Getting Started with the Zynq-7000 family using EBAZ4205 as a development board

Before I get started, I would like mention that most of my information (about 95%) on the EBAZ4205 was obtained from github repositories, personal blogs, AliExpress sellers, torrents, Xilinx website, user sites and reading up on various discussions that we had on the EBAZ4205 Telegram group. I would like to express my gratitude here. All the reference items used to make this document have been listed at the end of this document, be sure to check out the sources listed as they are the truly authors! I only collected in one document all the information available and updated just a few things.

Quick description of the parameters of the board:

- Xilinx ZYNQ 7010 (XC7Z010-CLG400-1) as the main control.
- There are three versions of DRAM (as far as I know), which are Etron EM6GD16EWKG-12H (DDR3 SDRAM 2G-Bit 128Mx16 1.5V 96-Pin F-BGA), Micron MT41K128M16 (screen printing keyword: D9MPTK) and UniIC SCB15H2G160AF-13K. They are both 256MiB of DDR3 (the latter is DDR3L).
- The flash memory uses Winbond W29N01HV, a 128MiB×8bit NAND.
- The single-port Fast Ethernet chip uses WIZnet IP101GA, LQFP-48 package, and 3.3V power supply.
- The overall power supply of the motherboard uses 12V DC, but someone on the forum seems to be able to run it with 5V; the interface type used for power supply is Molex 5557 series 6pin.
- The finished board also comes with 1 channel PS debugging serial port, standard 14pin JTAG debugging port supporting XILINX downloader, 2 channels PWM output, uSD card, two LEDs, two buttons (one is not soldered), and 14*3 PL-side IO interfaces. For the AXI GPIO on the PL side, some merchants have already made a matching adapter board, providing RS232, VGA, OV7670 and other expansion interfaces, but the price is not cheap, if you need your own drawing board is also a good choice.

There are some different models, also with different memory chips, but I realized there are two different variants of the EBAZ4205. The variants are easily identifiable by the components mounted on each other. One, has the optocouplers for the FAN connectors mounted and has the crystal for PHY Ethernet chip mounted. The another one, called by someone “the crystal less variant” don’t have the optocouplers mounted and don’t have the PHY Ethernet crystal mounted.

Photos of the board without modifications (crystal less variant):

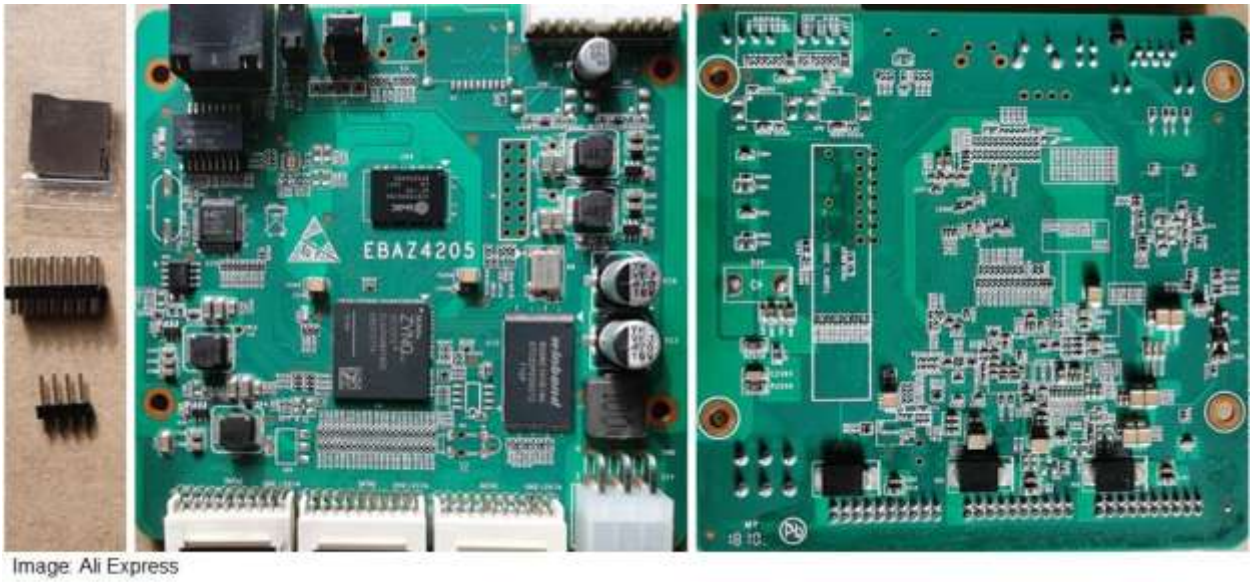


Figure 1: PCB Assembly without mod

Photos of the board with modifications (crystal variant):

