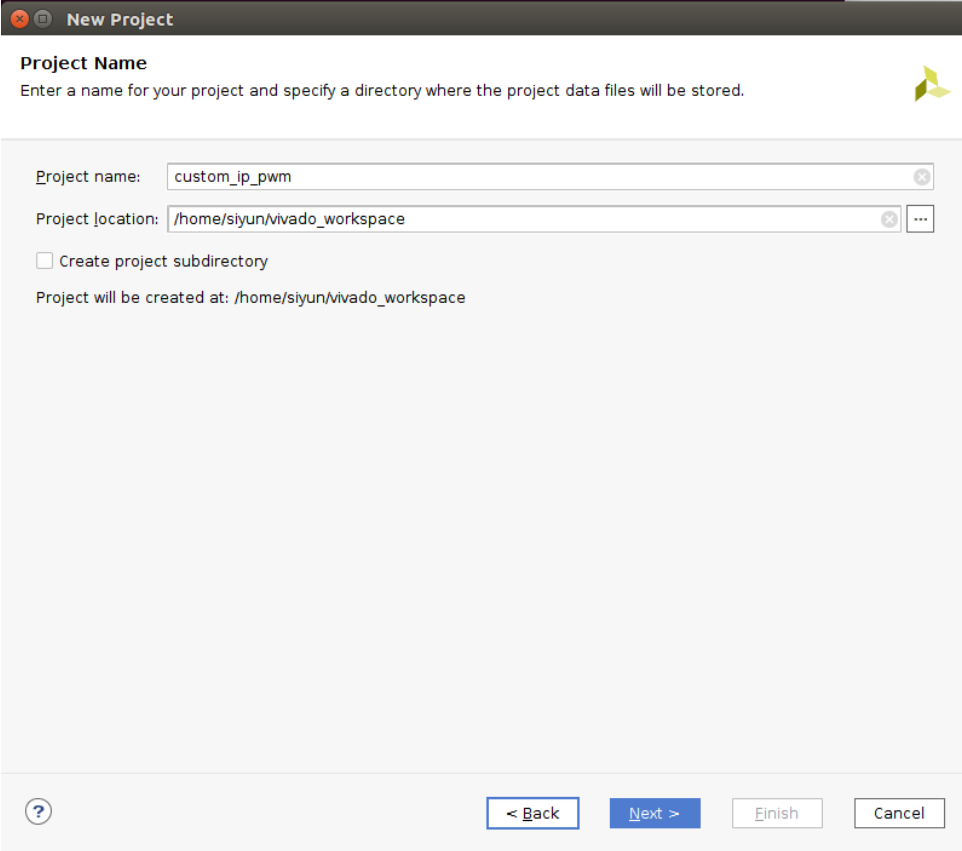


1. Custom IP

1. Vivado 실행 후 New project 생성



New Project

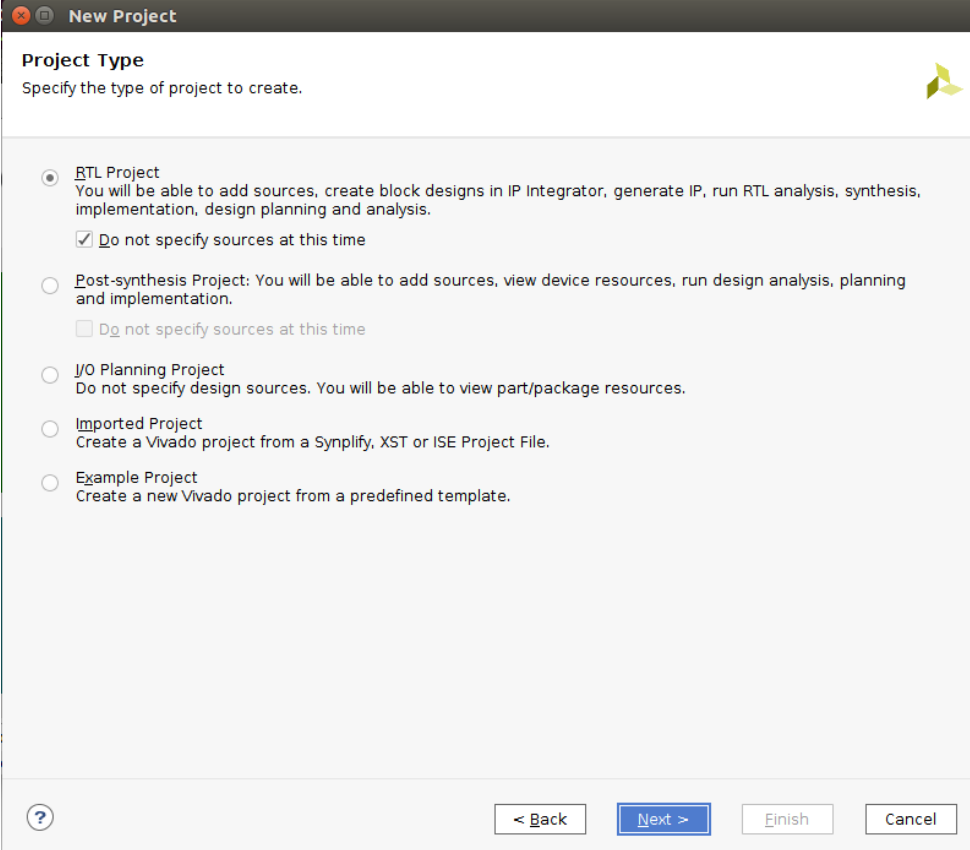
Project Name
Enter a name for your project and specify a directory where the project data files will be stored.

Project name:

Project location:

☐ Create project subdirectory

Project will be created at: /home/siyun/vivado_workspace



New Project

Project Type
Specify the type of project to create.

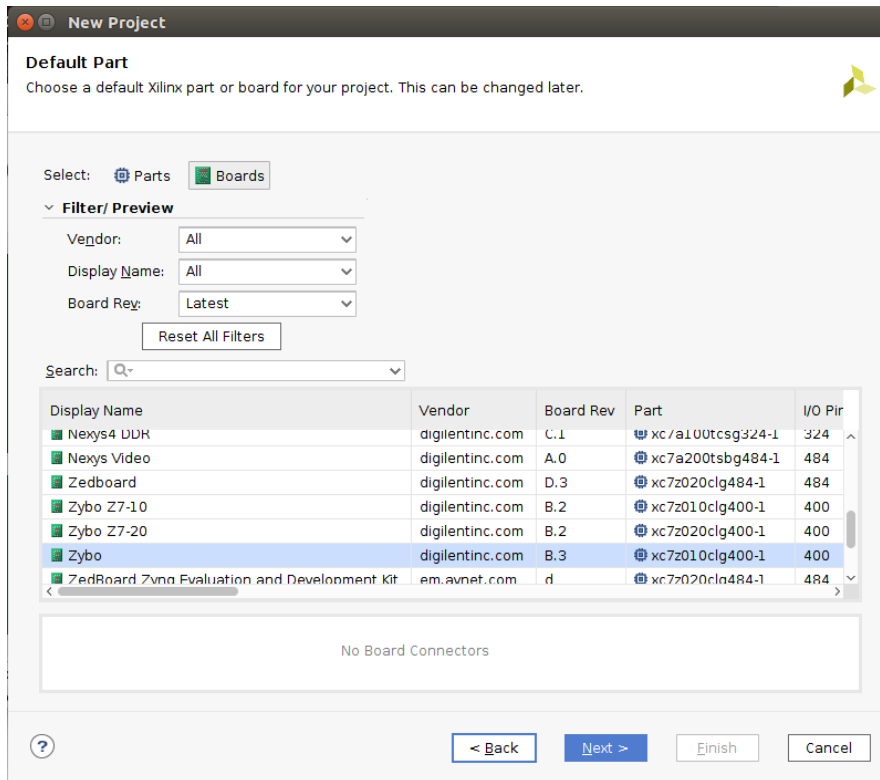
☒ **RTL Project**
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☒ Do not specify sources at this time

☐ **Post-synthesis Project:** You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

☐ **I/O Planning Project**
Do not specify design sources. You will be able to view part/package resources.

☐ **Imported Project**
Create a Vivado project from a Synplify, XST or ISE Project File.

☐ **Example Project**
Create a new Vivado project from a predefined template.

