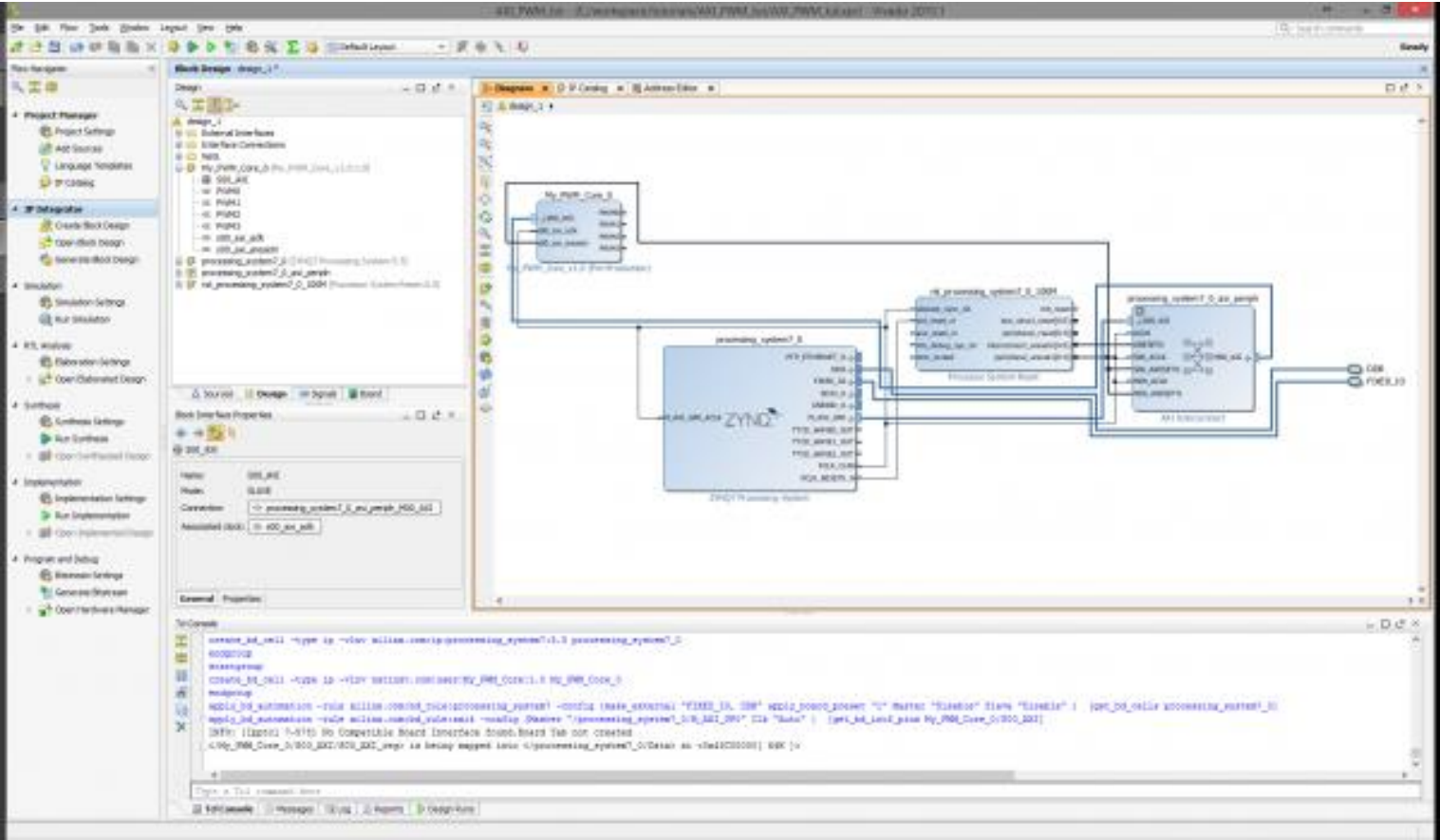
7.3) Add IP 버튼을 다시 사용하여 PWM 코어를 추가한다.