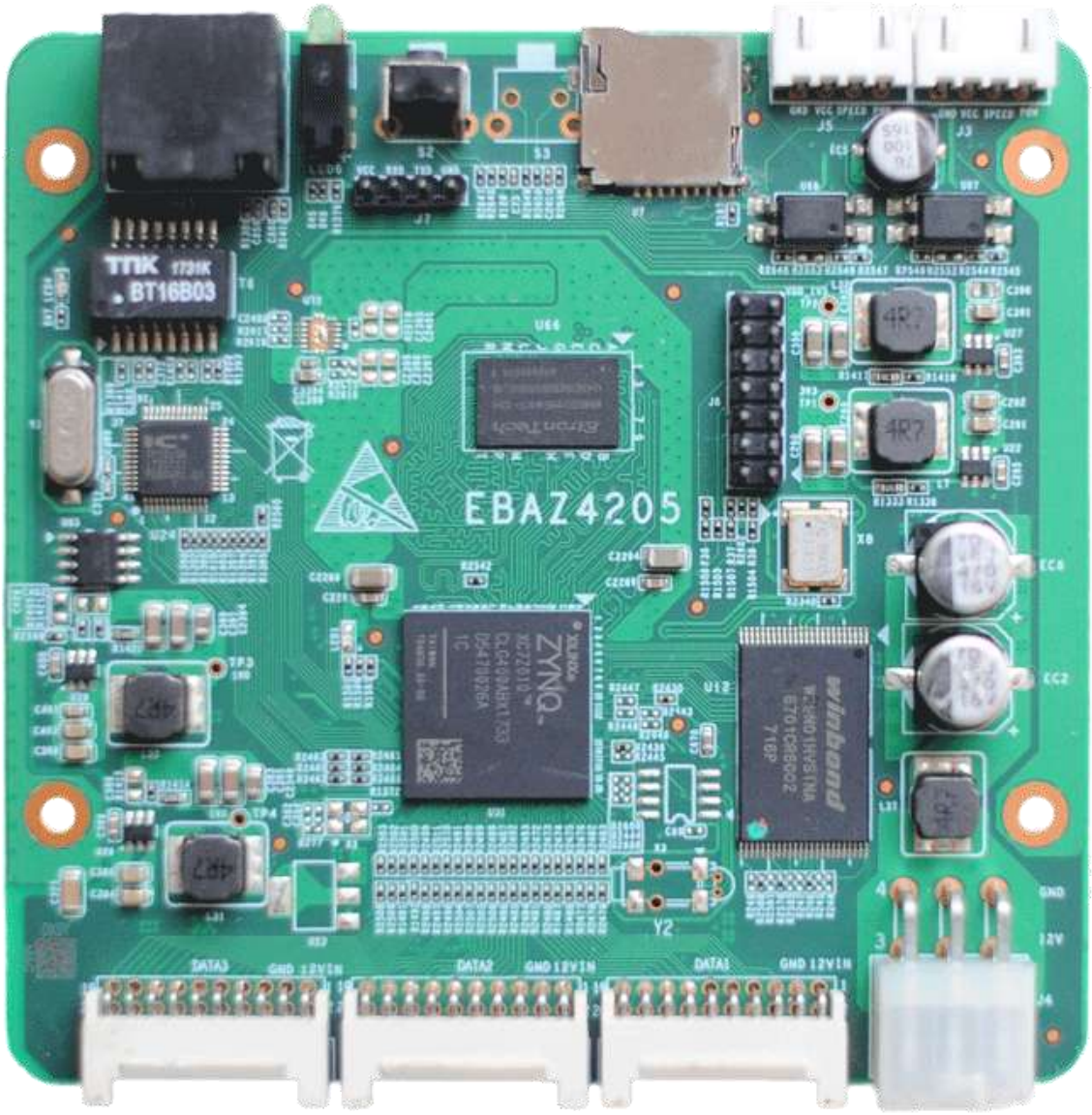


Figure 2: EBAZ4205 assembled top view

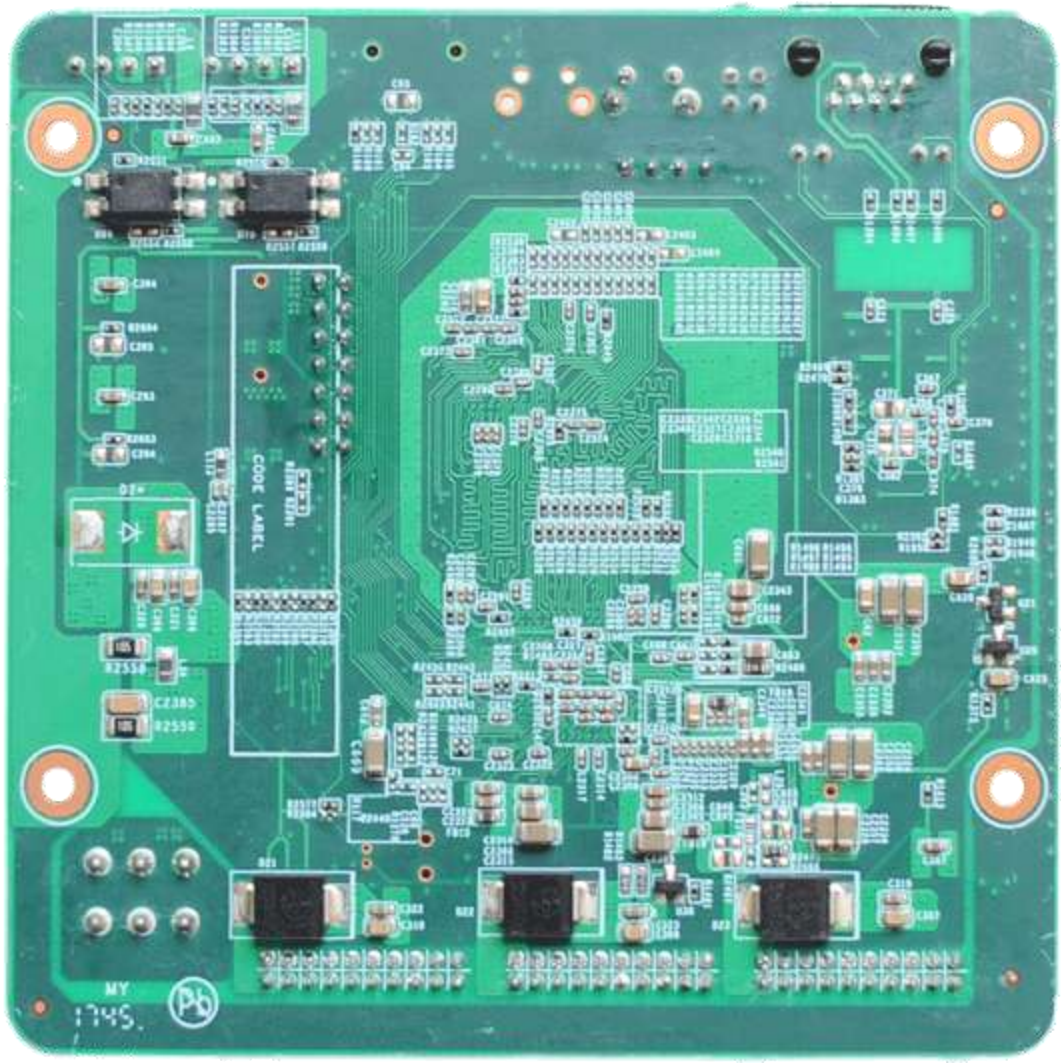


Figure 3: EBAZ4205 assembled bottom view

A bit of history

Looking for a cheap ZYNQ development board, I found a Reddit post (https://www.reddit.com/r/FPGA/comments/jvjskn/anyone_have_experience_with_chinese_zynq_boards/) that talked of some Zynq-7000 boards out of China that were from decommissioned Bitcoin mining rigs that were going for the cheap, around $10. If you compare these PCBs with the price of the Zynq alone, this is really really cheap!

This development board was the control card of Ebang Ebit E9+ BTC miner. (https://www.ebang.com.cn/about) They could be reconfigured as development boards with a few minor modifications.

Figure 4: Ebit E9+ BTC miner

The EBAZ4205 was not designed as a development board, and as such, there isn't an official documentation to refer to when trying to get started.

Although there are already multiple posts mentioning the schematic and hardware in use, most of them lack details on the FPGA and GNU/Linux point of view. In this document, I will try to test its functionality, document problems or pitfalls (if any) encountered during the hardware "bring-up", and provide some form of BSP (Board Support Package) for the board so that it can be used in any project.