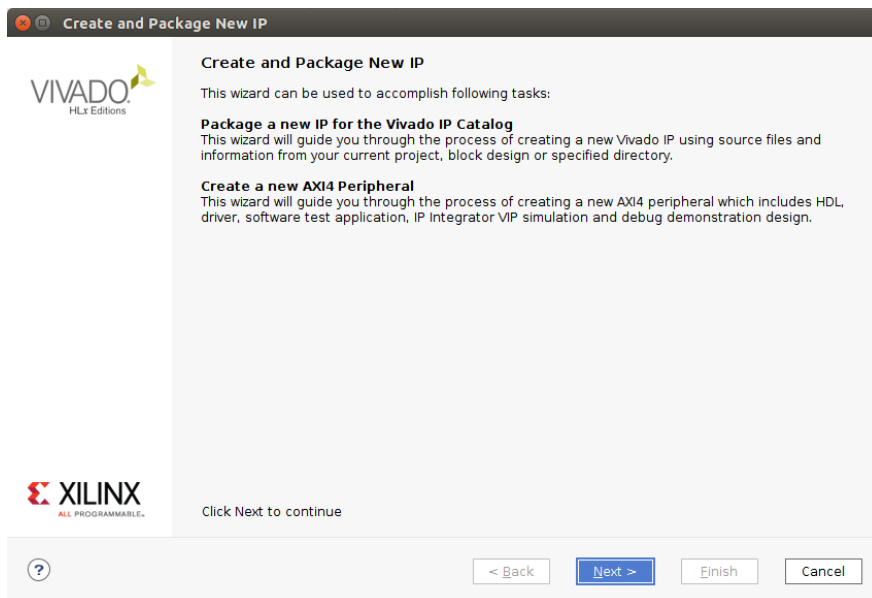


Next → Finish 새로운 프로젝트를 생성했다.

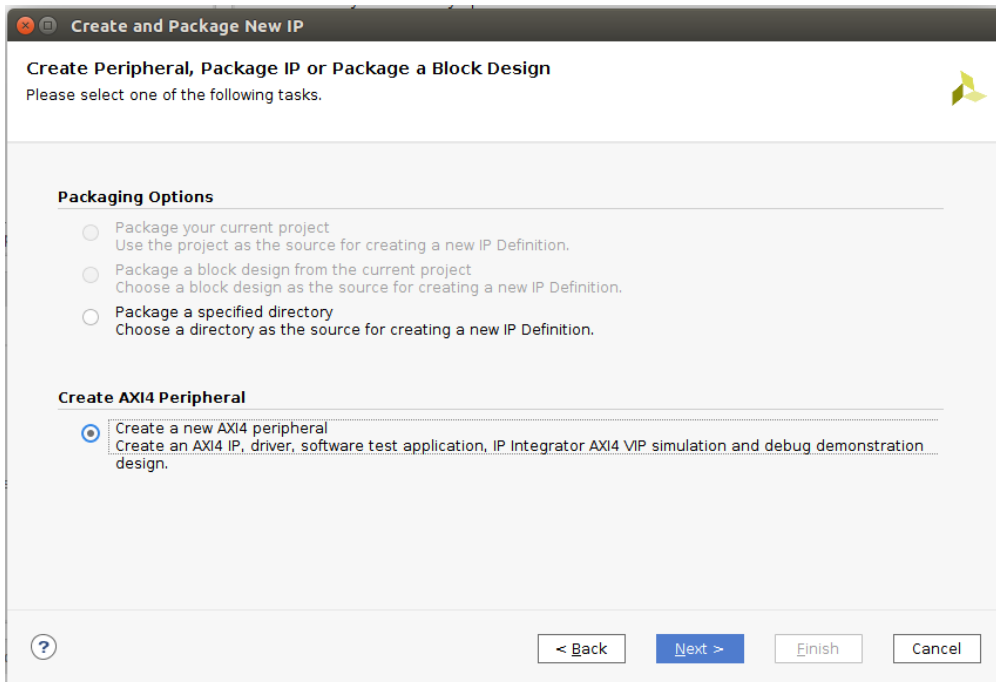
2. Create and Package New IP

커스텀 IP 생성.

Tools → Create and Package New IP



Create a new AXI4 peripheral 선택 후 다음



Create and Package New IP

Create Peripheral, Package IP or Package a Block Design
Please select one of the following tasks.

Packaging Options

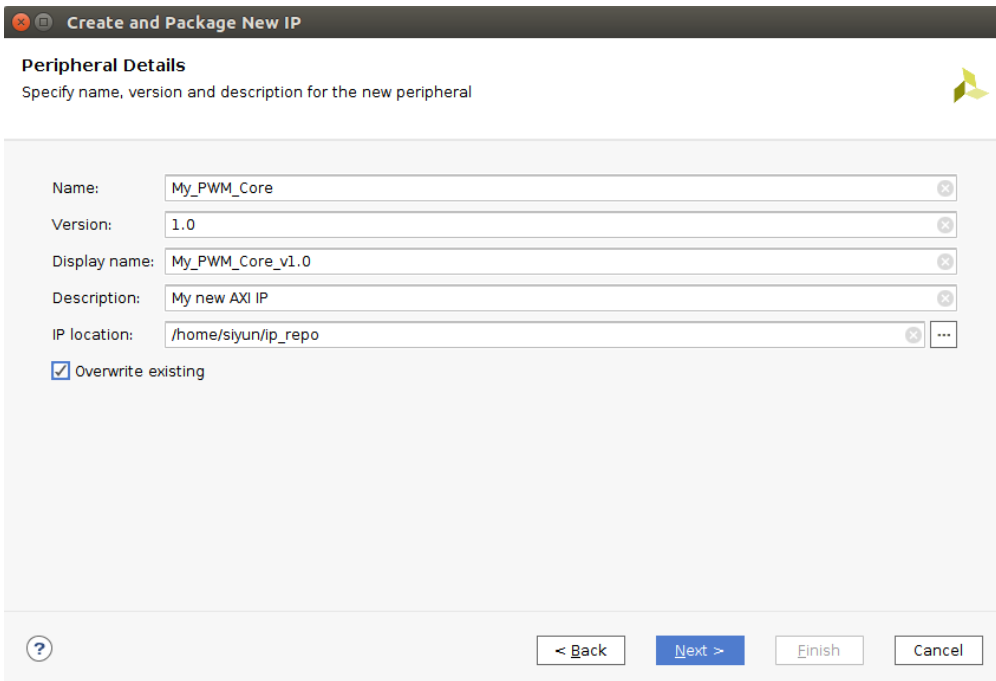
- ☐ Package your current project
Use the project as the source for creating a new IP Definition.
- ☐ Package a block design from the current project
Choose a block design as the source for creating a new IP Definition.
- ☐ Package a specified directory
Choose a directory as the source for creating a new IP Definition.

Create AXI4 Peripheral

- ☒ Create a new AXI4 peripheral
Create an AXI4 IP, driver, software test application, IP Integrator AXI4 VIP simulation and debug demonstration design.

? < Back Next > Finish Cancel

Name 에 입력하고싶은 Custom IP 의 이름을 입력 후 next



Create and Package New IP

Peripheral Details
Specify name, version and description for the new peripheral

Name: My_PWM_Core

Version: 1.0

Display name: My_PWM_Core_v1.0

Description: My new AXI IP

IP location: /home/siyun/ip_repo

☒ Overwrite existing

? < Back Next > Finish Cancel

아무것도 수정하지 말고 next 나중에 수정함.

Create and Package New IP

Add Interfaces
Add AXI4 interfaces supported by your peripheral

☐ Enable Interrupt Support

Interfaces
S00_AXI

Properties:

- Name: S00_AXI
- Interface Type: Lite
- Interface Mode: Slave
- Data Width (Bits): 32
- Memory Size (Bytes): 64
- Number of Registers: 4 [4..512]

My_PWM_Core_v1.0

< Back Next > Finish Cancel

IP 를 원하는 모듈로 수정해야 하기 때문에 Edit IP 선택 후 Finish

Create and Package New IP

Create Peripheral

Peripheral Generation Summary

1. IP (user.org:user:My_PWM_Core:1.0) with 1 interface(s)
2. Driver(v1_00_a) and testapp [more info](#)
3. AXI4 VIP Simulation demonstration design [more info](#)
4. AXI4 Debug Hardware Simulation demonstration design [more info](#)

Peripheral created will be available in the catalog :
/home/siyun/ip_repo

Next Steps:

- ☐ Add IP to the repository
- ☒ Edit IP
- ☐ Verify Peripheral IP using AXI4 VIP
- ☐ Verify peripheral IP using JTAG interface

