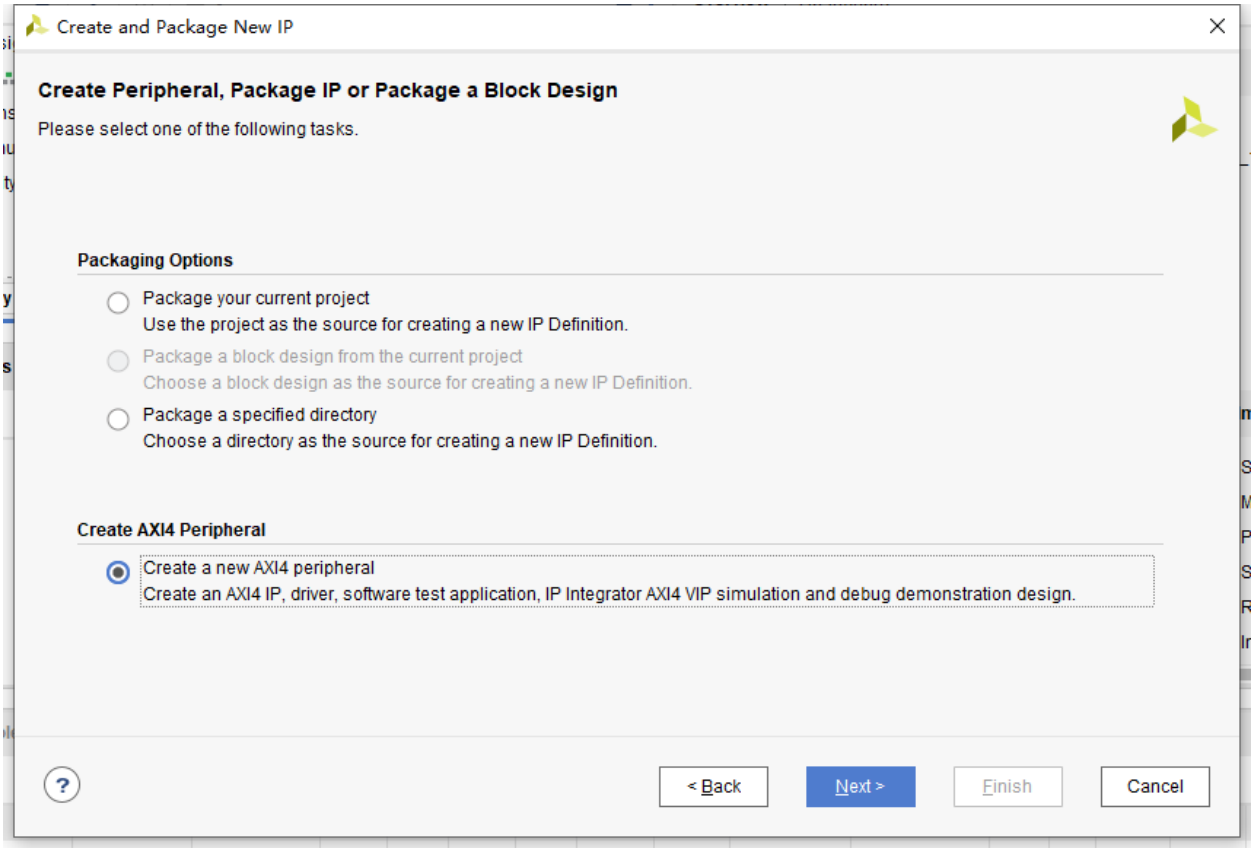
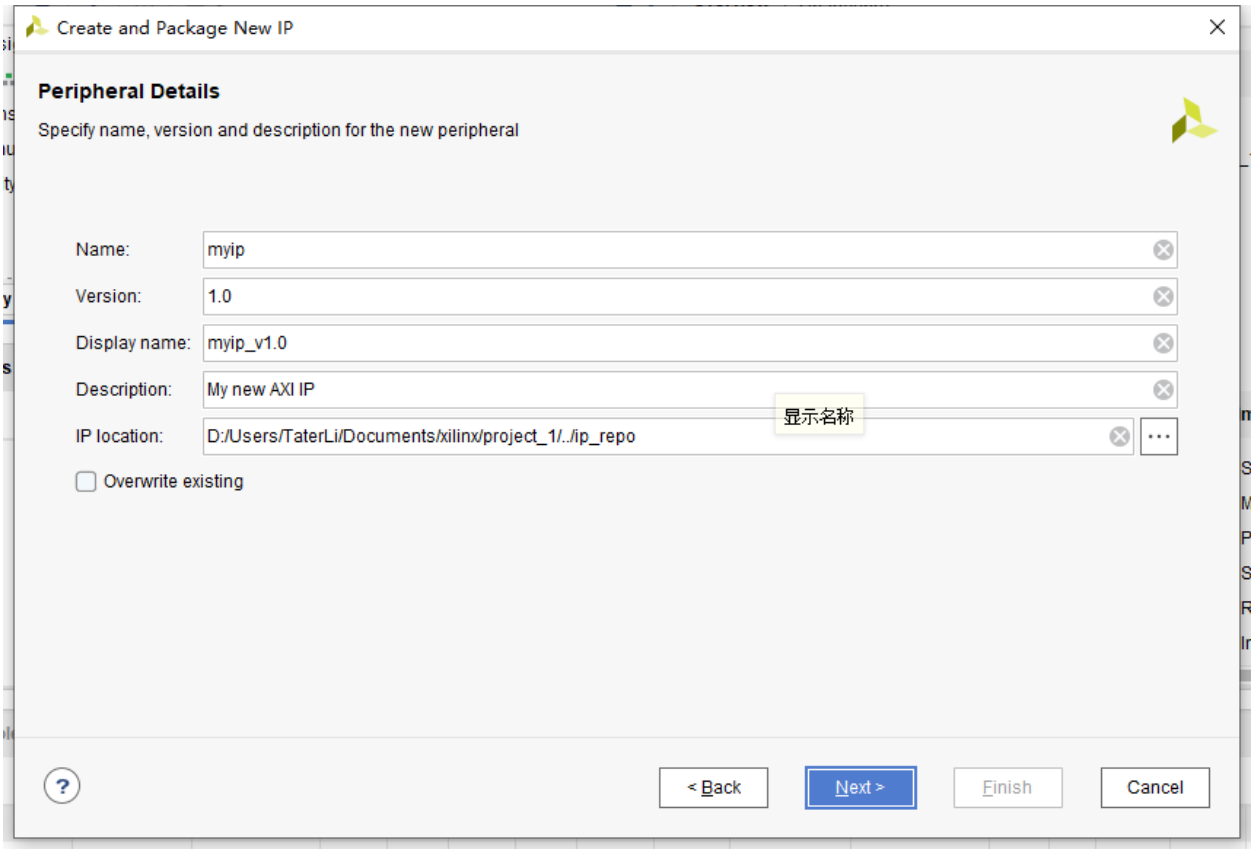
# [L11]设计一个外设 + 仿真测试

之前有学习到AXI,不过实际设计一个外设的时候,不用实现AXI这个轮子,只需要在轮子上做自己东西就足够了,因为之前已经大致了解AXI协议了,所以现在改起代码来应该得心应手了.