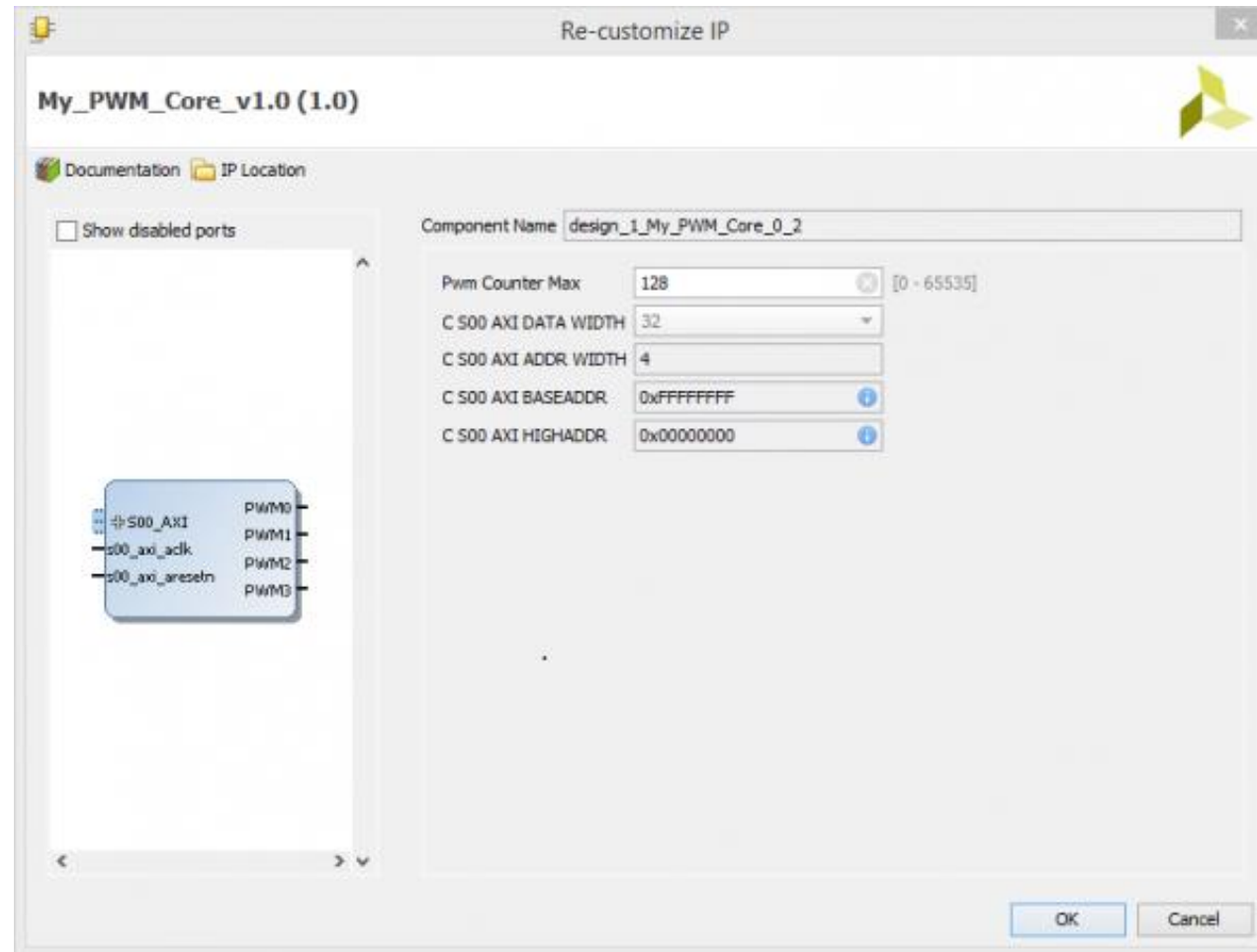
7.4) Block Automation 을 실행한다.



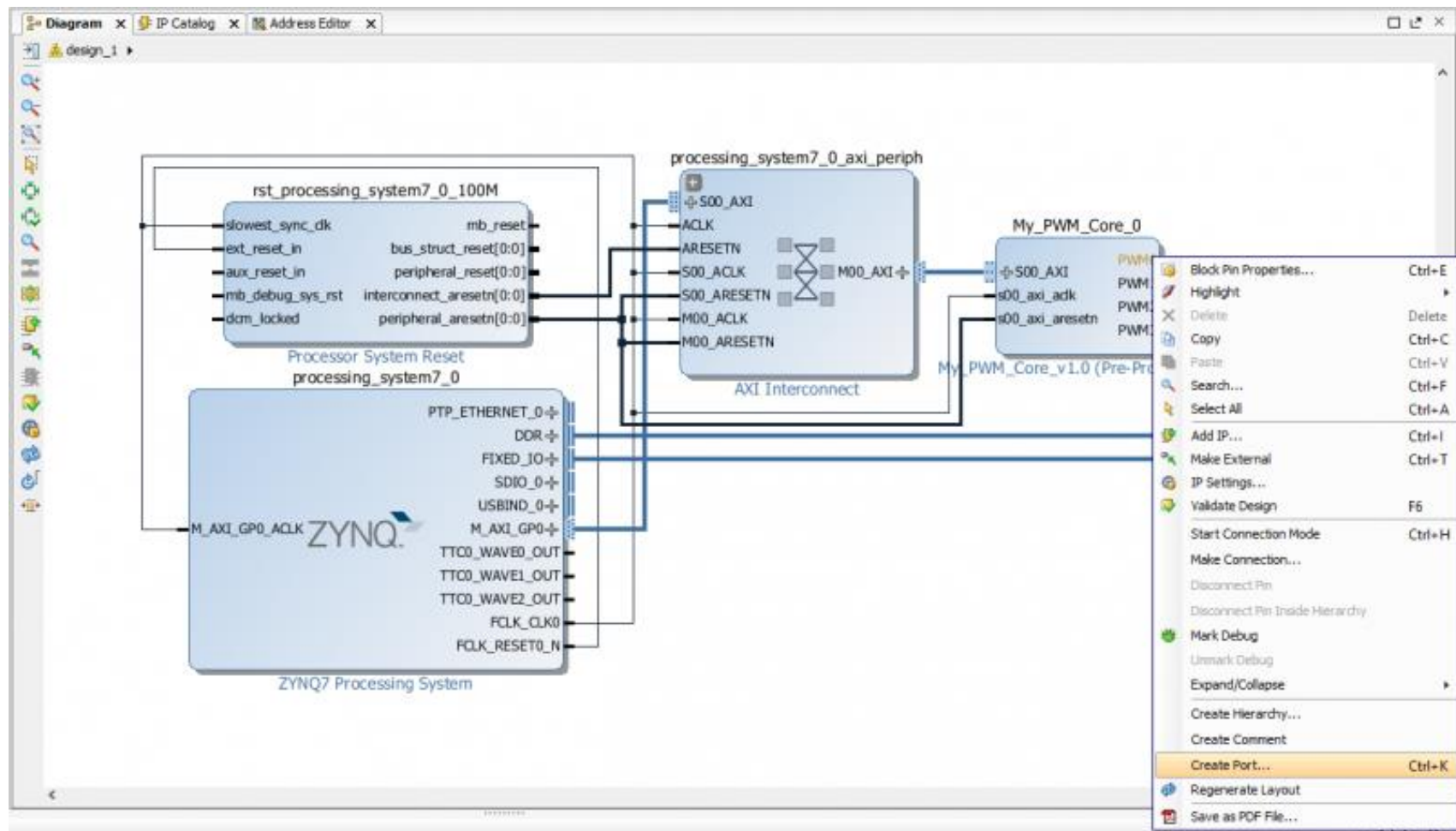
7.5) 설계는 아래와 같아야 한다:



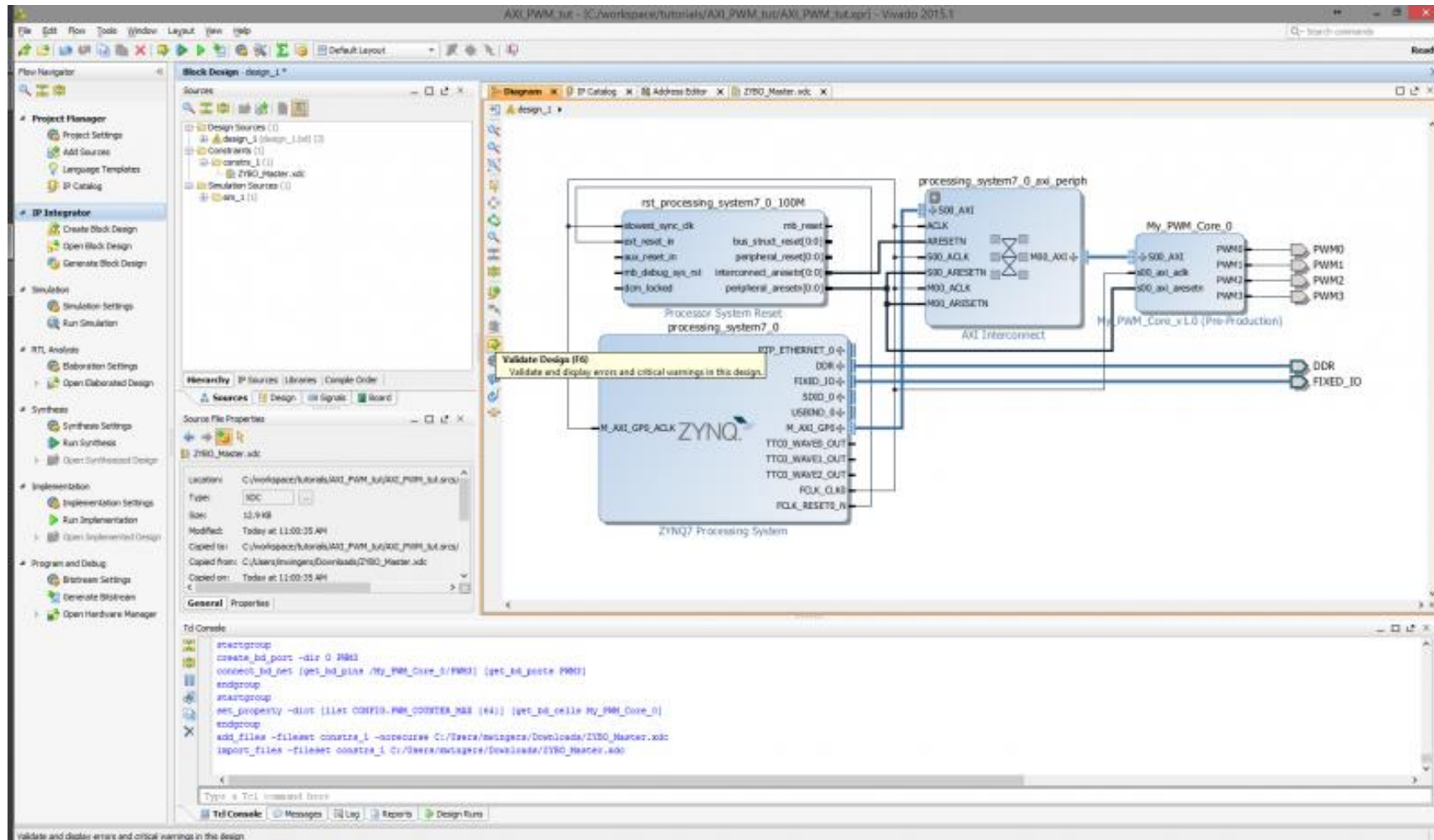
- 7.6) PWM 코어를 더블 클릭하여 앞서 만든 parameter 를 모두 사용자 정의한다.
PWM Counter Max 를 1024 로 설정하고 OK 를 클릭한다.