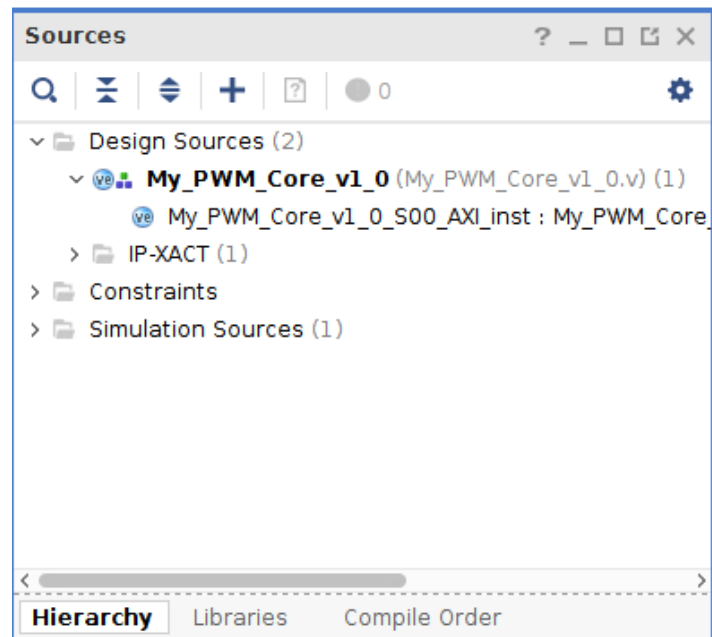
Click Finish to continue

< Back Next > Finish Cancel

Finish 하면 Source 에 2 개의 파일이 생긴다.

위에 My_PWM_Core_v1_0 은 탑모듈이고 My_PWM_Core_v1_0_S00_AXI_inst 탑모듈 안의 로직이다.

VHDL 컴포넌트 와 같은 개념



먼저 하위모듈인 My_PWM_Core_v1_0_S00_AXI_inst 부터 수정하도록 한다.

Users to add parameters here 에 파라미터를 하나 추가한다

파라미터는 C 언어에서 #define 같은 역할을 한다. PWM 주기설정

```
module My_PWM_Core_v1_0_S00_AXI #
(
    // Users to add parameters here
    parameter integer PWM_COUNTER_MAX = 2000000,
    // User parameters ends
    // Do not modify the parameters beyond this line
)
```

users to add ports here 에는 우리가 만든 포트를 입력한다. zynq 클럭으로 동작하여 pwm 파형을 내보내므로 PWM0 핀을 적어줘야한다. 핀설정이 헛갈릴 경우 Vivado 임플리멘테이션 I/O Port 를 확인해도 무관하다.

```
    // Users to add ports here
    output wire PWM0,
    // User ports ends
    ...
```

실제 동작 로직이다. Verilog 또는 VHDL 은 IP 만들때 아주 유용하다.

```
// Add user logic here
reg [31:0] 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;
// User logic ends
```

이제 top_module 을 수정해보도록 한다.

첫번째로 파라미터 추가이다. 이 파라미터는 default 값이 될것이다.

```
module My_PWM_Core_v1_0 #  
(  
    // Users to add parameters here  
    parameter integer PWM_COUNTER_MAX = 128,  
    // User parameters ends  
    ...  
)
```

PWM0 을 패키지할 PWM0 을 추가한다. (이름을 다르게해도 상관없다)

베릴로그 top_module 을 공부하면 더 쉽게 이해가 가능하다.

Wire 를 써준 이유는 탑모듈은 겉 겹대기일뿐 실제핀이랑은 선으로 연결되 있기 때문이다.

```
    // Users to add ports here  
    output wire PWM0,  
    // User ports ends  
    // Do not modify the ports beyond this
```

Top module 이 다른모듈과 연결된것을 정의한 것들이다.

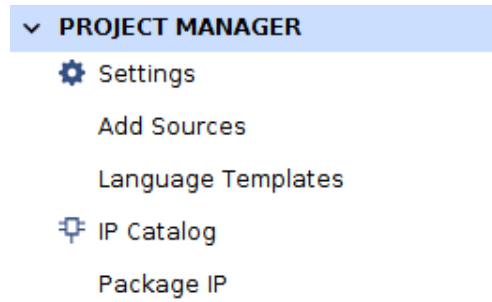
예전 VHDL 의 컴포넌트와 같다.

.PWM_COUNTER_MAX(PWM_COUNTER_MAX) 와

.PWM0(PWM0) 을 추가하였다.

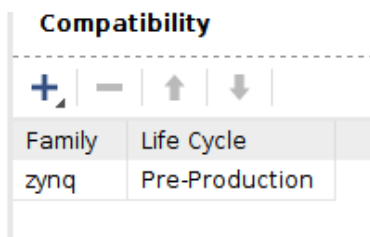
```
    // Instantiation of Axi Bus Interface S00_AXI  
    My_PWM_Core_v1_0_S00_AXI # (  
        .C_S_AXI_DATA_WIDTH(C_S00_AXI_DATA_WIDTH),  
        .C_S_AXI_ADDR_WIDTH(C_S00_AXI_ADDR_WIDTH),  
        .PWM_COUNTER_MAX(PWM_COUNTER_MAX) //new  
    ) My_PWM_Core_v1_0_S00_AXI_inst (  
        .PWM0(PWM0), //new  
        .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)  
    );
```

이제 베릴로그 코드는 수정이 완료되었다.
왼쪽 프로젝트 매니저에서 Package IP 를 클릭한다.

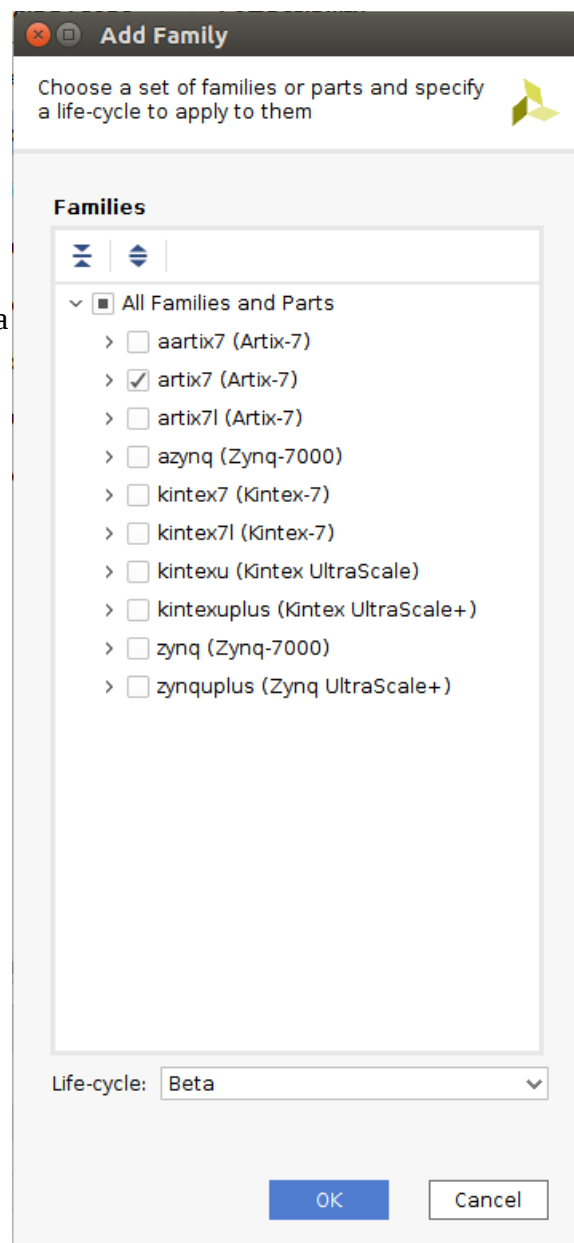


왼쪽 목록들 중 Compatibility 를 선택한다.

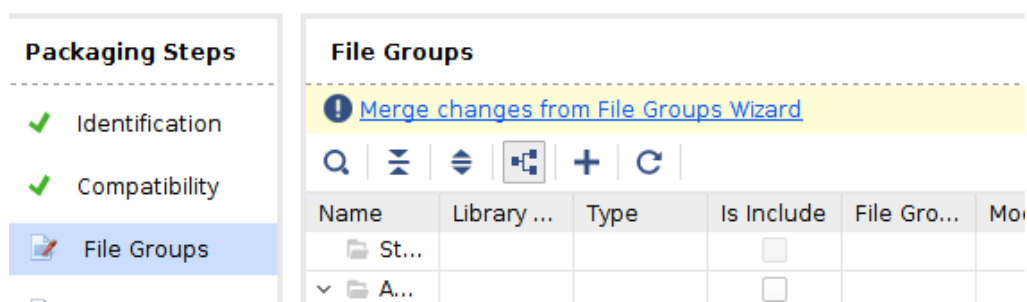
+버튼을 클릭하여 Add Family Explicitly 을 선택한다.