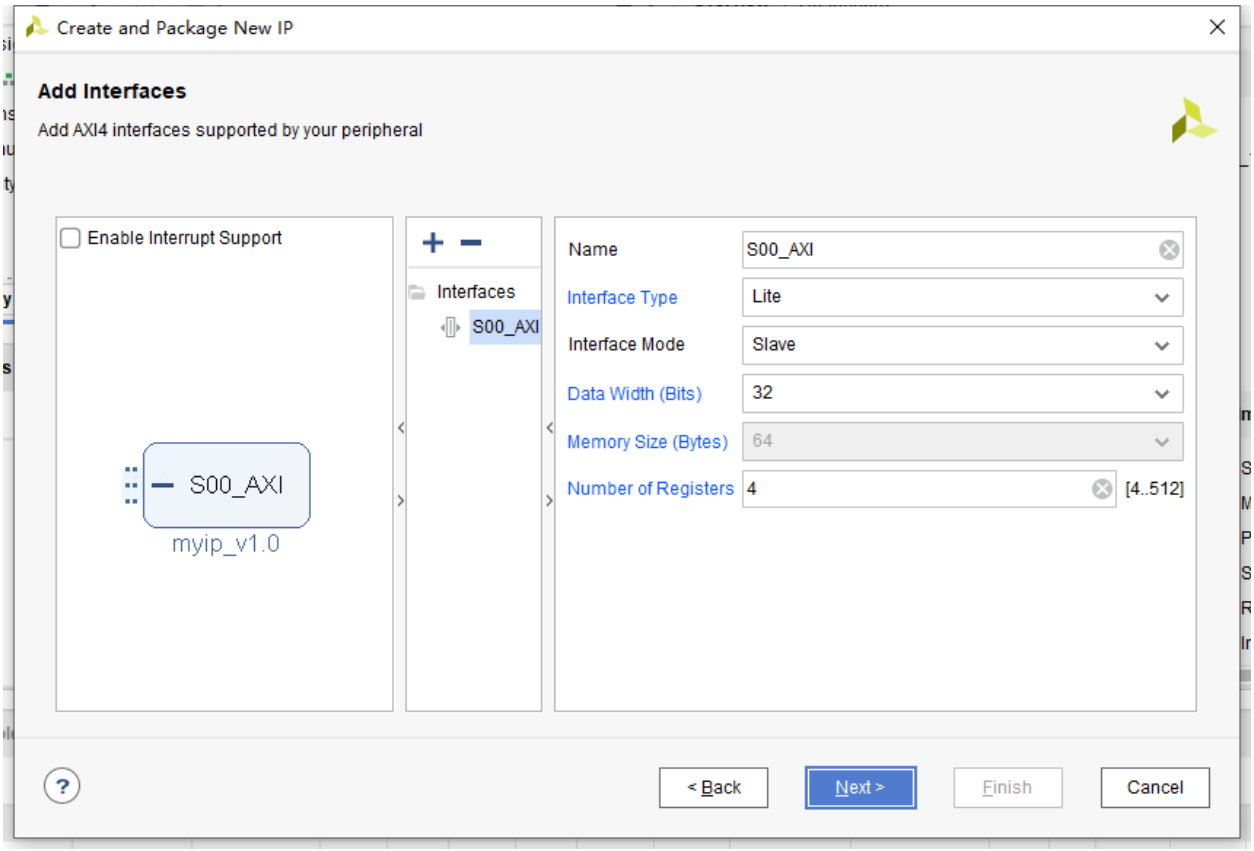
进入Tools→Create and Package New IP菜单中创建一个IP.



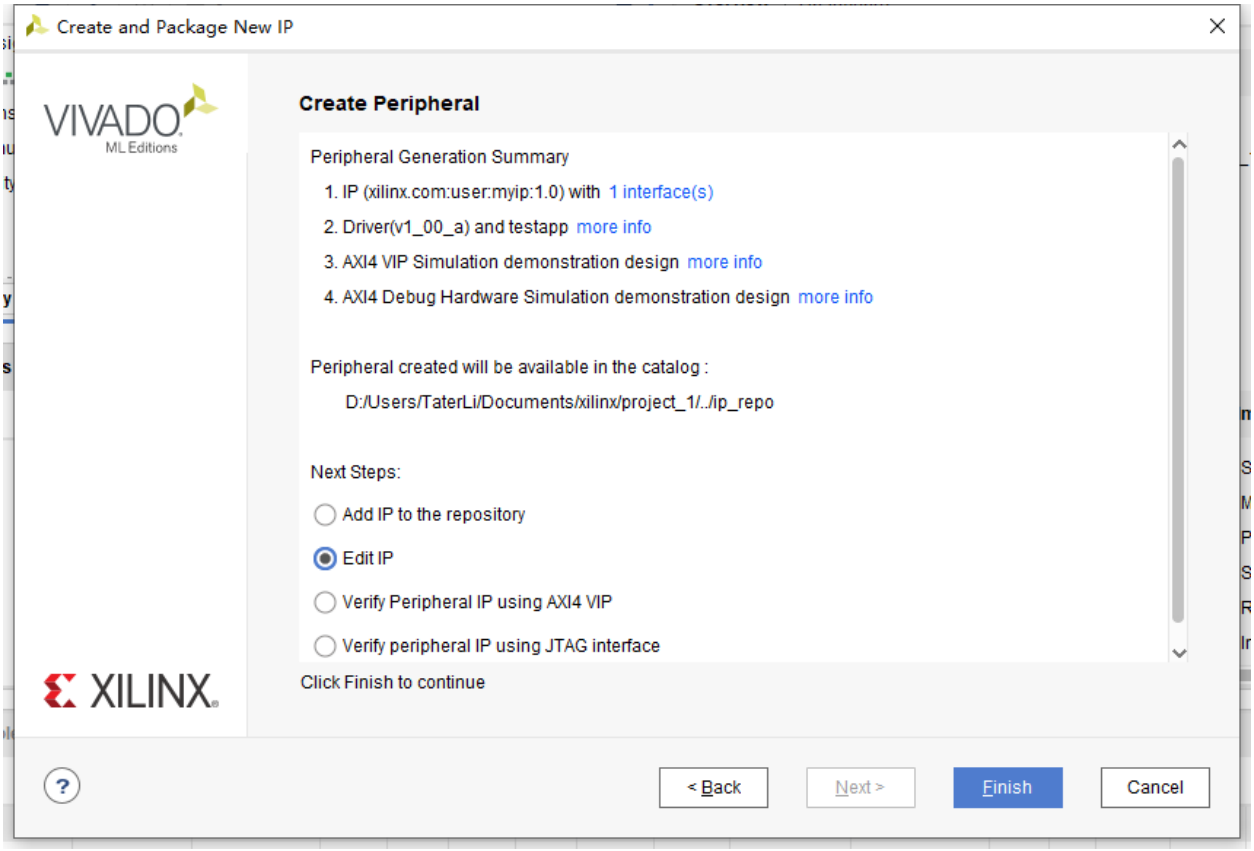

信息就随便了.


Create and Package New IP
Peripheral Details
Specify name, version and description for the new peripheral
Name:
myip
Version:
1.0
Display name:
myip_v1.0
Description:
My new AXI IP
显示名称
IP location:
D:/Users/TaterLi/Documents/xilinx/project_1/../ip_repo
Overwrite existing
< Back
Next >
Finish
Cancel

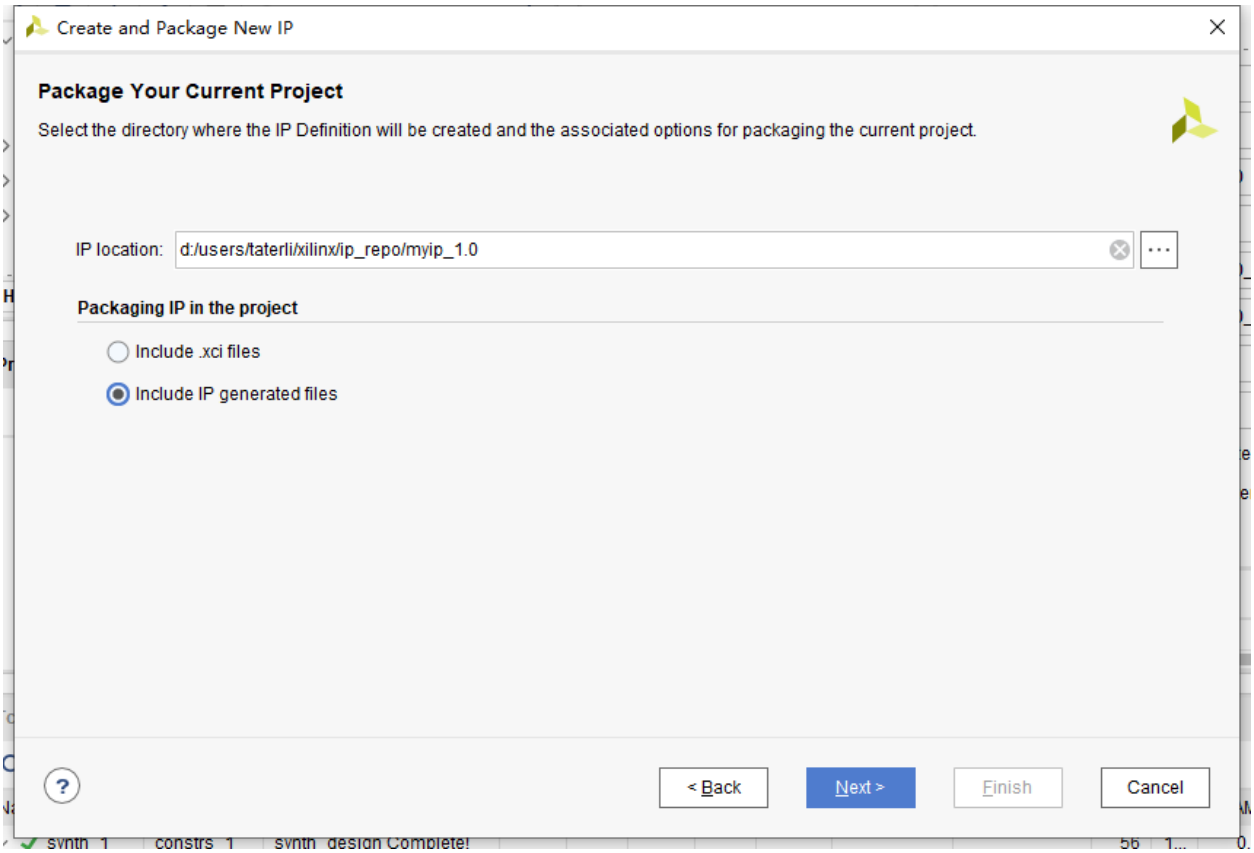
寄存器依然保持4个.