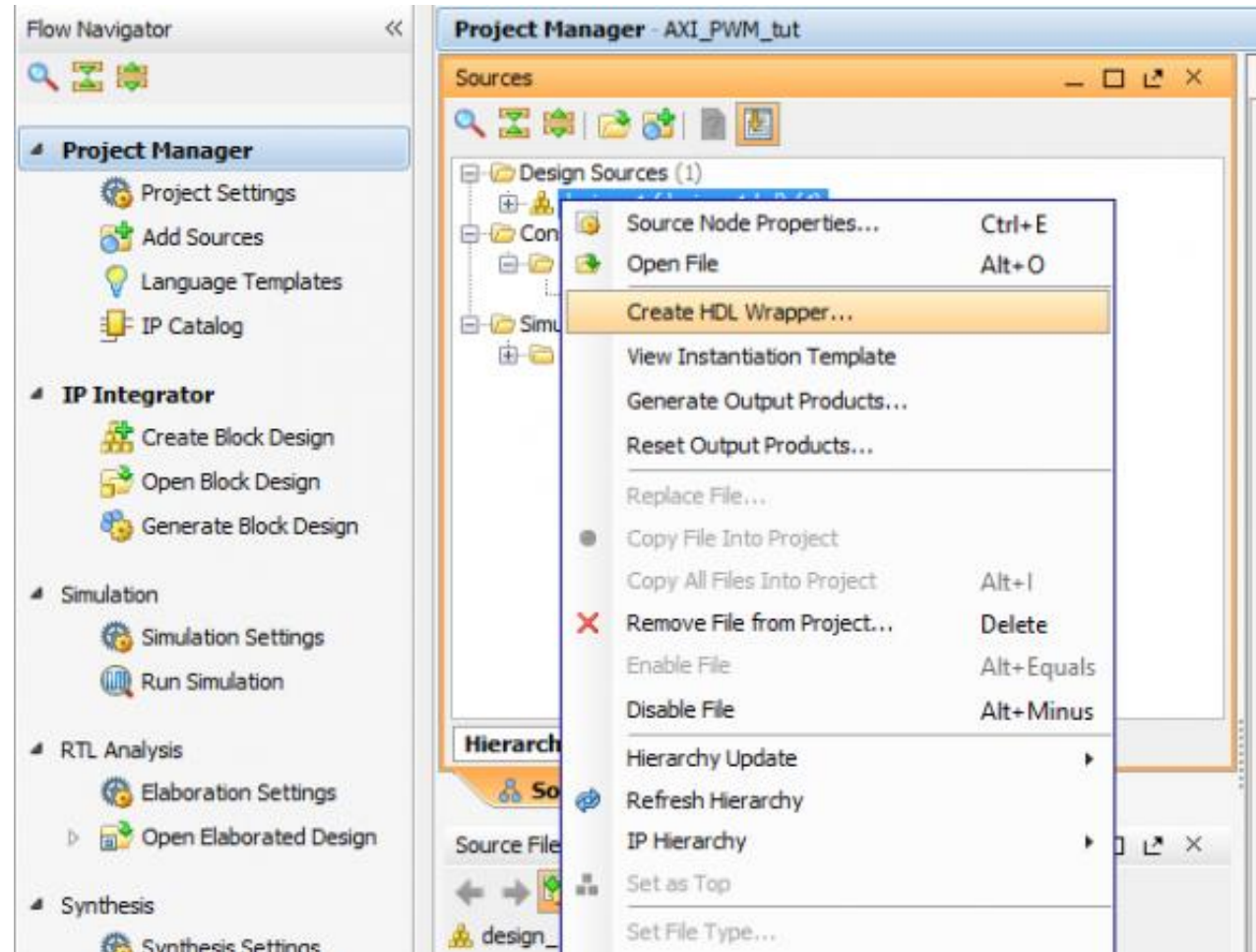
7.7) 각 PWM 신호를 마우스 우클릭 하고 Create Port... 를 선택한다.



7.8) Validate Design 버튼을 클릭한다.



7.9) Project Manager 에서 "design_1.db" 파일을 마우스 우클릭하고 Create HDL Wrapper 를 누른다.



7.10) design_1_wrapper.v 를 더블 클릭하여 편집기에서 연다.
PWM 신호의 이름을 적는다.
여기서 다운로드 할 수 있는 "Zybo_master.xdc" constraint 파일을 추가한다.
https://reference.digilentinc.com/media/zybo/zybo_master.zip
xdc 파일의 LED 코드 줄의 주석 처리를 제거하고 아래와 같이 변경한다:

```
##IO_L23P_T3_35  
set_property PACKAGE_PIN M14 [get_ports {led[0]]}  
set_property IOSTANDARD LVCMOS33 [get_ports {led[0]]}
```

```
##IO_L23N_T3_35  
set_property PACKAGE_PIN M15 [get_ports {led[1]]}  
set_property IOSTANDARD LVCMOS33 [get_ports {led[1]]}
```

```
##IO_0_35  
set_property PACKAGE_PIN G14 [get_ports {led[2]]}  
set_property IOSTANDARD LVCMOS33 [get_ports {led[2]]}
```

```
##IO_L3N_T0_DQS_AD1N_35  
set_property PACKAGE_PIN D18 [get_ports {led[3]]}  
set_property IOSTANDARD LVCMOS33 [get_ports {led[3]]}
```

이를 아래와 같이 변경한다:

```
##IO_L23P_T3_35  
set_property PACKAGE_PIN M14 [get_ports PWM0]  
set_property IOSTANDARD LVCMOS33 [get_ports PWM0]
```

```
##IO_L23N_T3_35  
set_property PACKAGE_PIN M15 [get_ports PWM1]  
set_property IOSTANDARD LVCMOS33 [get_ports PWM1]
```