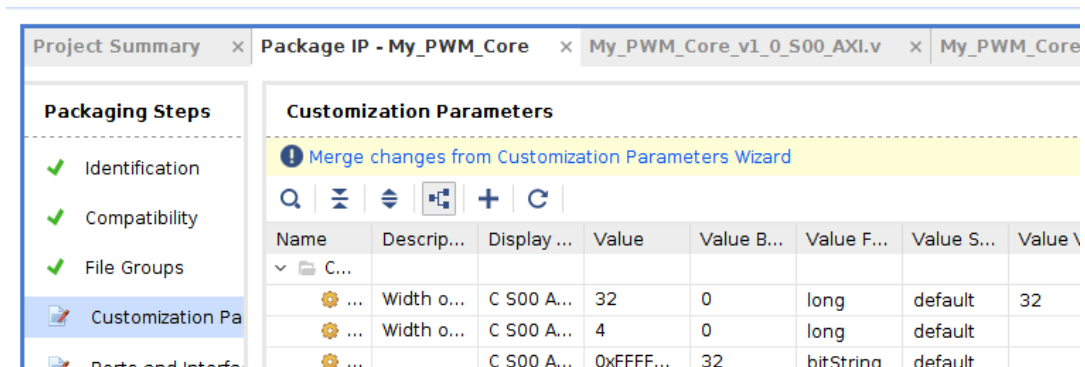
artix7 을 선택후 Life-cycle Beta 로 설정 후 OK 를 클릭한다.



왼쪽 목록에서 File Groups 을 선택후 Merge changes from file Groups Wizard 를 클릭한다.



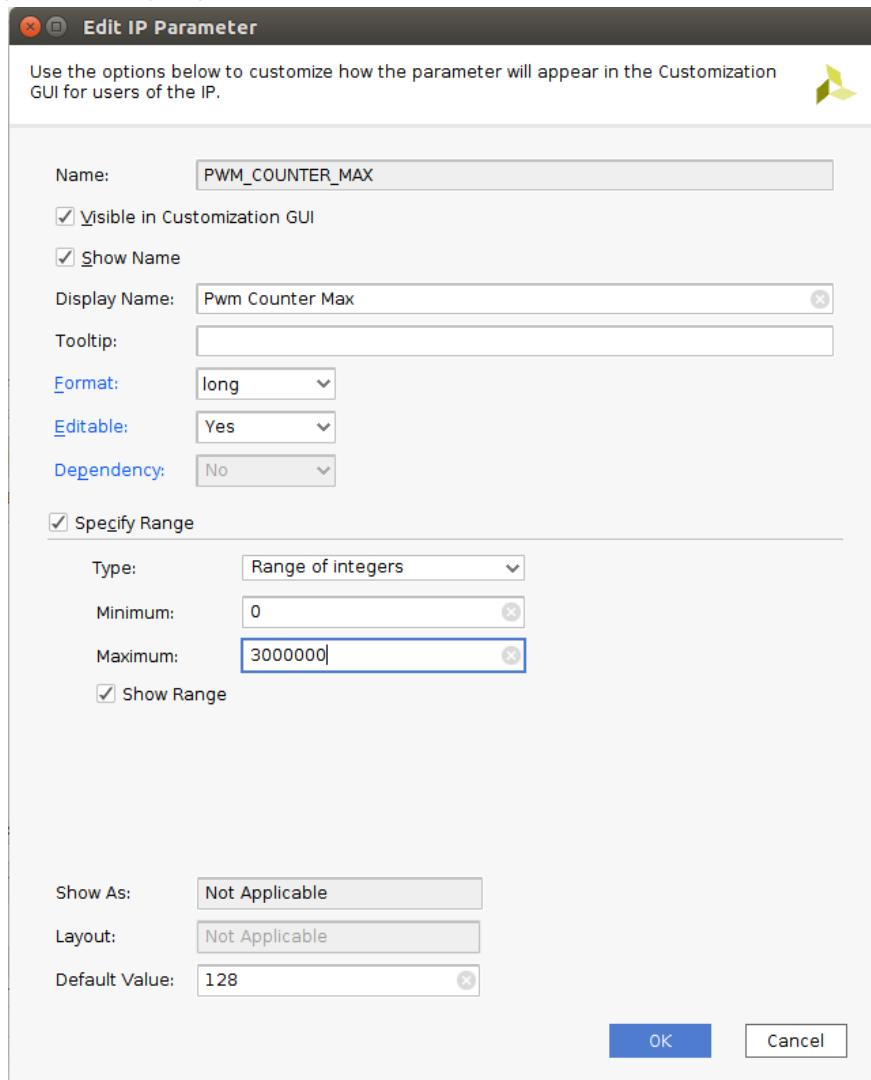
Customization Parameters 도 클릭하여 위와 같은 작업을 한 후



Hidden Parameters 에 PWM_COUNTER_MAX 를 더블클릭한다.

C_S00_AXI_HIGHA...		C S00 A...	0x0000...
Hidden Parameters			
PWM_COUNTER_MAX		Pwm Co...	128

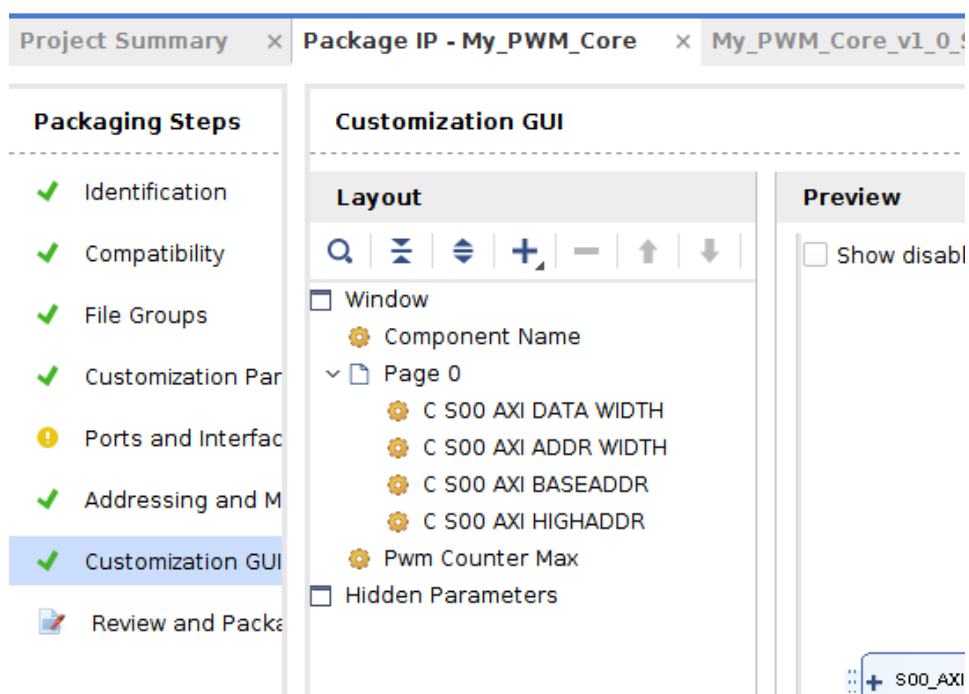
아래와 같이 설정후 Ok 를 눌러준다.



The 'Edit IP Parameter' dialog box is shown. It contains the following fields and options:

- Name: PWM_COUNTER_MAX
- ☒ Visible in Customization GUI
- ☒ Show Name
- Display Name: Pwm Counter Max
- Tooltip: (empty)
- Format: long
- Editable: Yes
- Dependency: No
- ☒ Specify Range
 - Type: Range of integers
 - Minimum: 0
 - Maximum: 3000000
 - ☒ Show Range
- Show As: Not Applicable
- Layout: Not Applicable
- Default Value: 128
- Buttons: OK, Cancel

Customization GUI 에서 Page 0 폴더 밖에 빠져있는 Pwm Counter Max 를 드래그하여 Page 0 에 넣어준다.