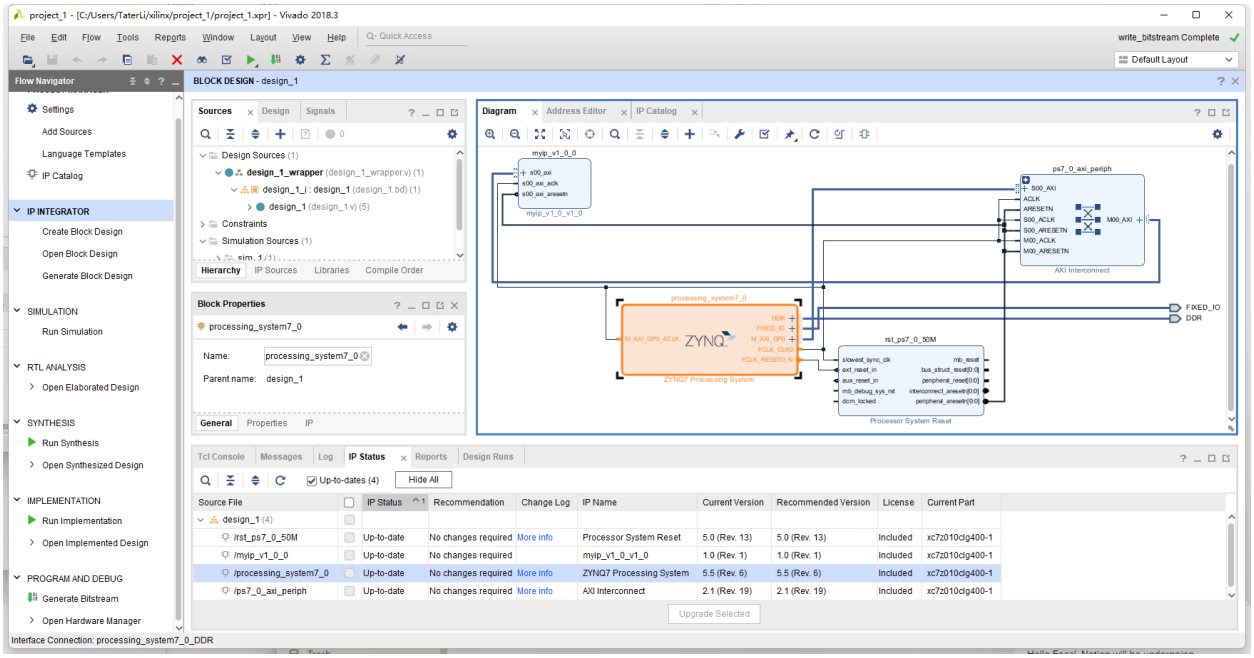

然后进入IP编辑看看这个IP.



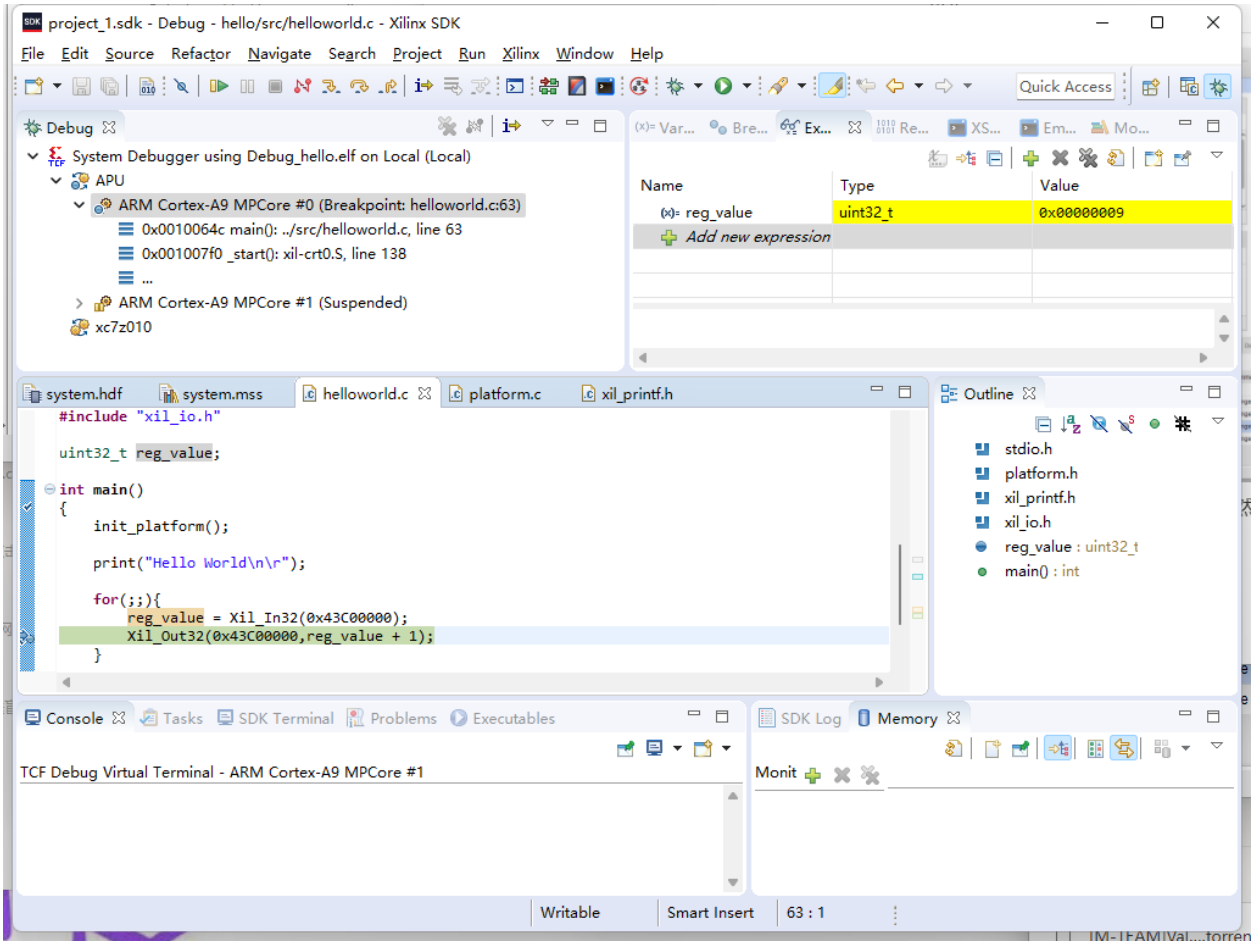

他会另起一个Vivado,这里就是IP编辑了,大致浏览下源码就知道是AXI操作的,不过这里先不讨论,先尝试打包IP,打包过程选择Include IP generated files.