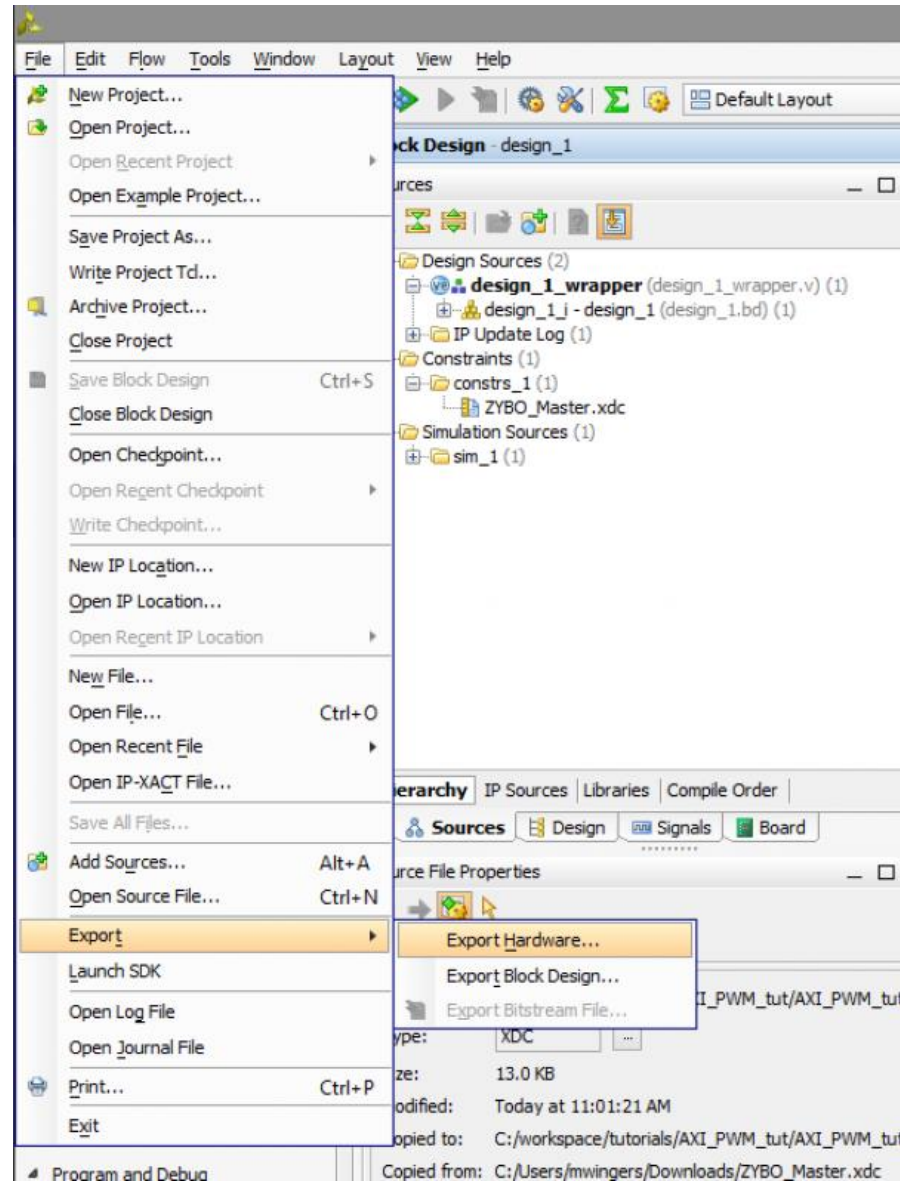
```
##IO_0_35  
set_property PACKAGE_PIN G14 [get_ports PWM2]  
set_property IOSTANDARD LVCMOS33 [get_ports PWM2]
```

```
##IO_L3N_T0_DQS_AD1N_35  
set_property PACKAGE_PIN D18 [get_ports PWM3]  
set_property IOSTANDARD LVCMOS33 [get_ports PWM3]
```

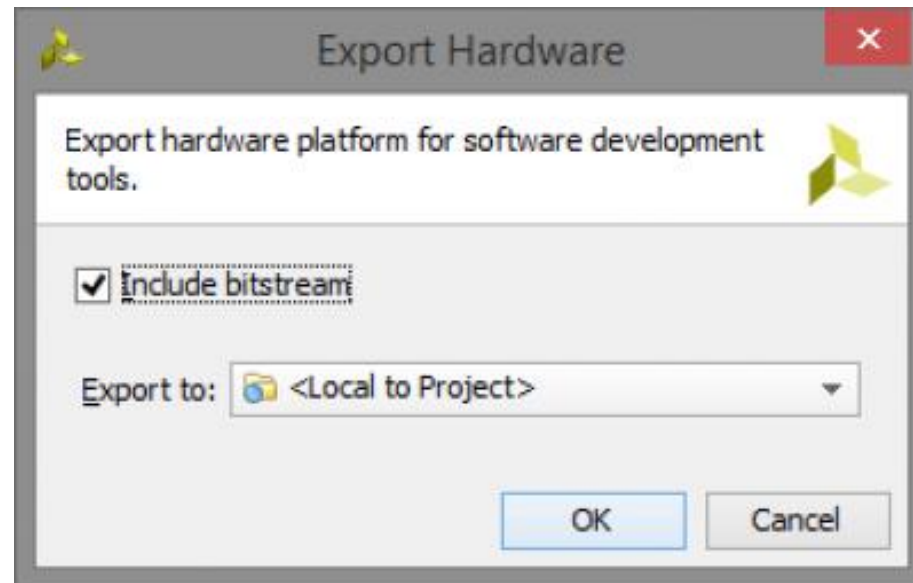
7.11) Generate Bitstream 을 클릭한다.

비트 파일을 빌드하는데 약간의 시간이 걸릴 수 있다.