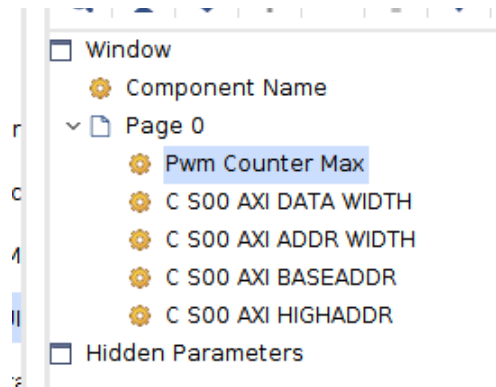
The 'Customization GUI' interface is shown. It includes a 'Packaging Steps' sidebar on the left and a 'Customization GUI' main area on the right. The 'Packaging Steps' sidebar shows the following steps:

- Identification
- Compatibility
- File Groups
- Customization Parameters
- Ports and Interfaces
- Addressing and Memory
- Customization GUI (highlighted)
- Review and Packaging

The 'Customization GUI' main area shows a 'Layout' section with a search bar and a list of parameters. The 'Page 0' folder is expanded, showing the following parameters:

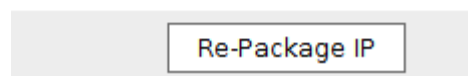
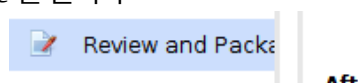
- C S00 AXI DATA WIDTH
- C S00 AXI ADDR WIDTH
- C S00 AXI BASEADDR
- C S00 AXI HIGHADDR
- Pwm Counter Max

The 'Hidden Parameters' section is also visible. The 'Preview' section on the right shows a checkbox for 'Show disabled'.



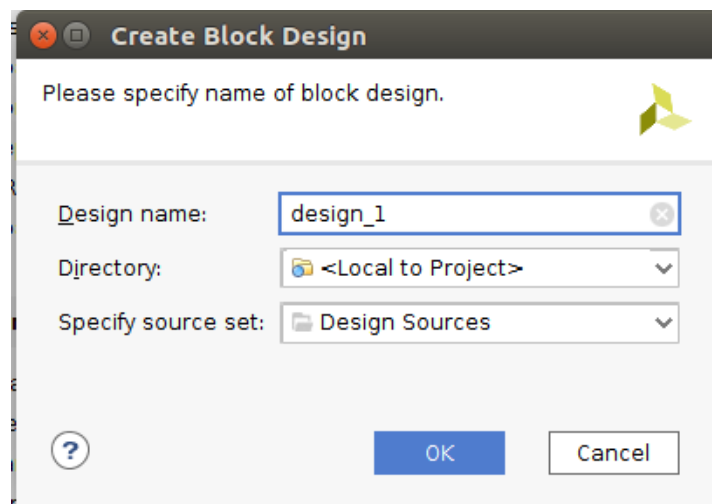
이제 왼쪽 목록에 Review and Package 를 클릭 후

Re-Package IP 를 클릭후 custom IP
생성을 완료한다.

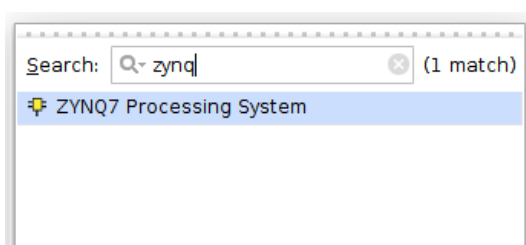


이제 Custom IP 가 완성이 되었다. 만든걸 이용해보면 다음과 같다

1.블록디자인을 만든다.

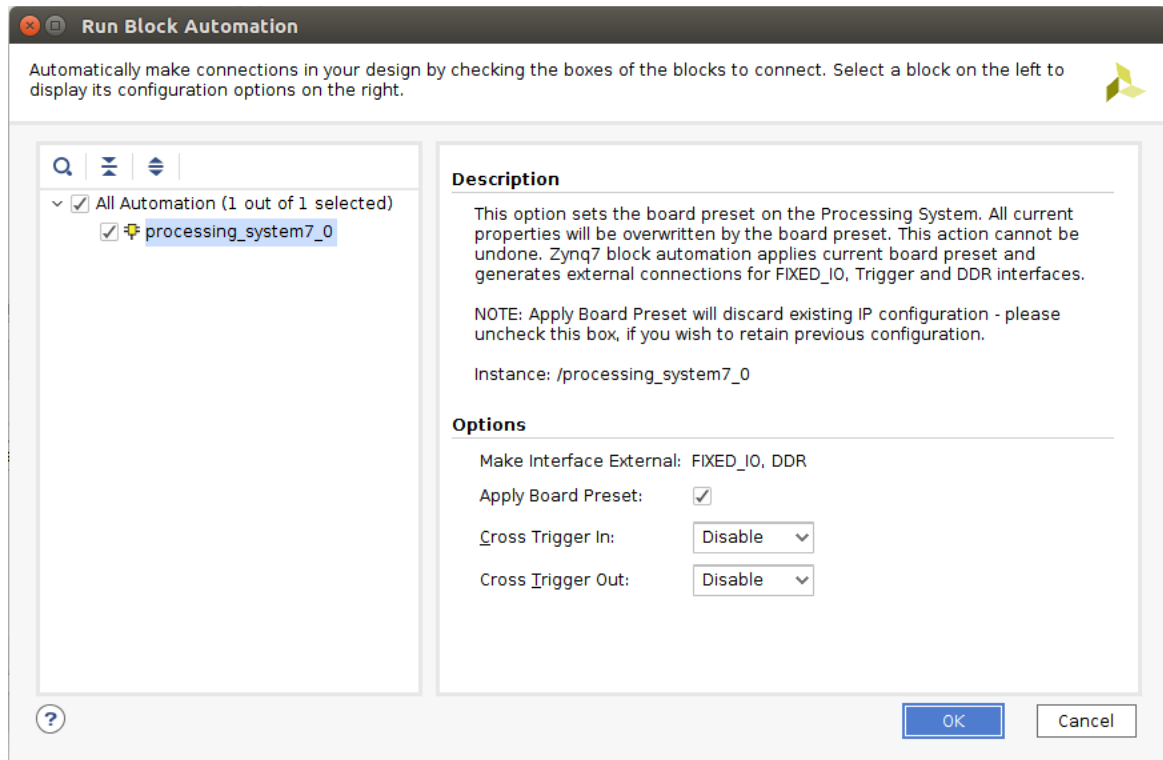


2.add IP 를 한다.

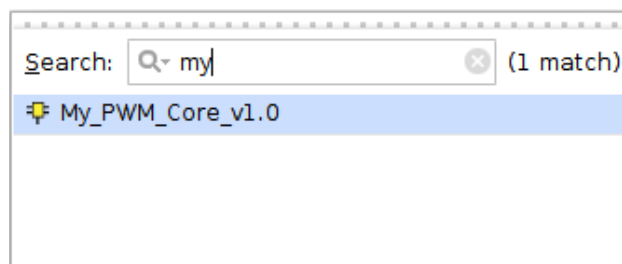


le. Run Block Automation

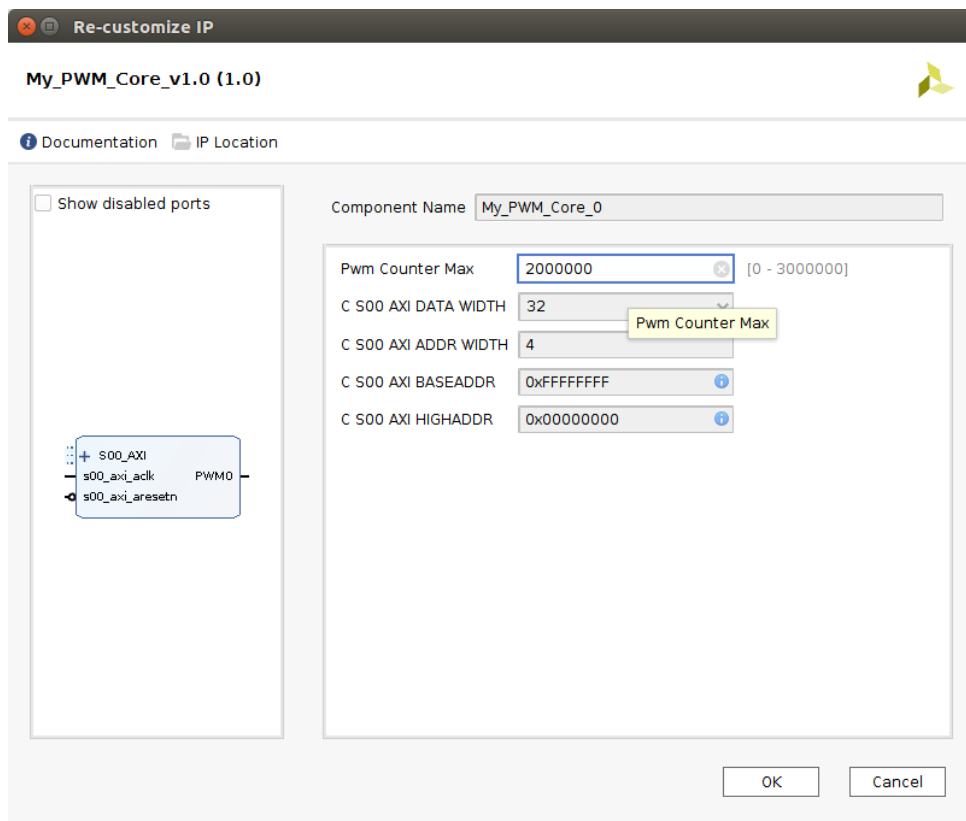
Zynq processing systemp 을 불러오고 Run Block Automation 클릭
아래와 같이 설정 후 OK