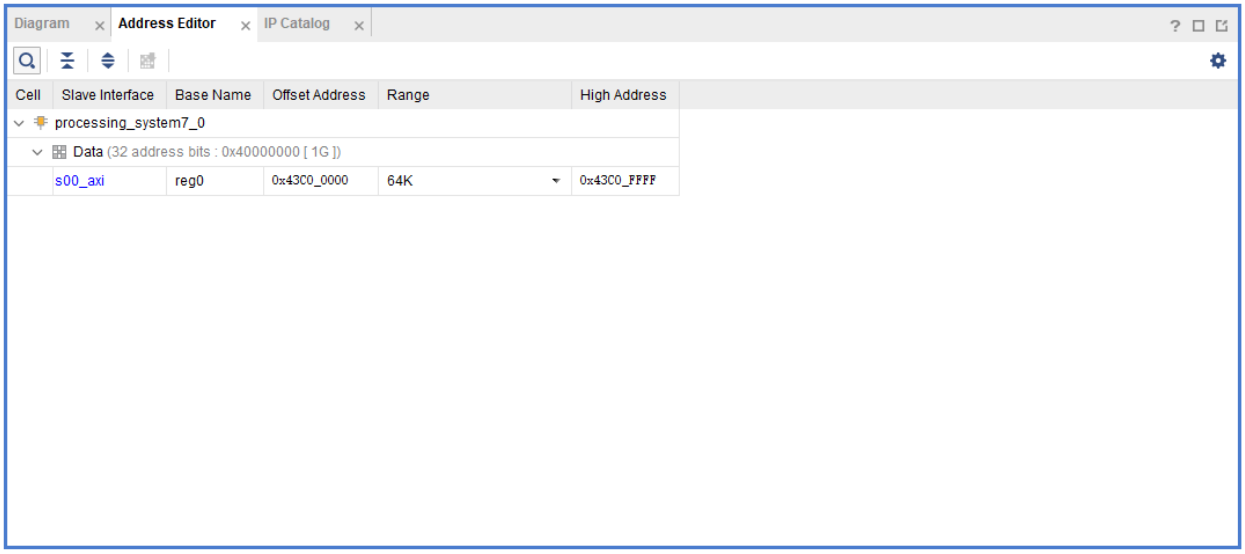

然后简单地把他们链接一起,如果IP写着Pre-Product可能无法运行,那就原封不动重新打包几次,可能是BUG.



然后就生成各种文件,包括bitstream,然后启动SDK,并且下载bitstream,调试测试.




好的,我们现在有了一个空外设了,位于0x43C00000地址,这个地址在Address Editor看得到.




不过我们现在明显还没实现功能嘛,我们现在做一个PWM外设,做之前我们要规划一下寄存器.

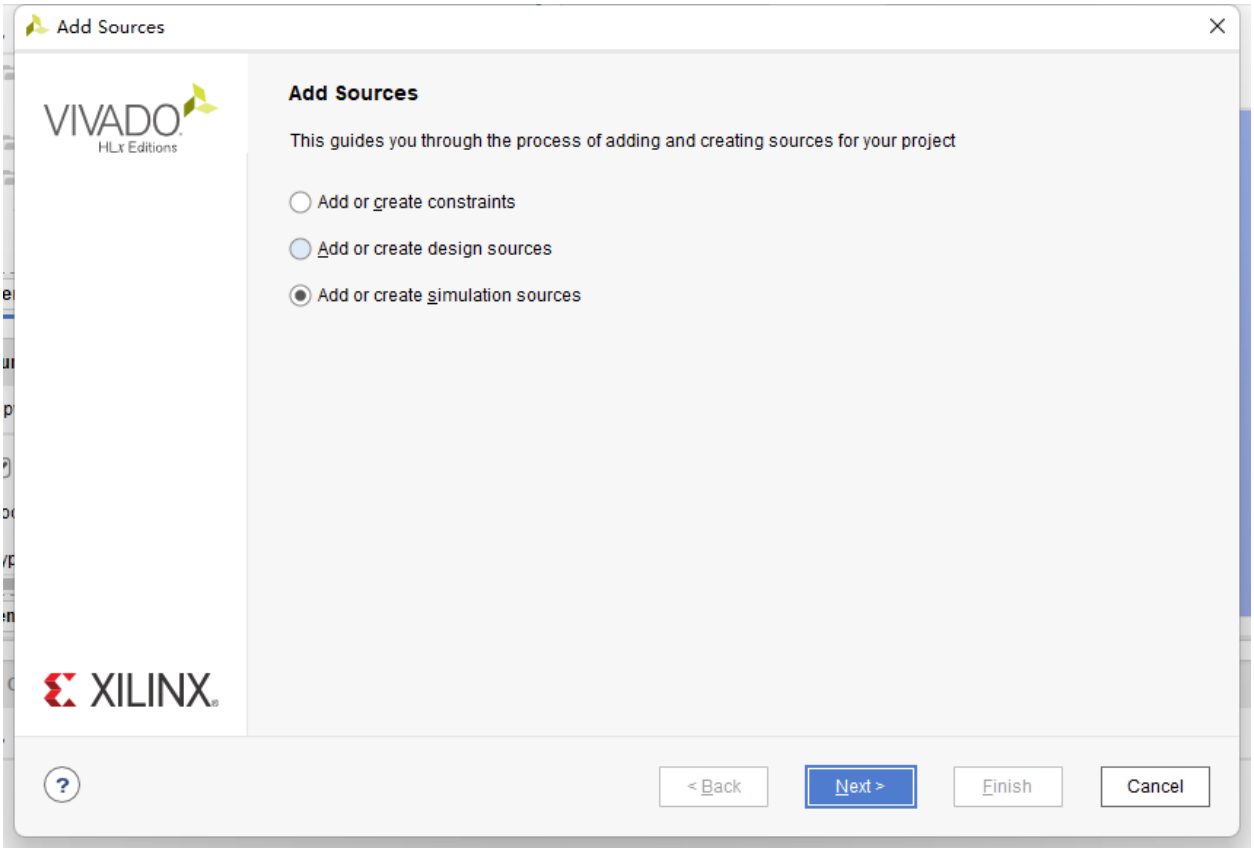
1. 定时器时基 REG_OFFSET 0x0 - RW
2. 定时器比较数 REG_OFFSET 0x04 - RW
3. 定时器使能 REG_OFFSET 0x08 - RW
4. 外设ID REG_OFFSET 0x10 - RO

再开一份Vivado来设计下IP的实际内部代码,主要是为了方便调试.

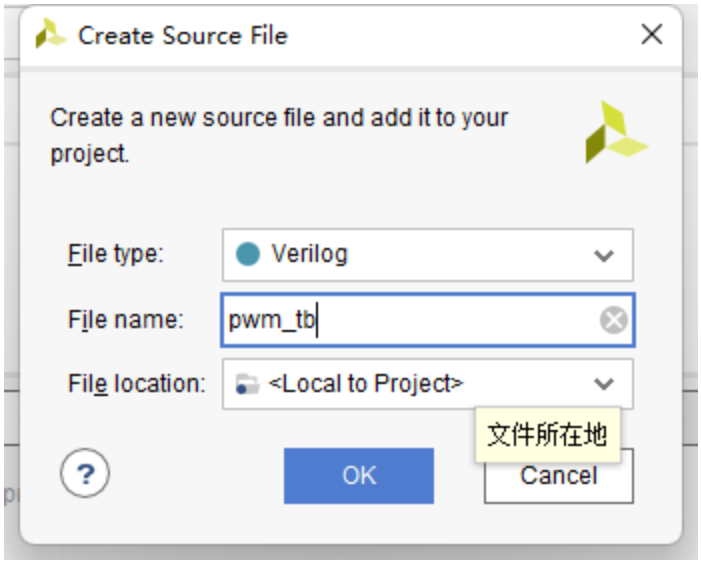
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2022/02/28 11:42:42
// Design Name:
// Module Name: pwm
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
```