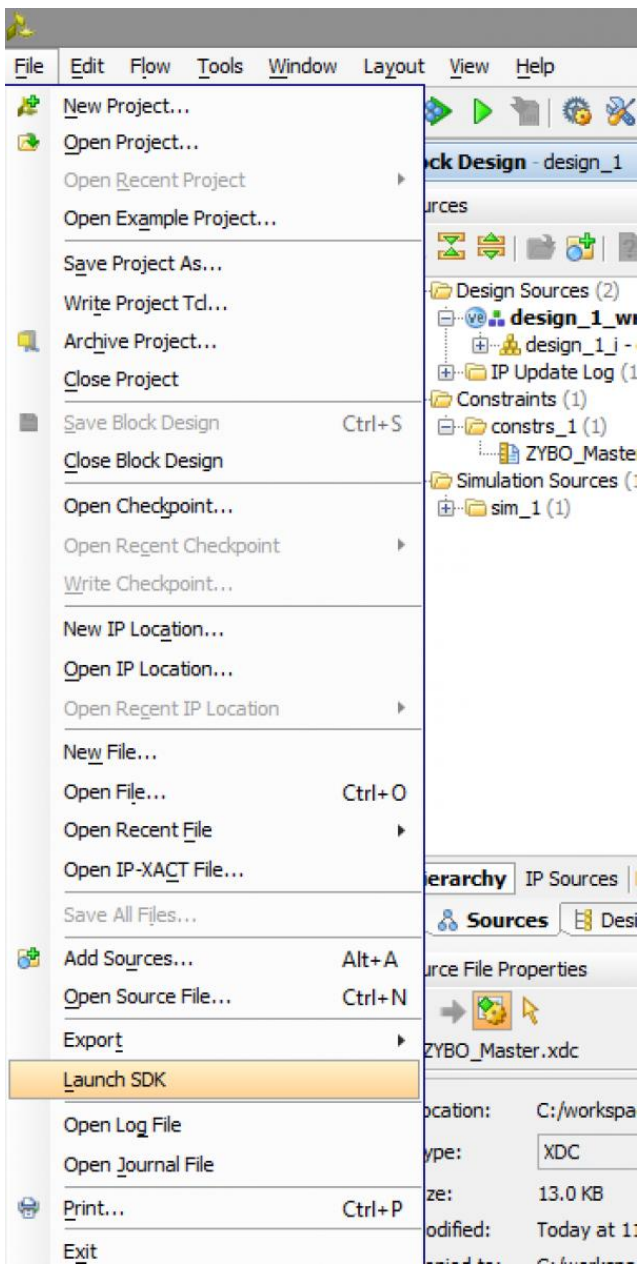
7.12) File -> Export -> Export Hardware ... 로 이동하여 HW 를 export 한다.



7.13) Include bitstream 박스를 체크하고 OK 를 클릭한다.