3. Add IP 에서 Custom IP 추가




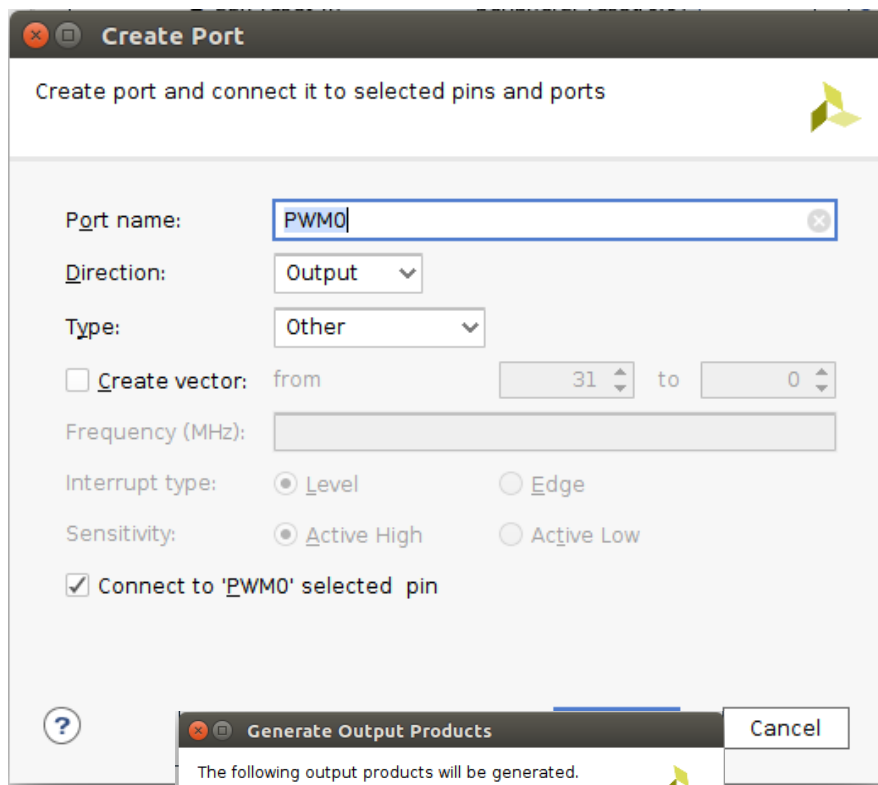
추가했으면
Custom IP
블록을 더블
클릭하여 아
래와 같이 설
정해준다. 설
정 완료 후
OK



4. Run Connection Automation

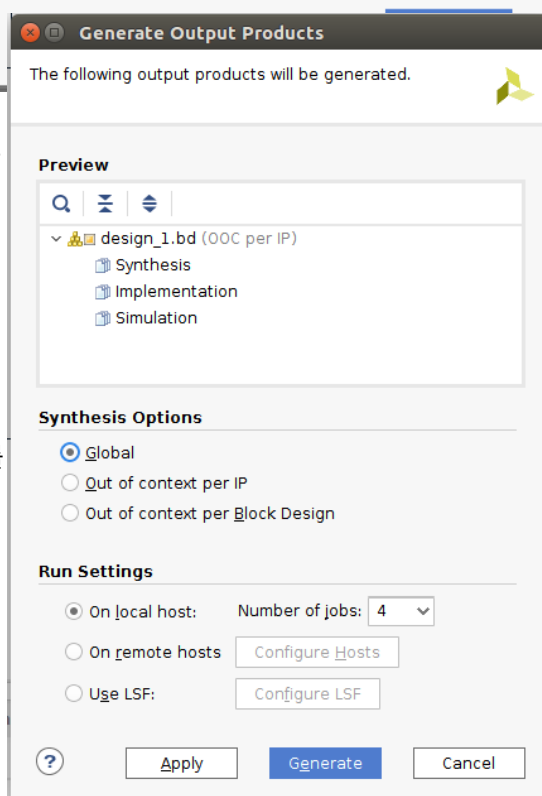
e. Run Connection Automation

5. 상단에 새로고침같이  생긴 버튼으로 회로 모양을 다잡아주고 Custom IP port 를 마우스 우클릭 해 port 를 create 해준다

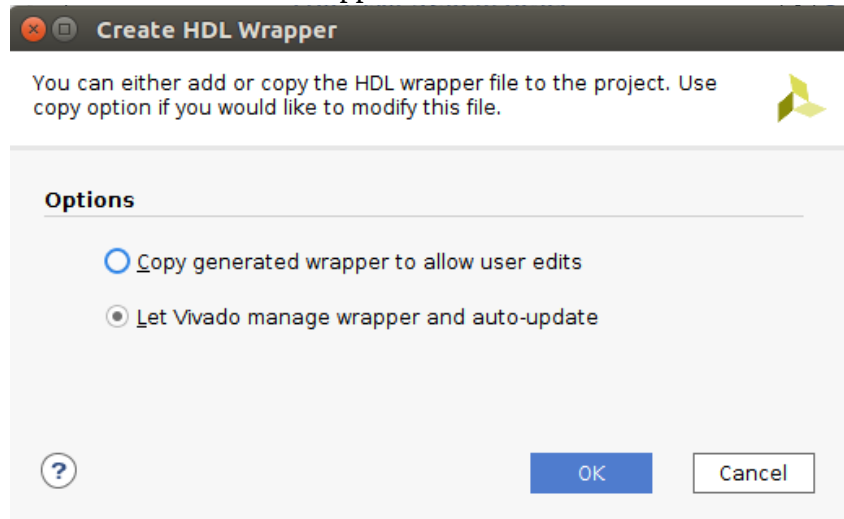


그 후 Validate Design (디자인 이 올바른지 검사)