```
//////////////////////////////////////////////////////////////////////////////////


module pwm(input wire clk,
    input wire rst,
    output reg pwm,
    input wire[31:0] base,
    input wire [31:0] ccr,
    input wire en);

reg [31:0] cnt;

always@(posedge clk or negedge rst)
begin
    if(!rst)
    begin
        cnt <= 'b0;
    end
    else
    begin
        if(cnt > base)
        begin
            cnt <= 'b0;
        end
        else
        begin
            cnt <= cnt + 'b1;
        end
    end
end

always@(posedge clk or negedge en)
begin
    if(!en)
        begin
            pwm <= 'b0;
        end
    else
    begin
        if (cnt > ccr)
        begin
            pwm <= 'b0;
        end
        else
        begin
            pwm <= 'b1;
        end
    end
end

endmodule
```

编写测试代码,先创建模拟源.




名字以_tb结尾.

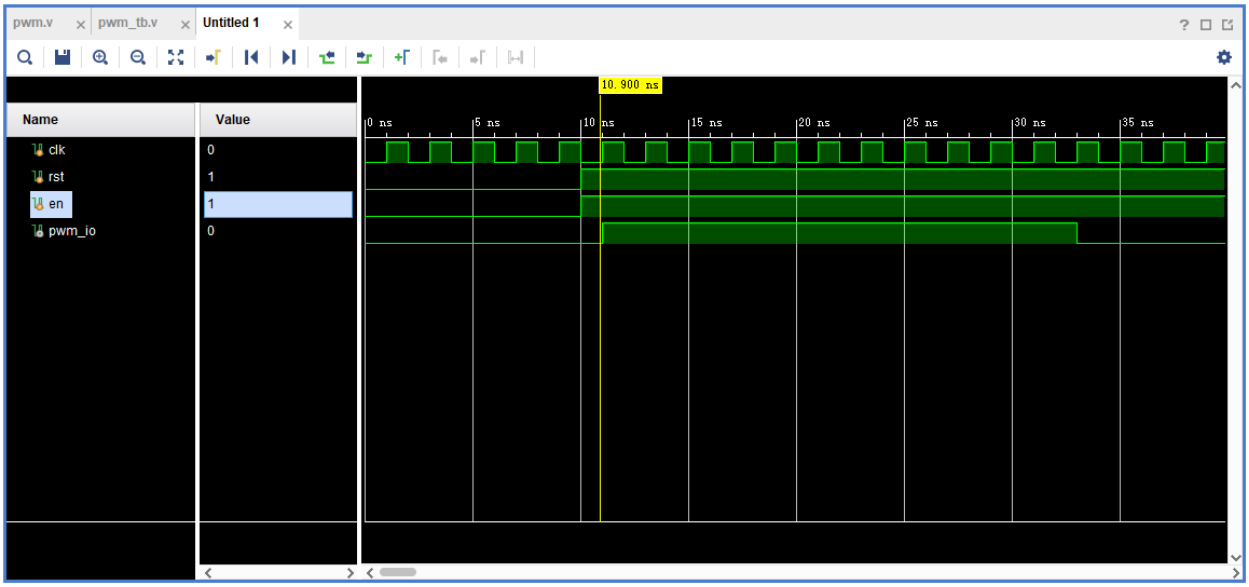


写入测试代码.

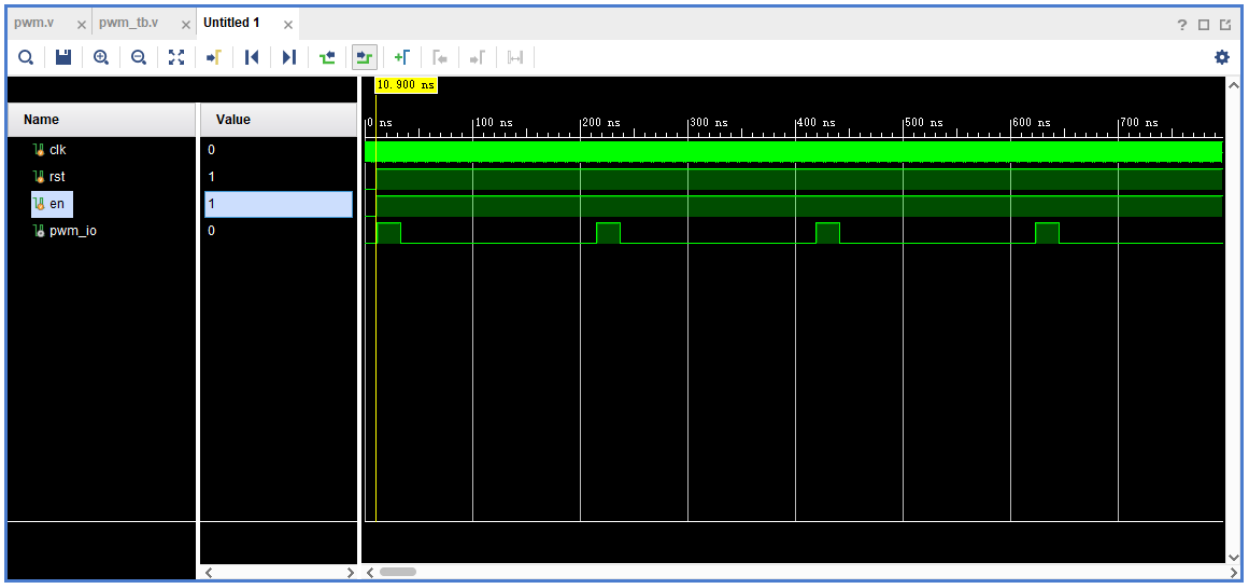
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2022/02/28 12:44:43
// Design Name:
// Module Name: pwm_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module pwm_tb();

reg clk;
reg rst;
reg en;

wire pwm_io;

always #1 clk = ~clk;

initial begin
clk = 0;
rst = 0;
en = 0;
#10
rst = 1;
en = 1;
end

pwm pwm_inst(
.clk(clk),
.rst(rst),
.pwm(pwm_io),
.base(100),
.ccr(10),
.en(en)
);
```

```
endmodule
```

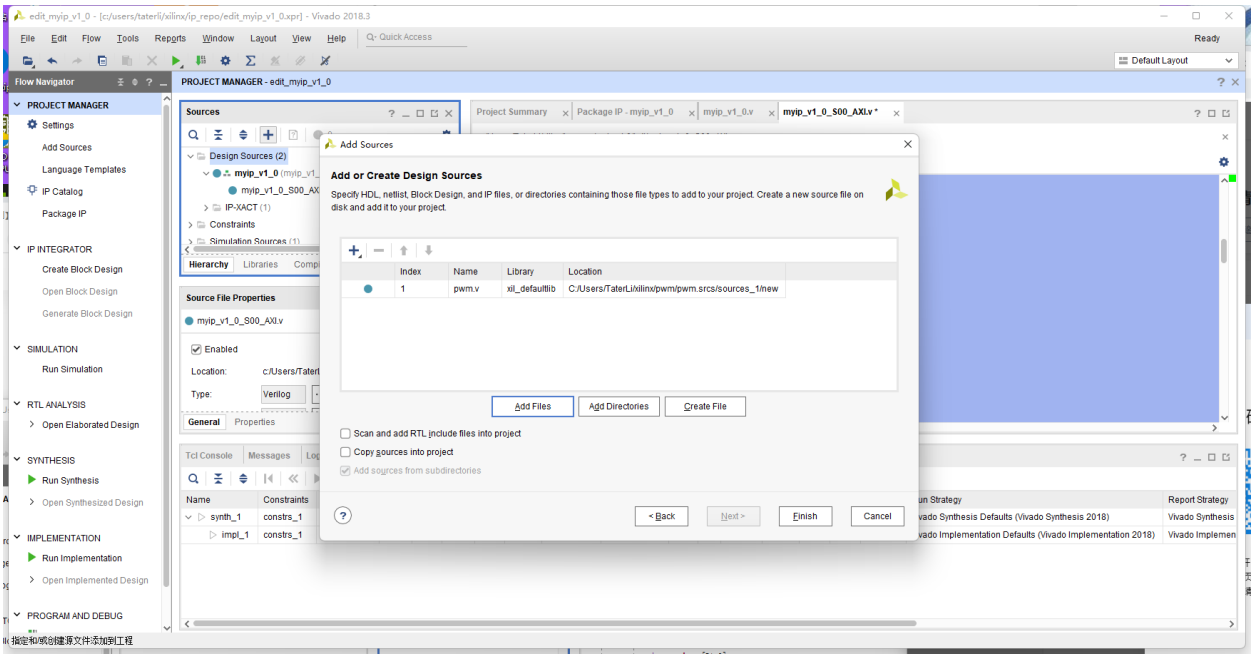
可以看到,在CCR=10时候已经拉低.