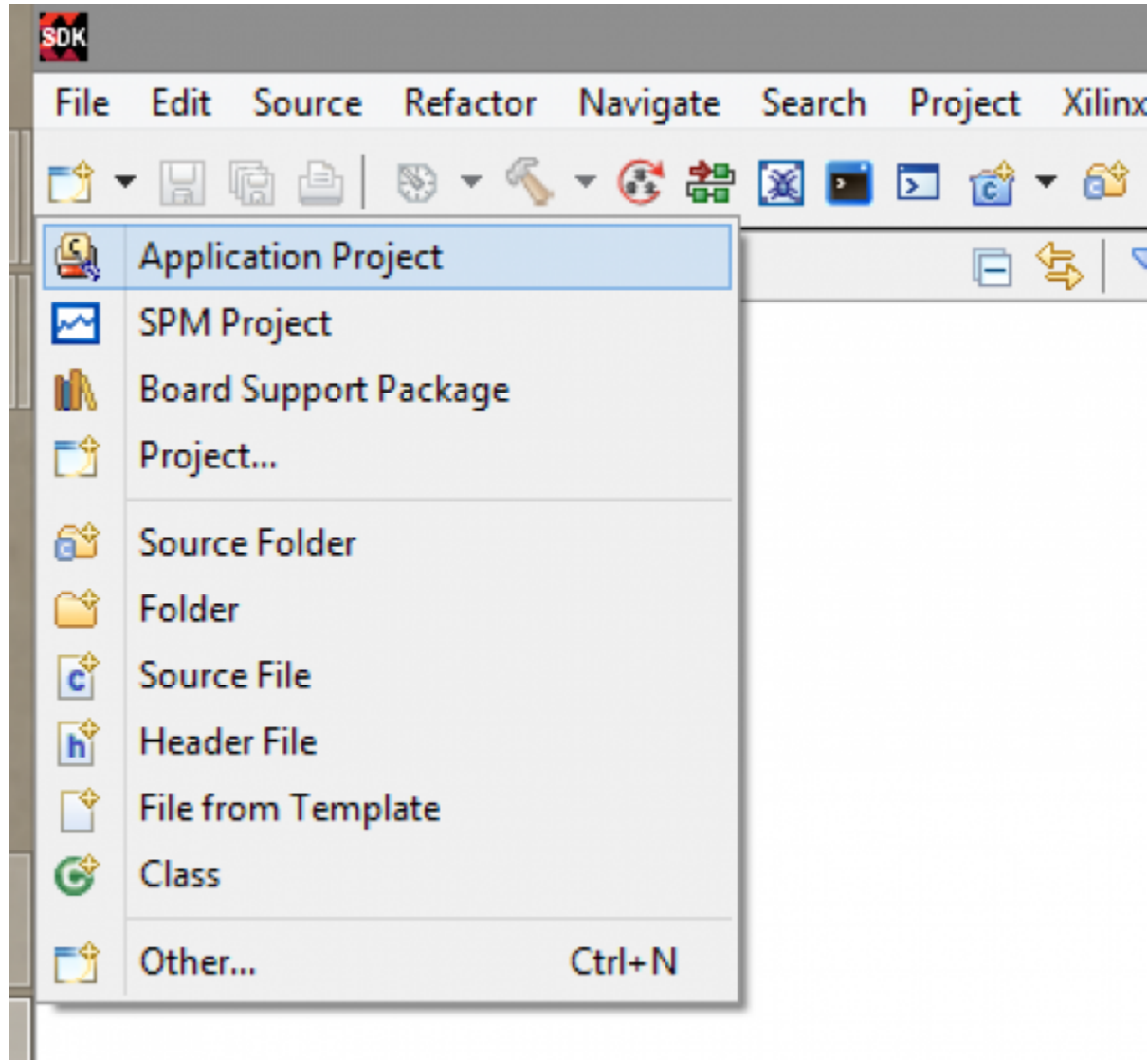


7.14) File -> Launch SDK 를 선택하고 팝업 창에서 OK 를 누른다.



8. Programming in SDK

8.1) 새로운 응용 프로그램 프로젝트 만들기



8.2) 창을 설정하고 Next 및 Finish 를 클릭한다.

New Project

Application Project
Create a managed make application project.

Project name:

☒ Use default location

Location:

Choose file system:

OS Platform:

Target Hardware

Hardware Platform:

Processor:

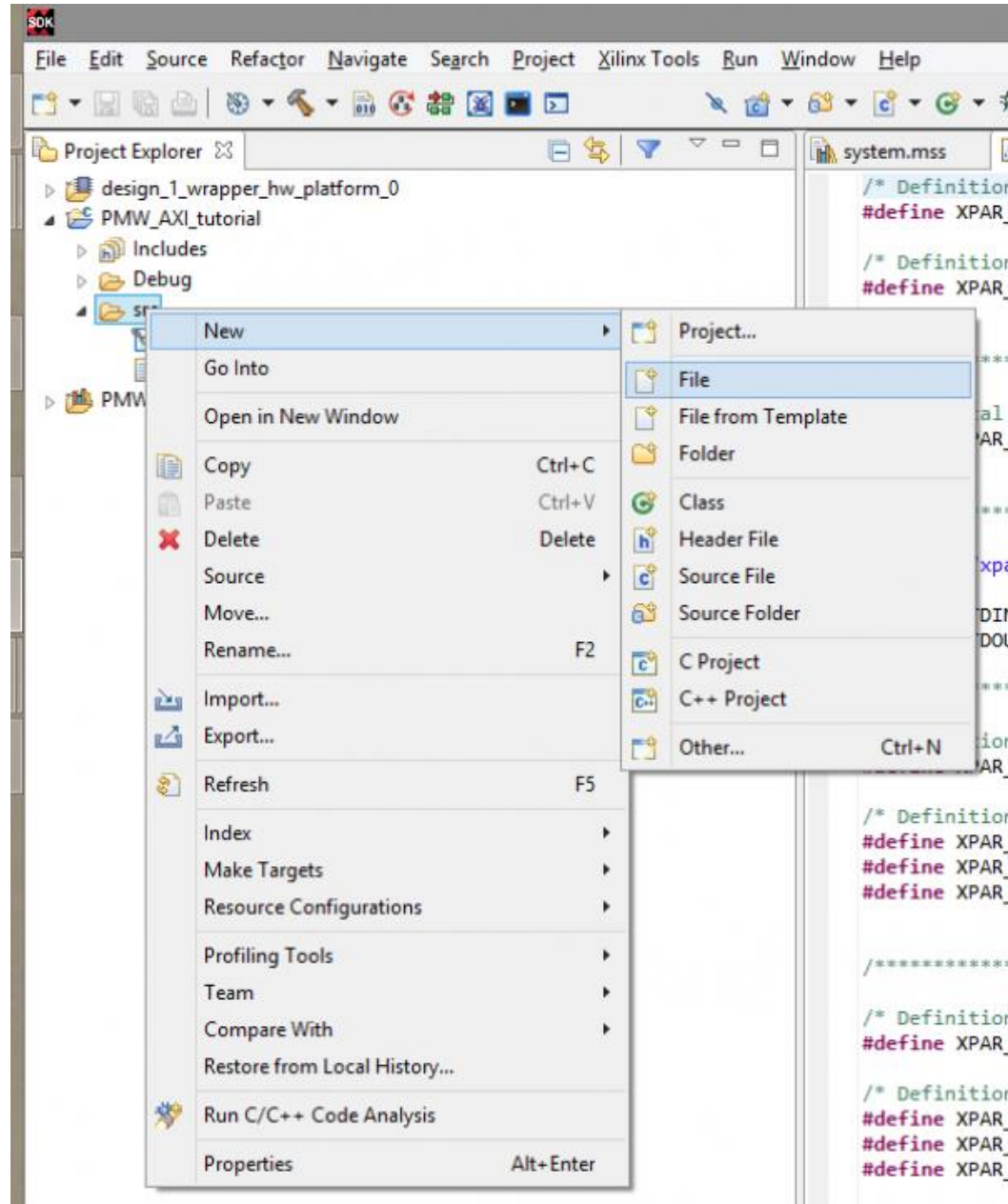
Target Software

Language: ☒ C ☐ C++

Board Support Package: ☒ Create New

☐ Use existing

- 8.3) PWM_AXI_tutorial -> src 폴더를 확장한다.
src 폴더를 마우스 우클릭하고 새로운 파일을 추가한다.
"main.c" 라는 이름의 파일을 만든다.