There was another model, EBAZ4203, which was very similar, except that 4205 is an updated version of 4203.

Difference between EBAZ4203 and EBAZ4205

Chip type	4205	4203
NAND	W29N01HVSINA	S34ML01G100TF100
DDR3	EM6GD16EWKG-12H	EM6GE16EWXD-12H

Table 1: Differences between 4205 and 4203

The brands and models of the two NANDs are different, they are the same size, both 128M. But DDR3 is different. In the model, 4205 is GD and 256M. 4203 is GE, 512M.

Figure 5: Ebang Ebit E9i and E10.1 control board

Schematic

Take a look on the schematics in order to have an idea about the board. There are more legible schematics rewritten for KiCAD but there are no projects for Altium as far as I know. However, be aware, that the schematic does not reflect the state of the board to 100% when ordered with “SD-Card mounted”, since the boot mode pins were altered! In my case, some PCB component names does not match with the schematic.

BOM

I couldn’t be able to find any BOM, so I’m trying to do it by my own means, but there is a huge task.

PCB

You can find the PCB for Altium on the files attached to this document. The original PCB file was imported to a newer version of Altium Designer using the DXP importer following these steps:

Figure 6: PCB Import process 1/4

Click on next

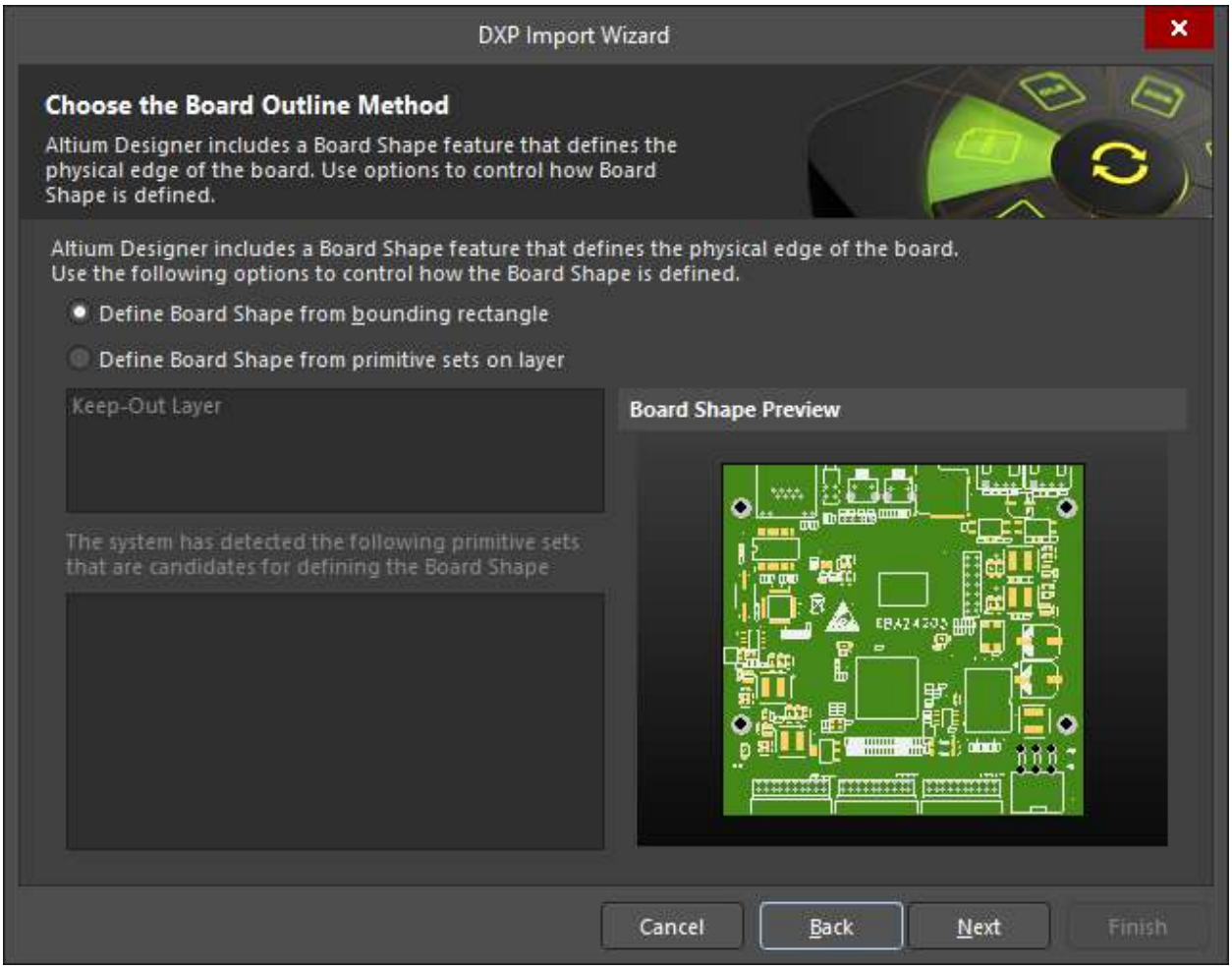


Figure 7: PCB Import process 2/4

Click on Next

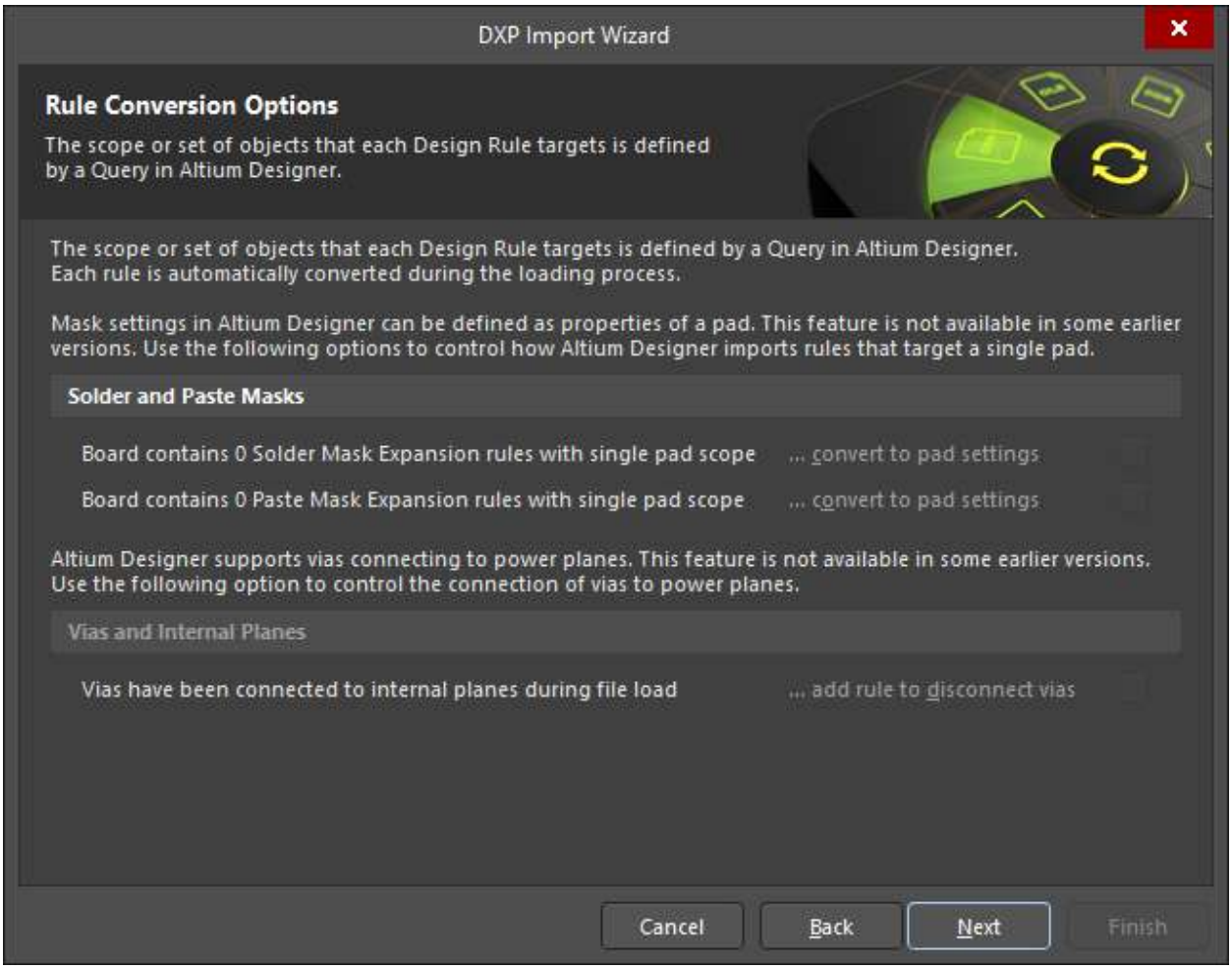


Figure 8: PCB Import process 3/4

Click on next

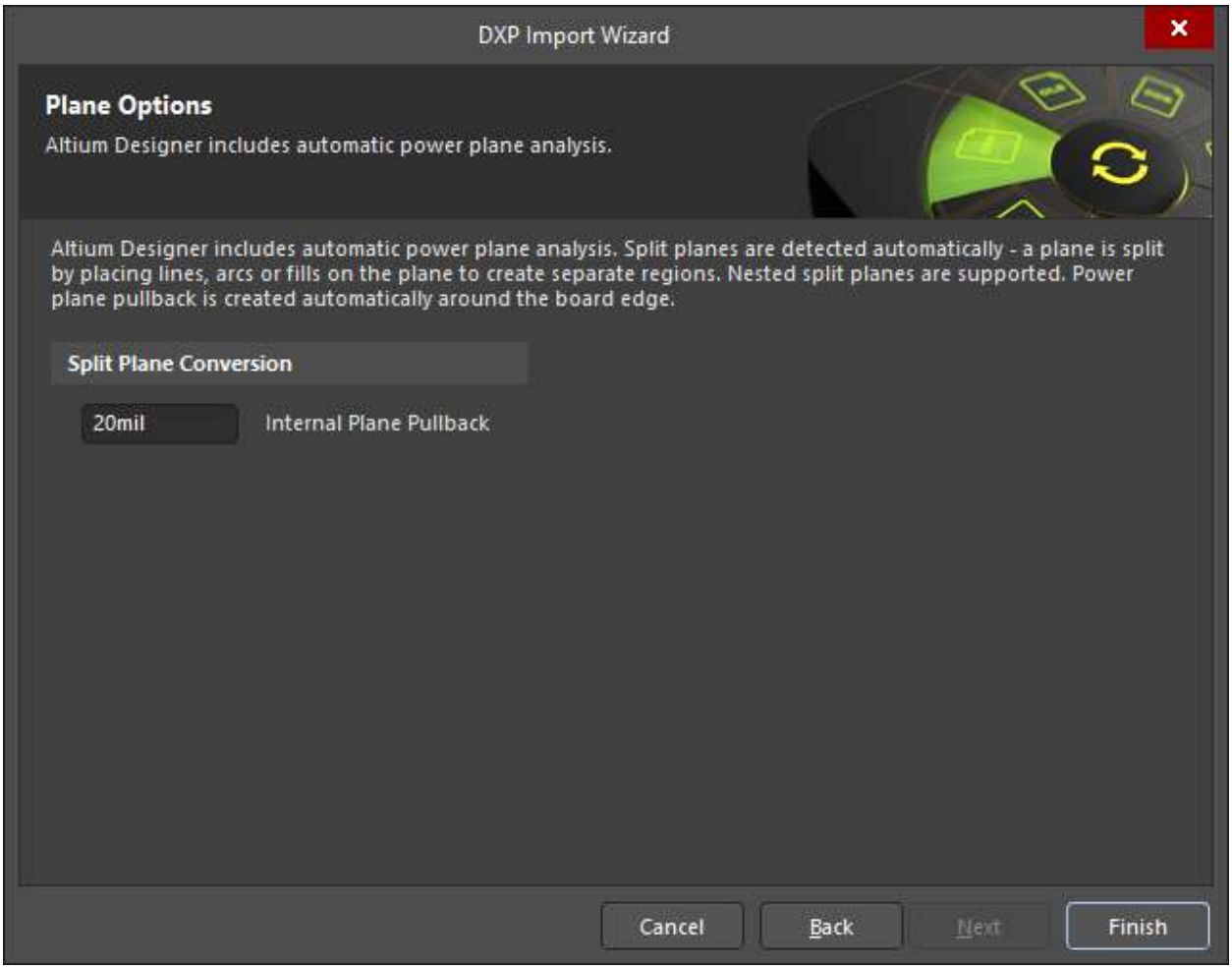


Figure 9: PCB Import process 4/4

Click on Finish.

It looks like this was not sufficient to do a good import (I do not know why), but I found an Altium file from a user with “corrected” PCB file for Altium.