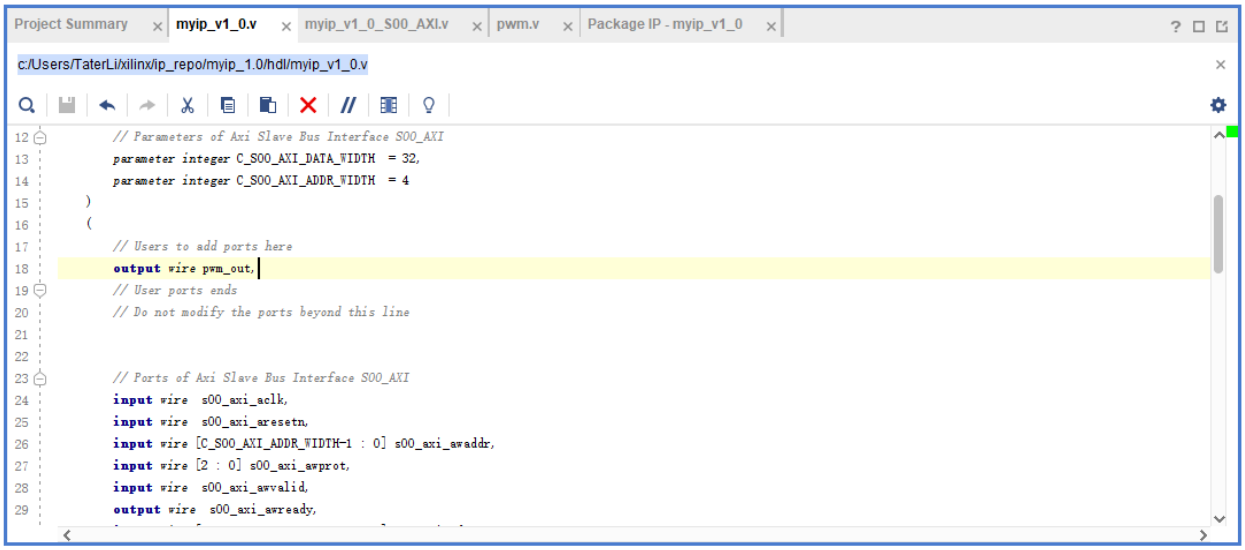

PWM占空比10%,也很正常.



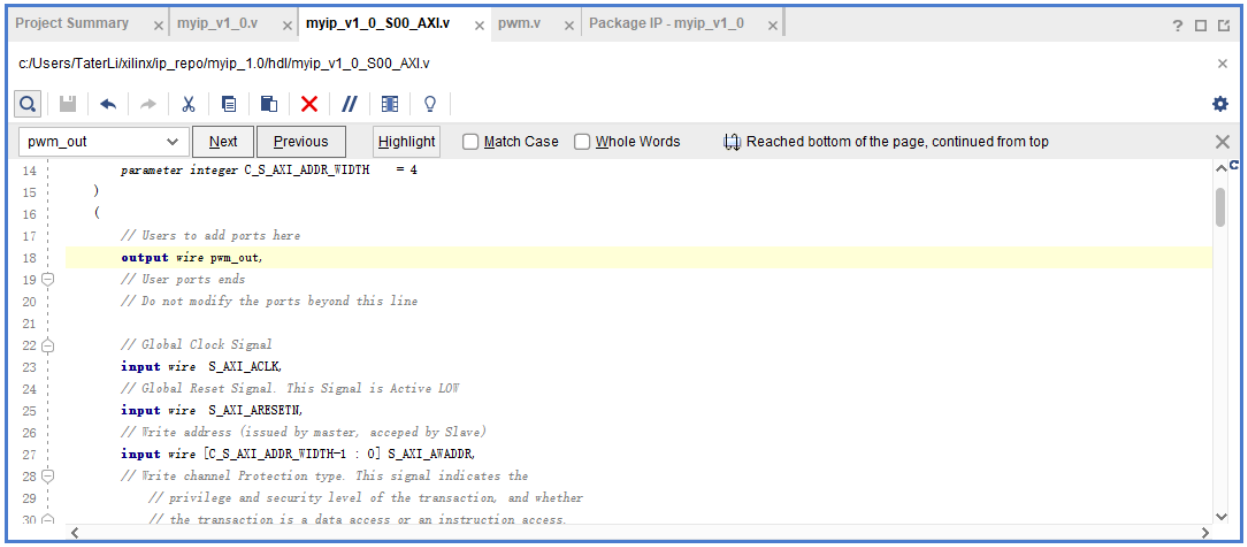

回到IP编辑,然后把PWM添加进来.



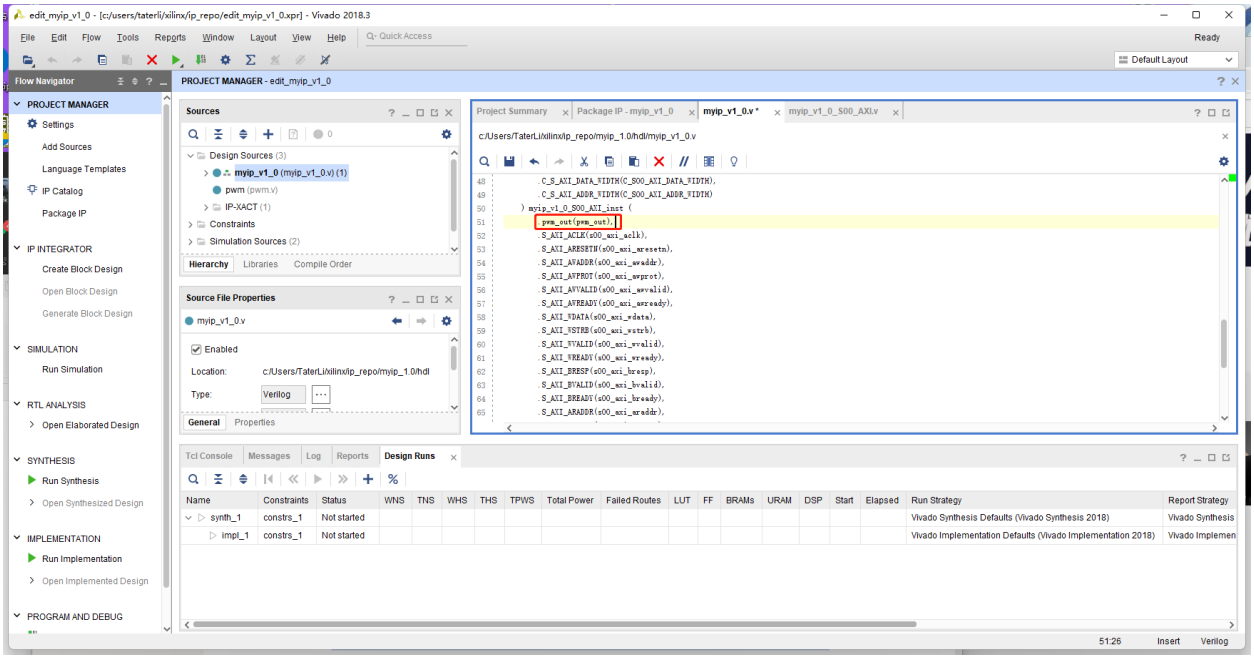

在IP顶部添加引脚输出.