8.4) 아래와 같이 내용을 추가한다:

```
#include "xparameters.h"
#include "xil_io.h"
```

```
///#define MY_PWM XPAR_MY_PWM_CORE_0_S00_AXI_BASEADDR //Because of a bug in Vivado 2015.3 and 2015.4, this value is not correct.
#define MY_PWM 0x43C00000 //This value is found in the Address editor tab in Vivado (next to Diagram tab)
```

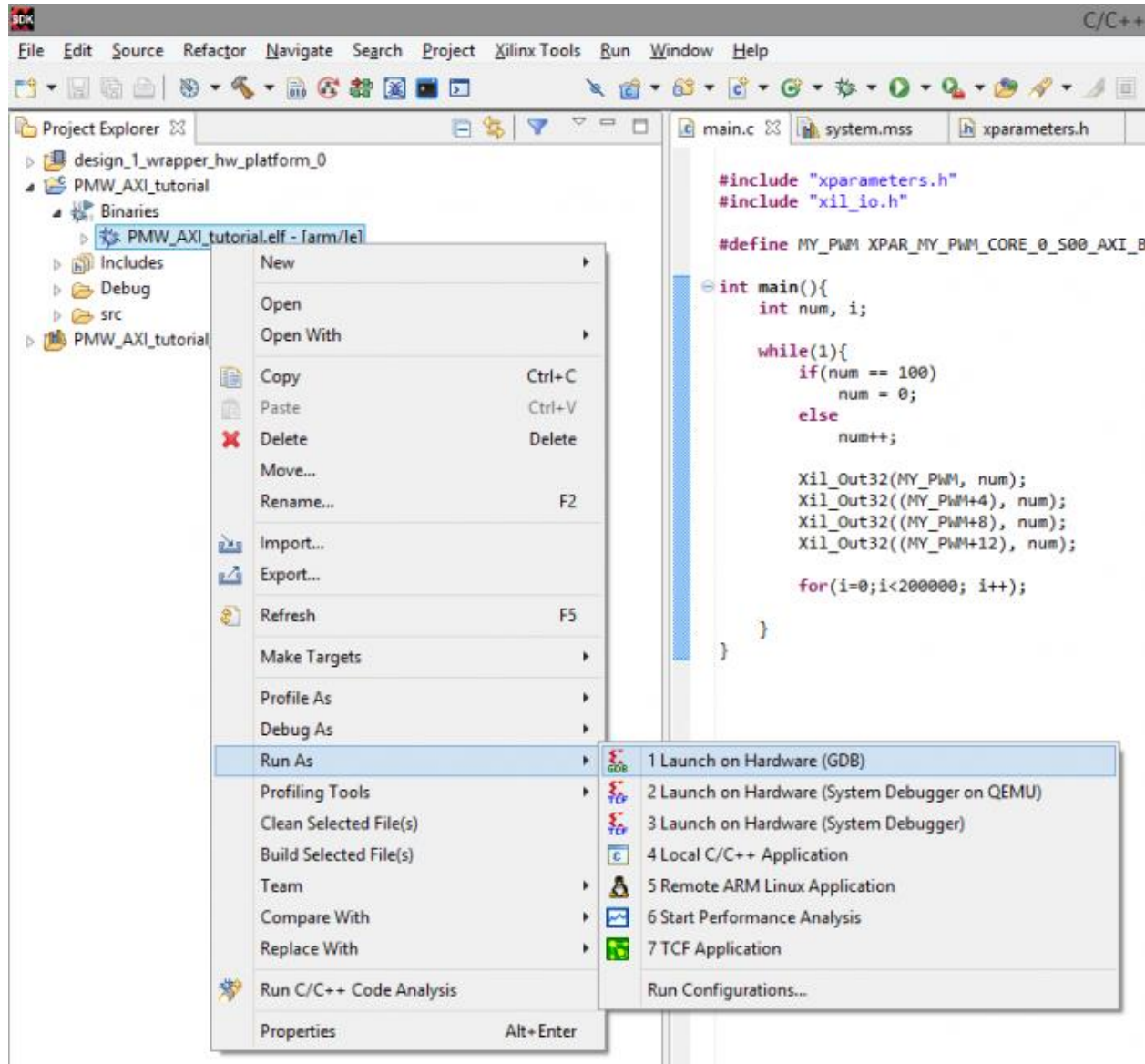
```
int main(){
    int num=0;
    int i;

    while(1){
        if(num == 1024)
            num = 0;
        else
            num++;

        Xil_Out32(MY_PWM, num);
        Xil_Out32((MY_PWM+4), num);
        Xil_Out32((MY_PWM+8), num);
        Xil_Out32((MY_PWM+12), num);

        for(i=0;i<300000; i++);
    }
}
```

- 8.5) FPGA 를 프로그래밍 하려면 Xilinx Tools -> Program FPGA 를 누른다.
SDK 애플리케이션을 ZYBO 로 로드하려면 PWM_AXI_tutorial 을 확장하고
Binary 를 확장하고 확장한 이후 "PWM_AXI_tutorial.elf" 를 우클릭한다.
그리고 Run As -> Launch on Hardware(GDB) 를 누른다.



9. Celebrate!

이제 ZYBO 의 4 개의 LED 가 깜빡인다.

커스텀 IP 코어를 만들었기 때문에 의자에 몸을 기대고 완성감을 느끼도록 한다.

References

1. <https://reference.digilentinc.com/learn/programmable-logic/tutorials/zybo-creating-custom-ip-cores/start>