Important components

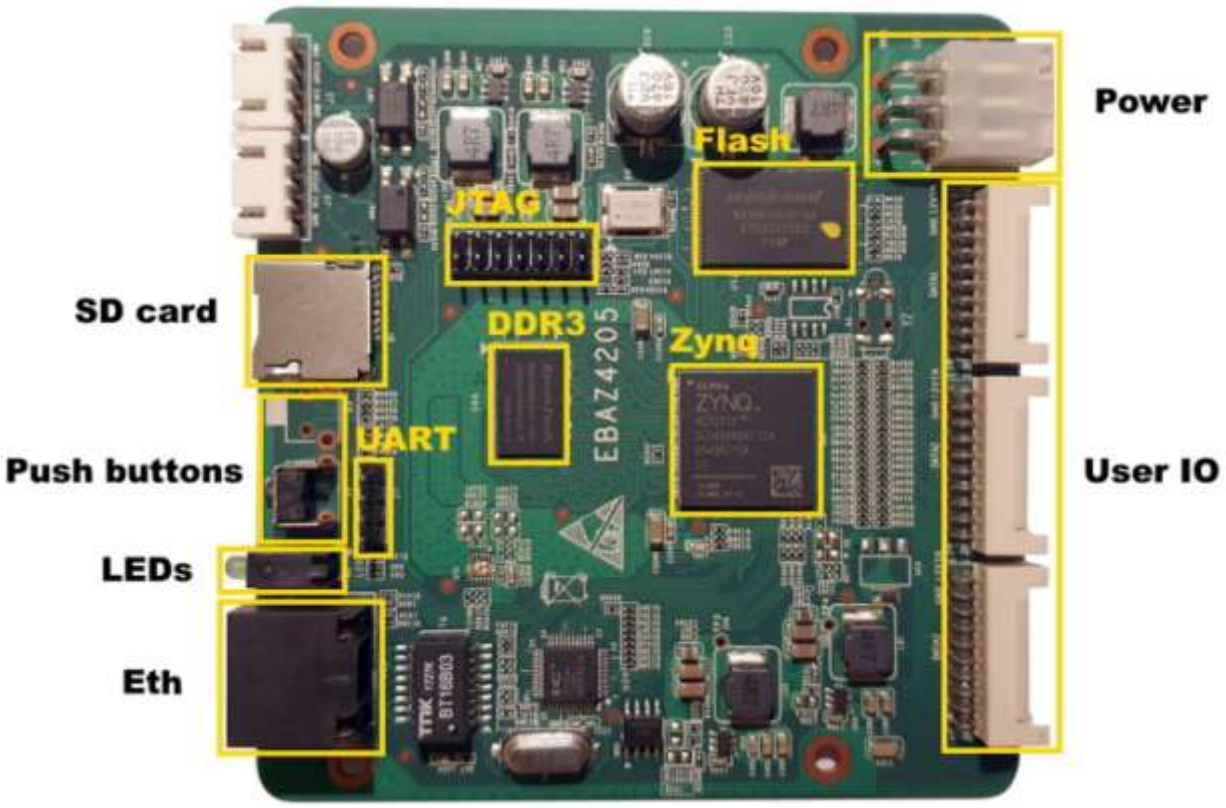


Figure 10: EBAZ4205 important components

Flash:

Figure 11: Flash picture

RAM Etron:

Figure 12: Etron RAM

Another RAM model:

Figure 13: UnilC RAM

ETH:

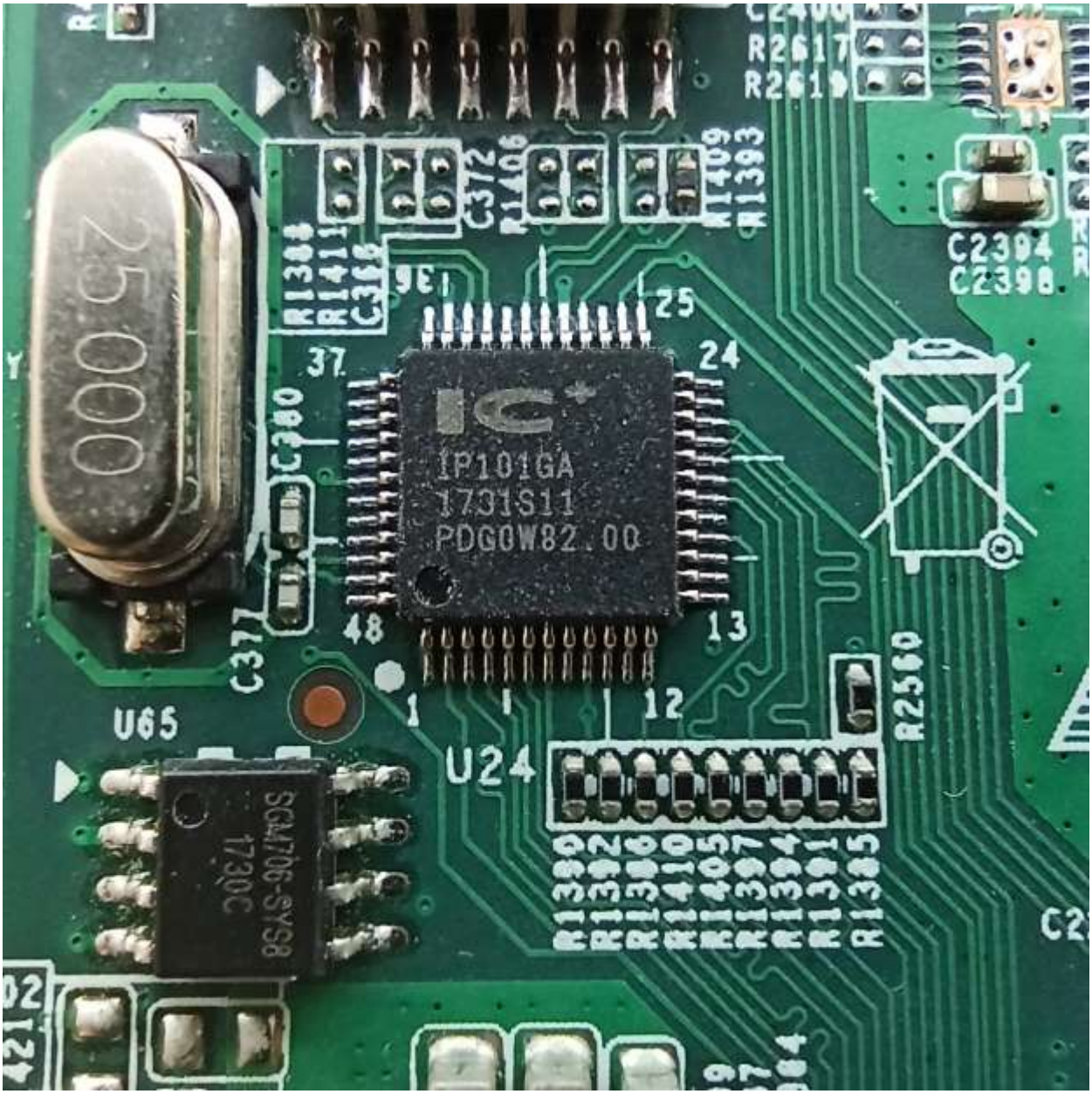


Figure 14: Ethernet IC

Zynq:

Figure 15: Zynq picture

Let's started

Finalized all the introductions, the next section will talk about what components are missing. If you want to test the card to see if it is working at least in a basic manner, I recommend you to check the minimum hardware setup (Missing components section, points 1, 2, 3 and 7) which involves serial communication, uSD slot and boot settings. Then, to connect the board (Interfaces pinout, points 1, 2, and 3) in order to got serial interface and power.