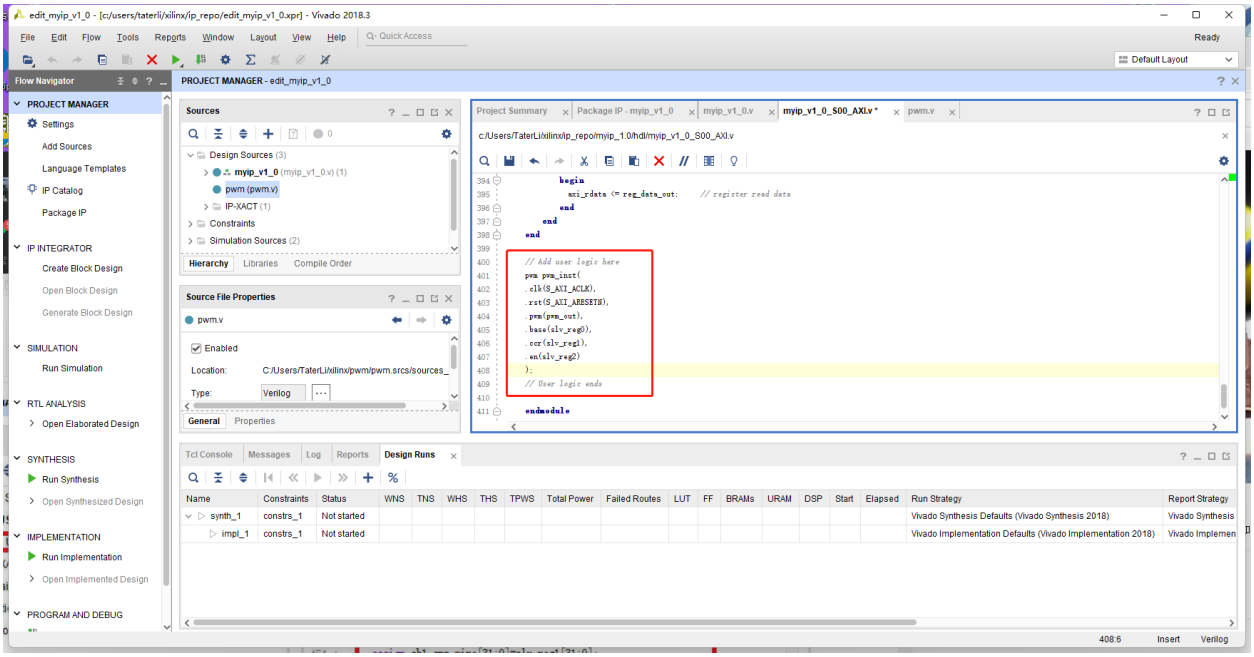

在这里也要声明出来.



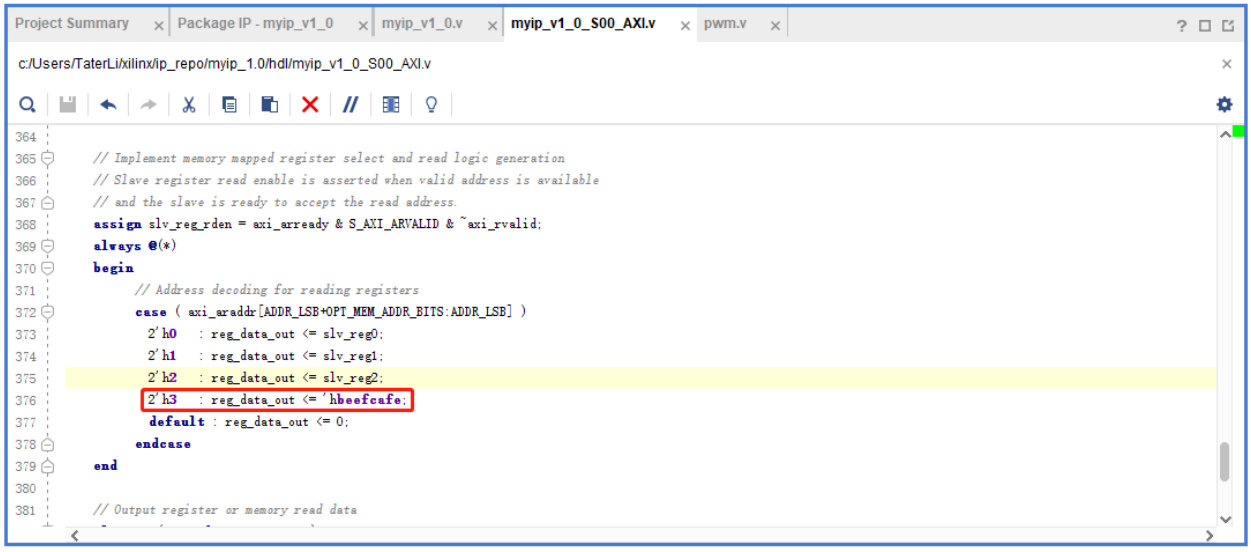

当然还要从顶层传递过去.



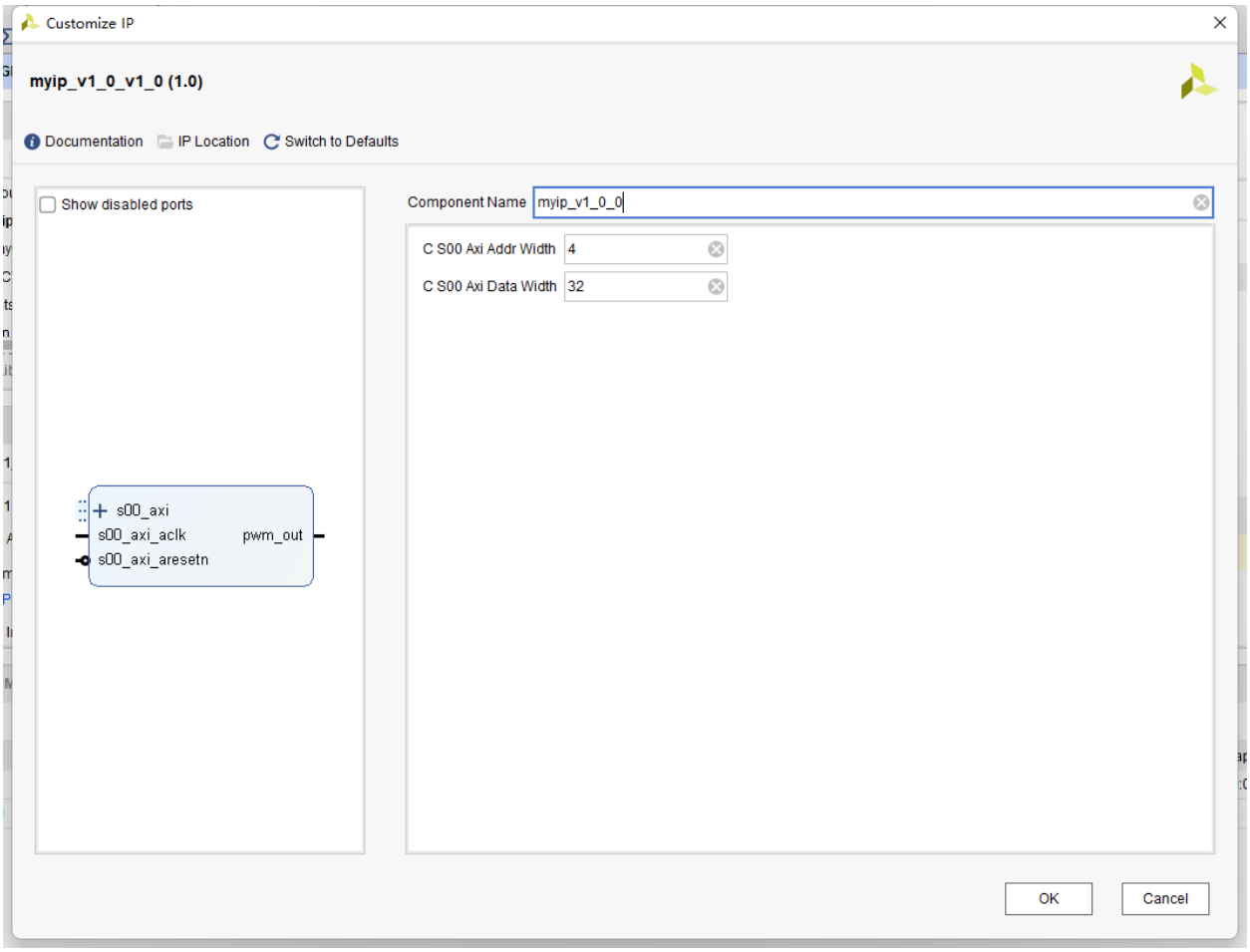

按照正常思路,那么要把实例化一下pwm模块.