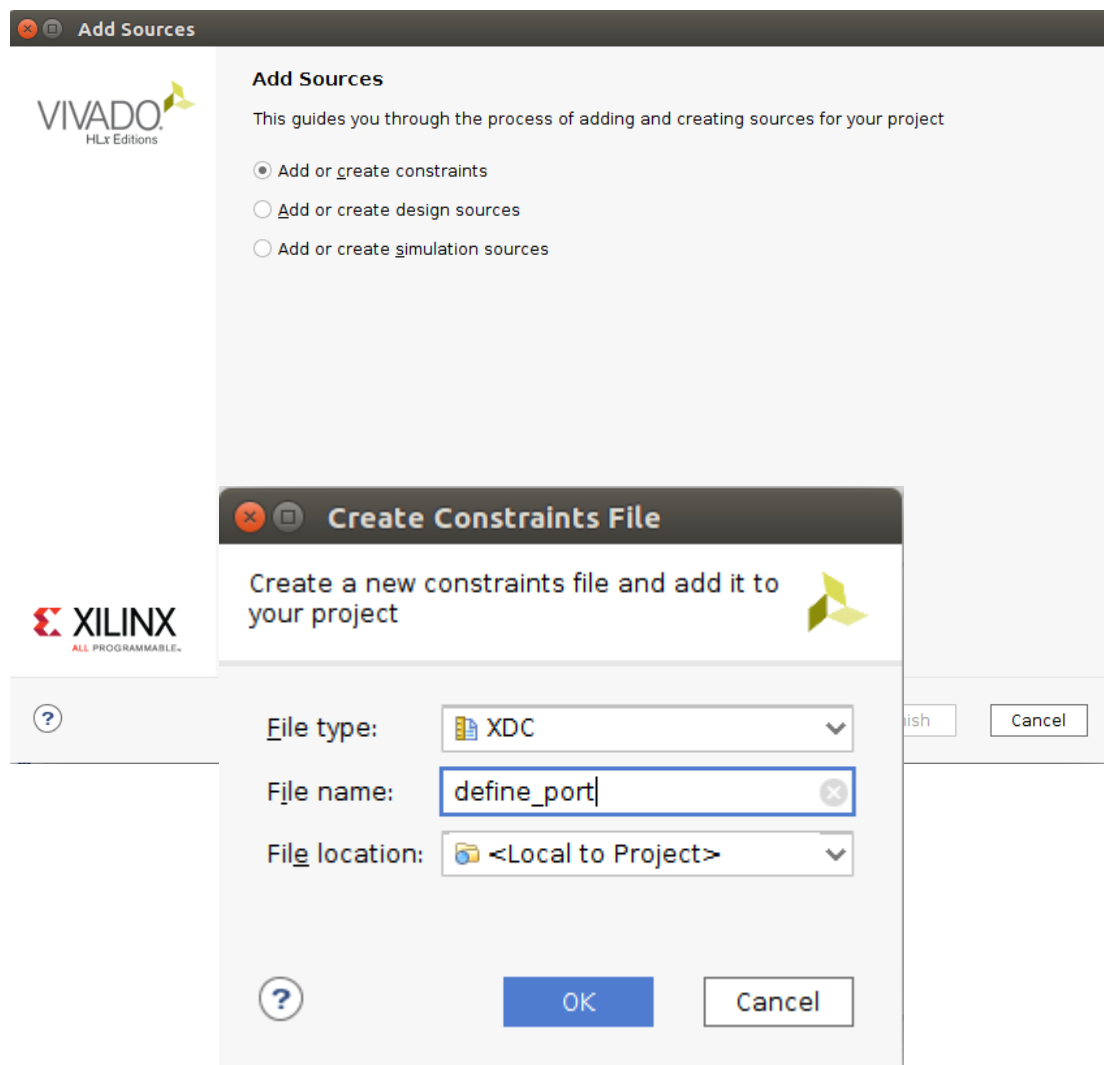
6. project manager source 에서 블록디자인 마우스 우클릭 후 Generate Output Products 를 오른쪽 그림과 같이 설정하여 재 너레이트 해준다.



7. 블록디자인 소스 우클릭 후 create HDL Wrapper 를 해준다.

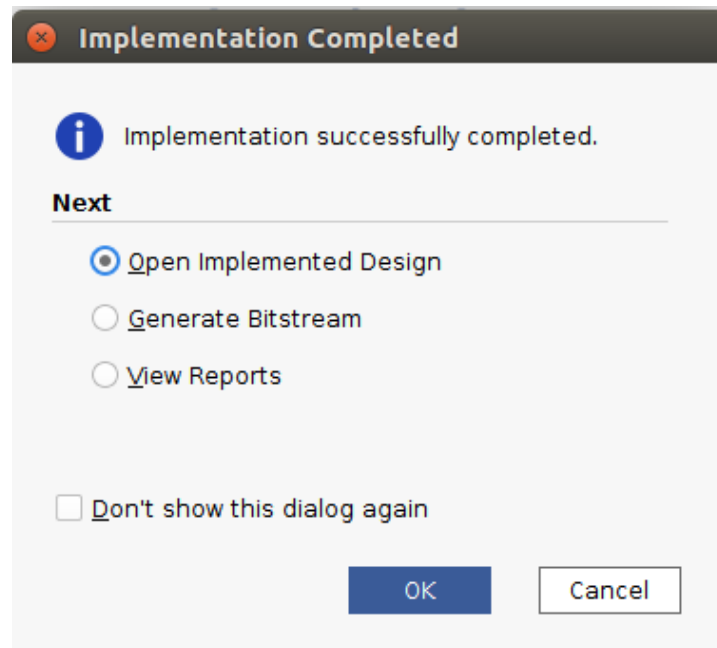


Constraints 소스를 추가해준다(포트정보 설정)



8.Run Implementation 을 한후 완료되면 Open Implemented Design 을 선택 후 OK

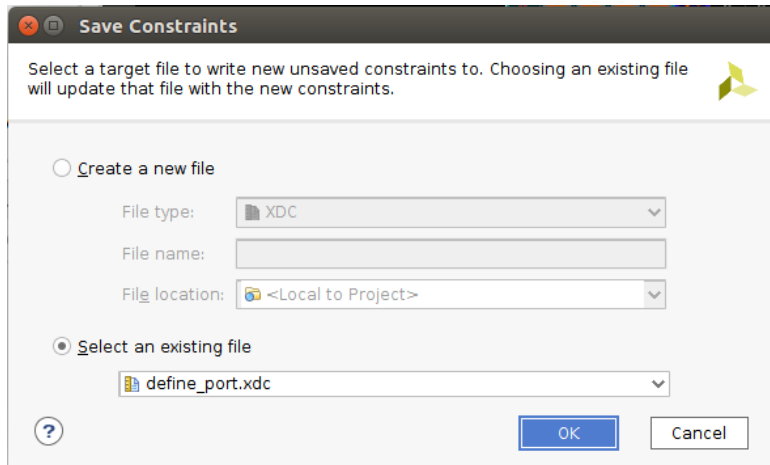
- ▼ IMPLEMENTATION
 - ▶ Run Implementation
 - > Open Implemented Design

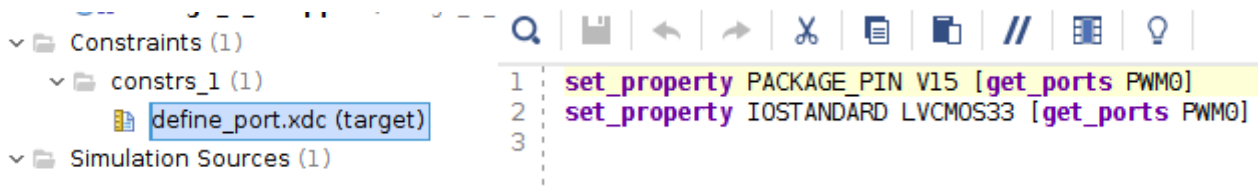


9.I/O Ports 에서 Scalar ports 밑에 PWM0 을 V15 핀 LVCMOS33 으로 맞춘 후 Save 한다.

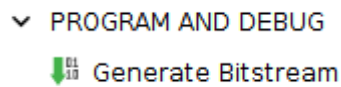
Tcl Console	Messages	Log	Reports	Design Runs	Timing	Methodology	Power	DRC	Package Pins	I/O Ports	x
<div><div><div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div></div></div></div>											
Name		Direction	Board Part Pin	Board Part Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std		
v All ports (131)											
DDR_16577 (71)		INOUT					✓	502	(Multiple)*		
FIXED_IO_16577 (59)		INOUT					✓	(Multiple)	(Multiple)*		
v Scalar ports (1)											
PWM0		OUT				V15	✓	34	LVCMOS33*		

Save 하면 Constraints 소스 파일에 저절로 포트 정보가 로드 된다.