Depending on your hardware configuration: if you acquired an "untouched" or "uSD slot soldered" PCB with or without ETH crystal and optocouplers, you need to do different things to try it. The section **first turn on** will talk about this.

Programming way

I bought a Chinese copy of DLC10 programmer for 30€ to use JTAG in Aliexpress. It worked like a charm with the Xilinx driver. But, if you are on a budget, you can use other tools to develop on the EBAZ board.

There is a software called xvc (from Xilinx Virtual Cable https://embed-me.com/ebaz4205-a-cheap-network-jtag-debugger-using-xvc/) which is able to use a ESP32 (https://github.com/kholia/xvc-esp8266) or a Raspberry PI Pico board (https://github.com/kholia/xvc-pico) to emulate a JTAG.

https://www.hackster.io/news/dhiru-kholia-s-project-turns-a-raspberry-pi-pico-into-a-xilinx-virtual-cable-xvc-for-fpga-dev-work-25a58d9a99a9

https://www.adiuvoengineering.com/post/microzed-chronicles-jtag-using-a-raspberry-pi-pico

There is another firmware, to modify a FT232 breakout board with a Digilent firmware or something like that https://gist.github.com/rikka0w0/24b58b54473227502fa0334bbe75c3c1. If you don't want to buy a FT232 breakout you can make your own version. There are some schematics around: https://download.csdn.net/download/weixin_42157664/13073636

There is a way that I didn't have tried, which consist in make a boot image every time you "compile" (sorry I will abuse of the word "compile" as the process implied in making the hardware, the software and the boot image, but you need to know that synthesis have nothing to do with a "standard" compile process) your project. In this way, keeping the boot resistors in the uSD position, you will be able to test your program without a JTAG cable. But always there is a but, you will not able to use the debugger. I know GDB is able to debug trough network but I don't have idea about how to use it.

I hope you can find a suitable programming way for your needs.

Missing components

I got my EBAZ4205 from AliExpress, but I also saw that there were some listings on eBay. I got my board for 30usd in February of 2022, but as of October 2022 the price is getting closer to 40usd; there is some demand for these boards. I bought a second one on October 2022.

The state in which your EBAZ4205 board arrives does depend on where you bought it from. Sometimes the board may come fully modded out, ready for development, but more often you may need to add some components to get the board more complete to work with. For a detailed breakdown of the various hardware components that may be missing on a board and what the specific components do, check out this section.

I'll be going through some hardware modifications that are required. So, get your soldering iron and computer on standby and let's get hacking.

These boards are supplied in various build states. This means you may not have all the parts (some shown in photos below) that you will need to get the board talking to you.

NOTE: I will mention along the document to PS (Processing System) and PL (Programmable Logic) "sides" of the Zynq. If you don't know what PS and PL is, please take a look to a Zynq book.

NOTE 2: At the end of the section, you will find a table with all the components required to do all the modifications.

NOTE 3: Soldering 0402 components it's not recommended at all if you are not familiar with small SMD soldering. You can ruin your board trying to do some of the modifications. So, do the minimum changes if you don't have the experience or the equipment required to do the modifications.

NOTE 4: Please pay attention, in each section there are differences between cards and different build states, so please read carefully each section to determine which requires intervention and which no.