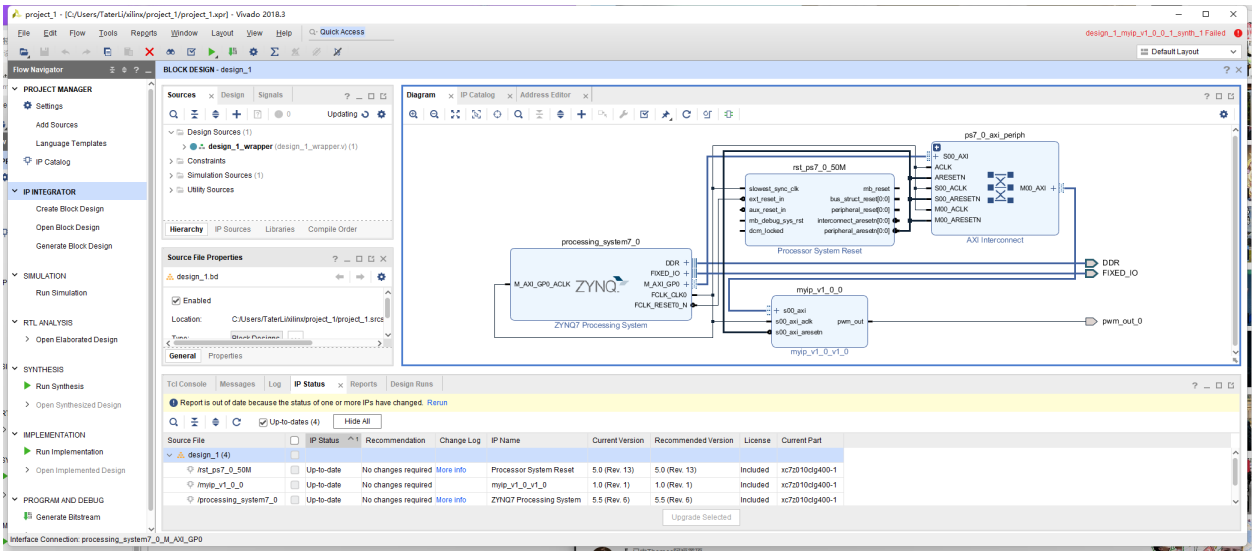
我们不是设计一个外设ID么,就直接改那个对应位置.




然后尝试打包逐步往上,确定模块长这个样子.(有个输出端口,有时候有BUG没刷出来.)


Customize IP
myip_v1_0_v1_0 (1.0)
Documentation
IP Location
Switch to Defaults
Show disabled ports
Component Name
myip_v1_0_0
C S00 Axi Addr Width
4
C S00 Axi Data Width
32
s00_axi
s00_axi_aclk
s00_axi_aresetn
pwm_out
OK
Cancel

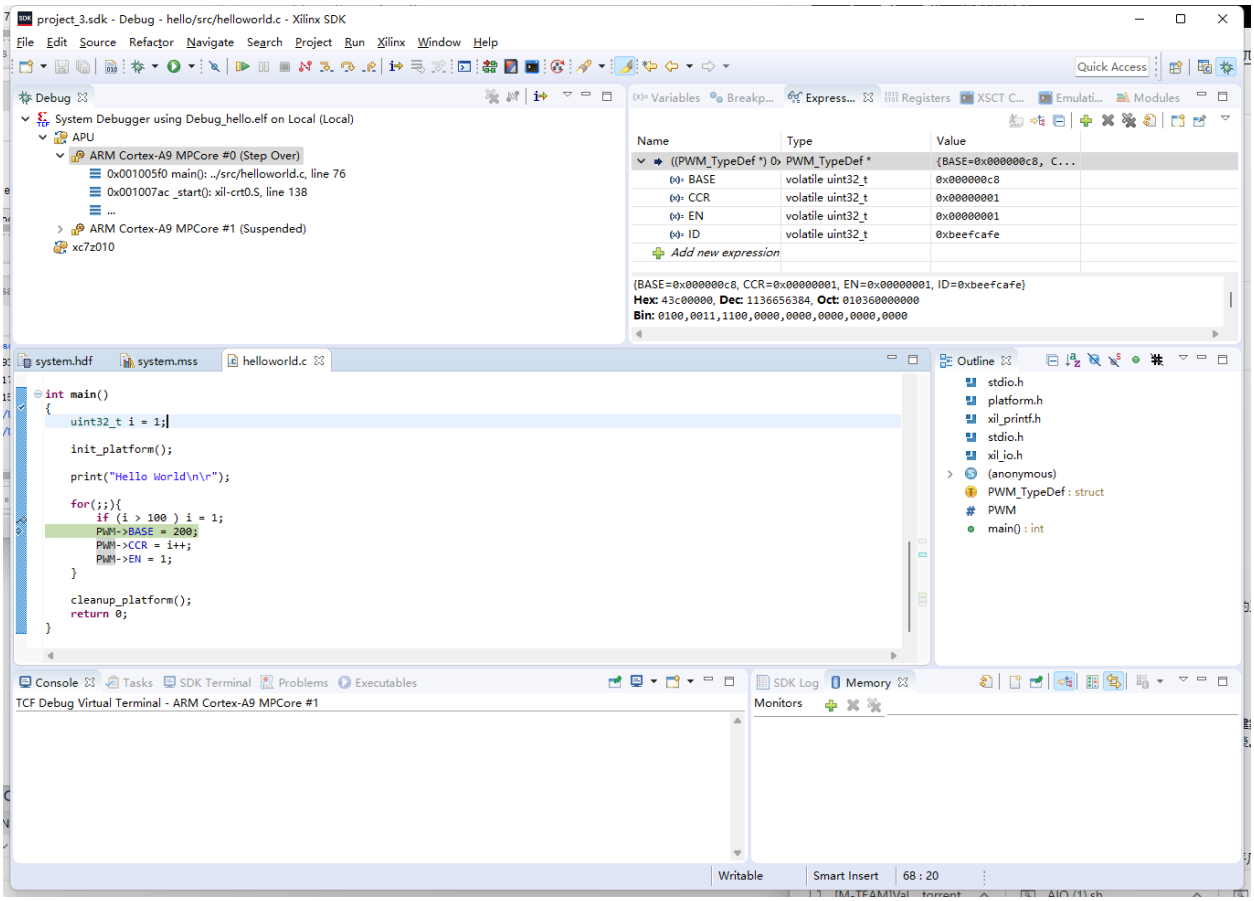
更新后记得导出引脚.



最后写个简单的测试代码.

```
/******************************************************************************
*
* Copyright (C) 2009 - 2014 Xilinx, Inc.  All rights reserved.
*
* Permission is hereby granted, free of charge, to any person obtaining a copy
* of this software and associated documentation files (the "Software"), to deal
* in the Software without restriction, including without limitation the rights
* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
* copies of the Software, and to permit persons to whom the Software is
* furnished to do so, subject to the following conditions:
*
* The above copyright notice and this permission notice shall be included in
* all copies or substantial portions of the Software.
*
* Use of the Software is limited solely to applications:
* (a) running on a Xilinx device, or
* (b) that interact with a Xilinx device through a bus or interconnect.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
* XILINX  BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
* WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF
* OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*
* Except as contained in this notice, the name of the Xilinx shall not be used
* in advertising or otherwise to promote the sale, use or other dealings in
* this Software without prior written authorization from Xilinx.
*
******************************************************************************/

/*
 * helloworld.c: simple test application
 *
 * This application configures UART 16550 to baud rate 9600.
 * PS7 UART (Zynq) is not initialized by this application, since
 * bootrom/bsp configures it to baud rate 115200
 *
 * ------------------------------------------------
 * | UART TYPE   BAUD RATE                        |
 * ------------------------------------------------
 *   uartns550   9600
 *   uartlite    Configurable only in HW design
 *   ps7_uart    115200 (configured by bootrom/bsp)
 */

#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
```

```c
#include <stdio.h>
#include "xil_io.h"


typedef struct
{
  volatile uint32_t BASE;
  volatile uint32_t CCR;
  volatile uint32_t EN;
  volatile uint32_t ID;
} PWM_TypeDef;

#define PWM ((PWM_TypeDef *) 0x43C00000)

int main()
{
  uint32_t i = 1;

    init_platform();

    print("Hello World\n\r");

    for(;;){
      if (i > 100 ) i = 1;
        PWM->BASE = 200;
        PWM->CCR = i++;
        PWM->EN = 1;
    }

    cleanup_platform();
    return 0;
}
```




另外其实很多基础IP都是预先设计好的,比如AXI GPIO/XADC等等,我们只要拖进来分配地址,像外设一样初始化他,之后便可以当普通GPIO/ADC等正常使用,其他外设也类似,具体看外设手册.