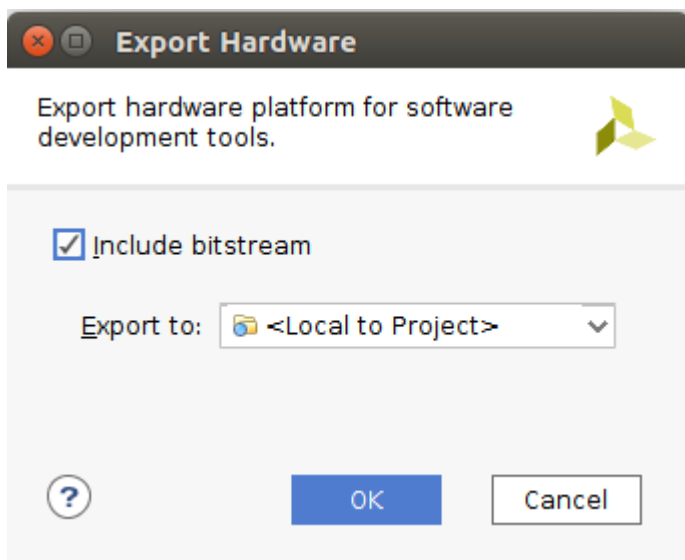




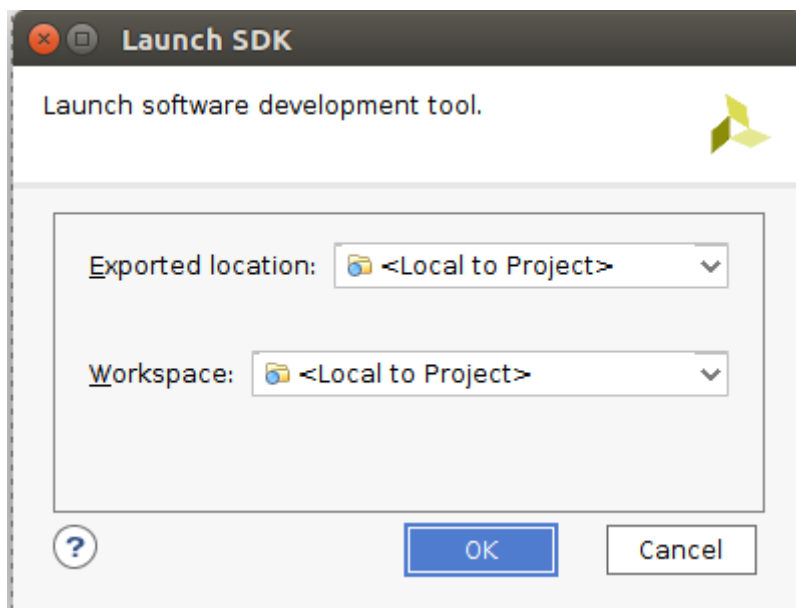
10. Generate Bitstream



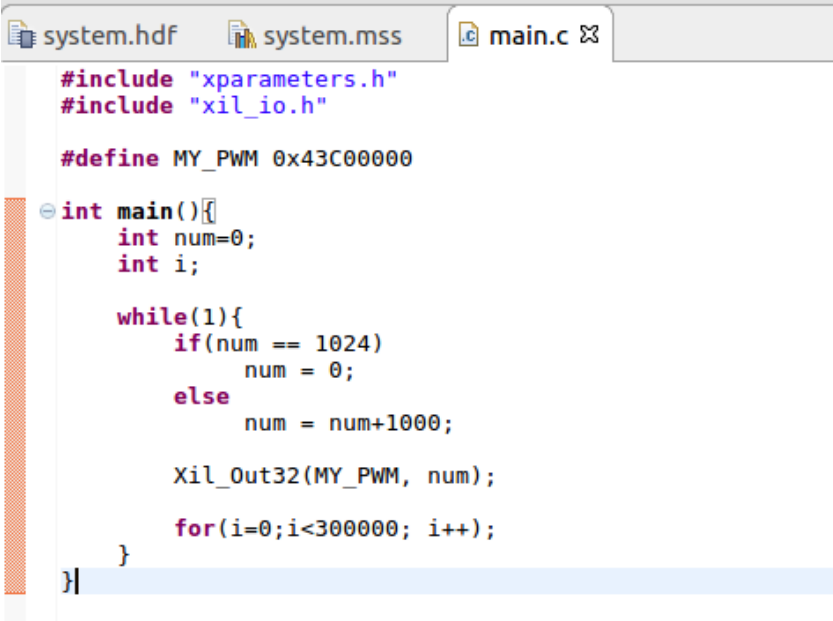
11. 비트스트림이 완료되면 file → Export Hardware -> include bitstream 체크 후 ok 를 눌러준다.



12. Launch SDK



SDK 에서는 New Application Project → empty project 로 생성한 후에
src 에 new file main.c 를 추가하여 아래와 같은 소스코드를 작성한다.



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

#define MY_PWM 0x43C00000

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

    while(1){
        if(num == 1024)
            num = 0;
        else
            num = num+1000;

        Xil_Out32(MY_PWM, num);

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

그 후 program FPGA 를 하고
application project 우클릭 후 Run as 에서 (GDB) 써있는거 선택해서 잘 돌아가는지 확인한다.

2. Make PWM device driver

이제 하드웨어 디자인은 위 과정에서 완료되었다.

이번에는 페타리눅스에서 디바이스를 다루는 과정을 진행하도록 한다.

프로젝트 생성할때는 hardware 폴더 밑에 생성하도록 한다.

만약 프로젝트를 위에 방식으로 만들었다면 경로 설정을 프로젝트명 → hardware 폴더 로 변경해서 작업을 진행하도록 한다.(그게 더 편리함)

나는 pwm custom ip 를 pwm_ip 폴더 아래 hardware 라는 폴더에 export 하였다.

1. petalinux-create -t project -n software --template
2. cd hardware/pwm_ip/pwm_ip.sdk/
3. petalinux-config --get-hw-description -p ../../software/
4. cd ~/pwm_ip/software/
5. petalinux-config -c u-boot
6. petalinux-build
7. petalinux-create -t apps -n device_driver --enable
8. cd components/apps/device_driver/
9. vi device_driver.c

이곳에다는 device_driver 소스코드를 작성한다. 소스코드는 아래와 같다.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/mman.h>
#include <fcntl.h>

#define IN      0
#define OUT    1

#define PWM_MAP_SIZE      0x10000

#define PWM_BASE_ADDR     0x43C00000

#define PWM_DATA_OFFSET   0x00
#define PWM_TRI_OFFSET    0x04

void usage(void){
    printf("**argv[0] -d <UIO_DEV_FILE> -i | -o <VALUE>\n");
    printf("  -d UIO device file - ex) /dev/uio0");
    printf("  -i Input from PWM\n");
    printf("  -o <VALUE> Output to PWM\n");
}

int main(int argc, char *argv[])
{
    int c, fd, value, direction = IN,i;
    int num=0;
    char *uiod;
    void *ptr;
    printf(" PWM UIO Test\n");
```

```

while( (c = getopt(argc, argv, "d:io:h")) != -1){
    switch(c){
        case 'd':
            uiod = optarg;
            break;
        case 'i':
            direction = IN;
        case 'o':
            direction = OUT;value = atoi(optarg);
            break;
        default :
            printf("Invalid Option: %c\n", (char)c);
            usage();
            return -1;
    }
}
fd = open(uiod, O_RDWR);
if(fd < 1){
    perror(argv[0]);
    printf("Onvalid UIO Device File: %s\n", uiod);
    usage();
    return -1;
}
ptr = mmap(NULL, PWM_MAP_SIZE, PROT_READ|
            PROT_WRITE,MAP_SHARED, fd, 0);

if(direction == IN){
    *((unsigned *)(ptr + PWM_TRI_OFFSET)) = 255;
    printf("%s:Input: %08x\n", argv[0], value);
}else{
    while(1)
    {
        if(num == 2000000)
            num = 0;
        else
            num = num + 10000;

        *((unsigned *)(ptr + PWM_TRI_OFFSET)) = 0;
        *((unsigned *)(ptr + PWM_DATA_OFFSET)) =num;// value;

        for(i=0; i<30000; i++);

        printf("%d,%x \n",num,ptr);
    }
}
munmap(ptr, PWM_MAP_SIZE);
return 0;
}

```

작성이 완료되었다면 다음 작업을 이어서 하도록 한다.

10. cd ~/pwm_ip/software/
11. petalinux-config -c rootfs
12. petalinux-config -c kernel
13. cd ../hardware/pwm_ip.sdk
14. petalinux-config --get-hw-description -p ../../software
15. cd ../../software/
16. petalinux-config
17. petalinux-config -c kernel

18. petalinux-config -c rootfs
19. cd subsystems/linux/configs/device-tree/
20. vi system-top.dts

디바이스 트리를 아래와같이 작성한다.(ip 이름도 똑같이 만들었다면 필자와 동일하다. 하지만 ip 이름이 다르면 address editor 에서 확인해 적어야한다)

```
/dts-v1/;
/include/ "system-conf.dtsi"
/ {
};
&clk0 {
    ps-clk-frequency = <50000000>;
};
&flash0{
    compatible = "s25fl128s1";
};
&usb0{
    dr_mode = "otg";
};
&gem0{
    phy-handle = <&phy0>;
    mdio{
#address-cells = <1>;#size-cells = <0>;
phy0: phy@1{
        compatible = "realtek,RTL8211E";
        device_type = "ethernet-phy";
        reg = <1>;
    };
};

&My_PWM_Core_0 {
    compatible = "generic-uio";
};
```

21. cd ~/pwm_ip/software/images/linux/
22. petalinux-build
23. source ~/xilinx/Vivado/2017.1/settings64.sh
24. source ~/xilinx/SDK/2017.1/settings64.sh
25. petalinux-package --boot --fsbl zynq_fsbl.elf --fpga
../././hardware/pwm_ip.runs/impl_1/design_1_wrapper.bit --uboot --force

이제 디바이스드라이버를 올리고 확인하면된다.