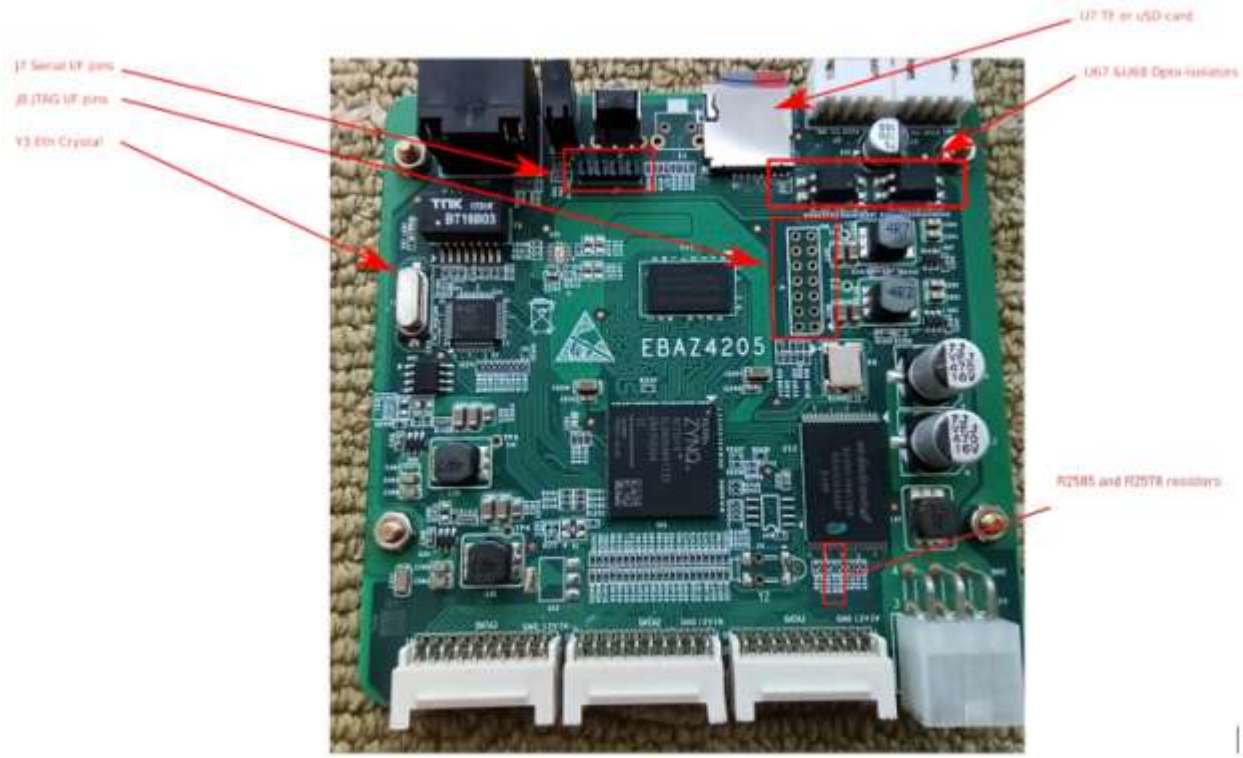


Figure 16: Top view crystal variant

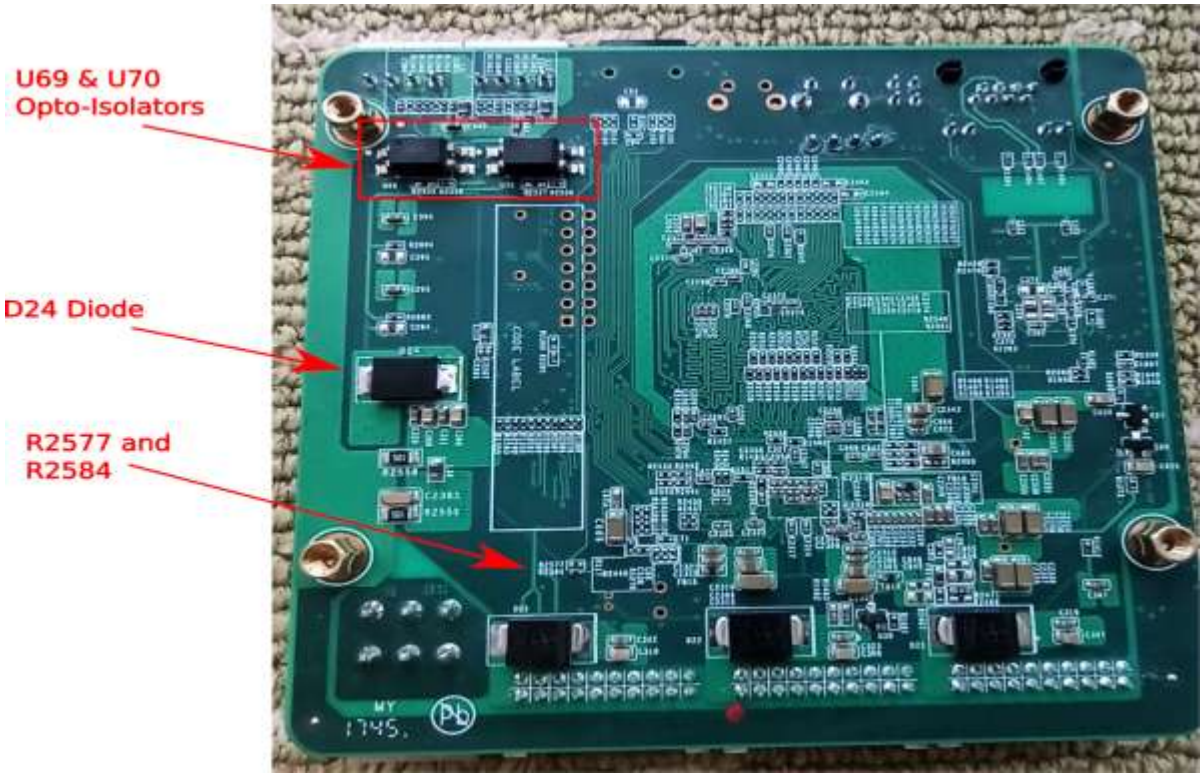


Figure 17: Bottom view crystal variant

According to the previous figures, the interesting points are:

1. **J7 Serial interface pins**. At least in the early stages of understanding the functions of this board, these pins make control and monitoring of the board easier. The serial interface is a 3V3 interface. So, make sure any USB-Serial module you connect can be set-up for this.

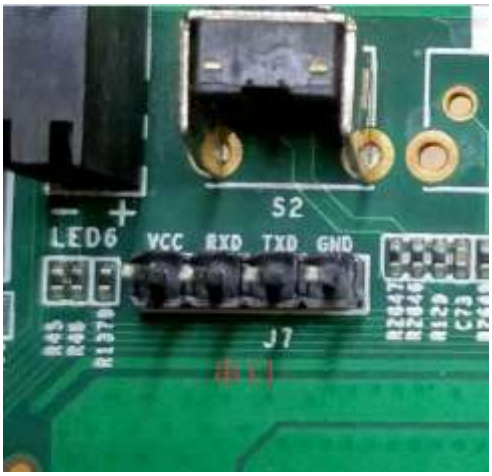


Figure 18: Serial pinout

MODIFICATION STEPS: These pins did come preinstalled on my board. As they are TH, it's easy to weld. Any 2.54mm male straight headers will work.

2. **J8 JTAG interface pins**. The JTAG pins will make it easier to update and monitor the PL side of the Zynq device.

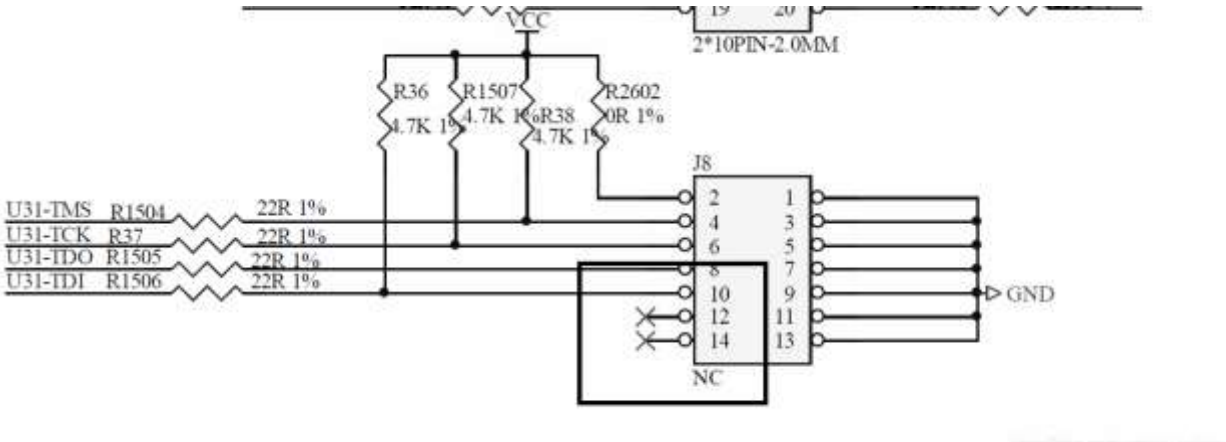


Figure 19: JTAG schematic

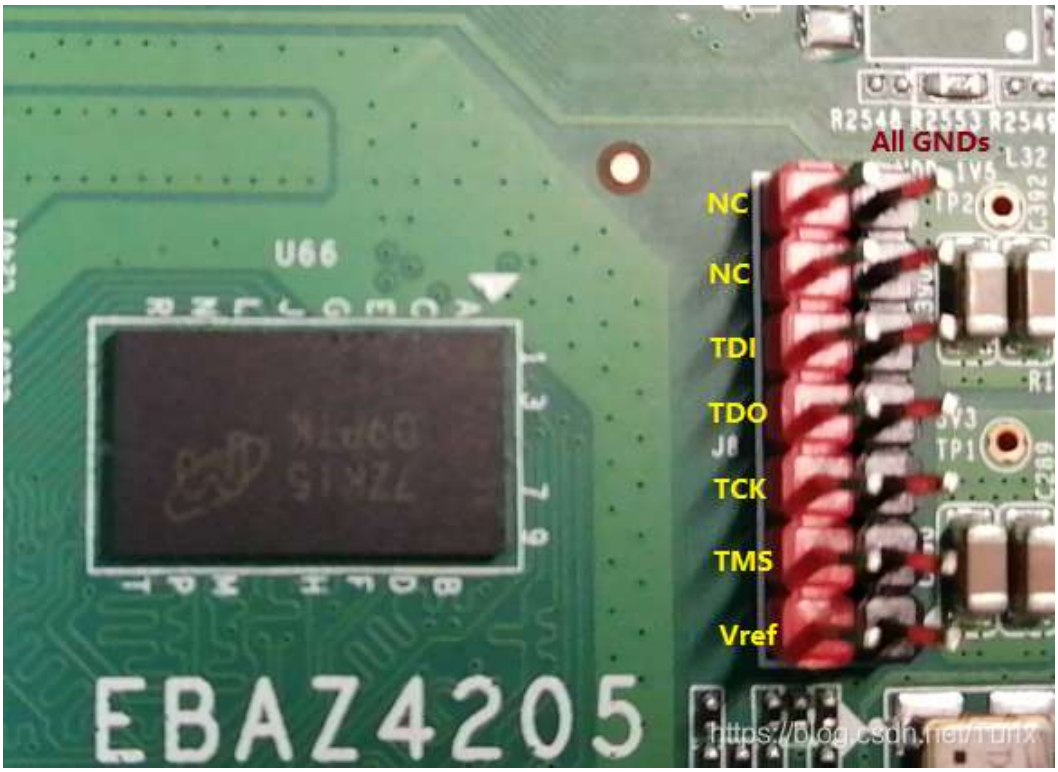


Figure 20: JTAG pinout

MODIFICATION STEPS: These pins did come preinstalled on my board. As they are TH, it's easy to weld. Any 2.54mm male straight headers will work.

3. **TF or uSD Card slot**. Again, you can live without this device, but your development life will be easier if you have one. If you are planning to work only with the NAND or JTAG, you can avoid the soldering. As we want to "convert" this board to a full development board with all the available options, we recommend to install one.

   MODIFICATION STEPS: This slot did come preinstalled on my board. It's easy to weld anyway.

4. **Y3 crystal**. (25.000MHZ) provides a clock signal to the ethernet transceiver (IP101GA).
   Crystal less variant: This device is not actually needed. The Ethernet PHY is connected to the PL side of the Zynq. This means the PL logic needs to be programmed for the Ethernet connector to work. The PL can provide the required clock instead of the crystal.
   But if you choose to get the 25 Mhz crystal, it is important to know that you'll need to also get two 22 pF capacitors (C380 and C377), and remove the R1485 resistor. Personally, I want this choice because I don't want to carry on this pin/function along the programming of the PL side.

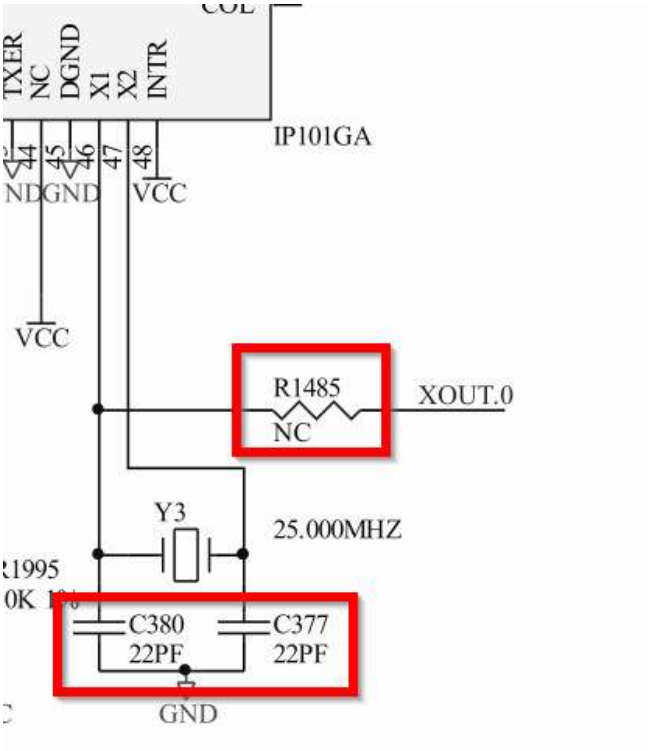


Figure 21: Ethernet crystal circuit

MODIFICATION STEPS: I was not able to weld the two capacitors 0402, because it's very difficult and too close to the PHY. So, I removed R1485 and welded two TH capacitors to a LED4 GND. It's not nice, but it's works :)

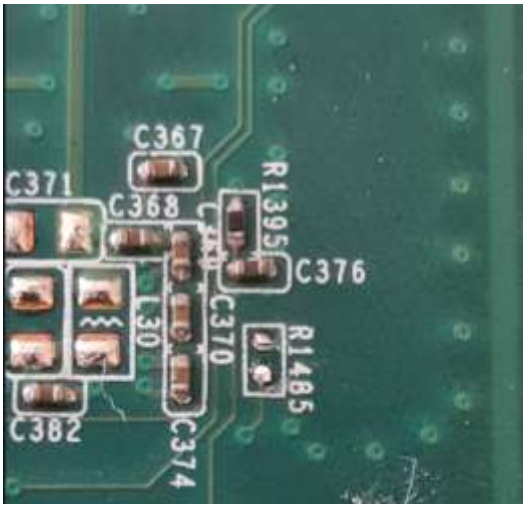


Figure 22: R1485 removed

Figure 23: Filter capacitors welded PHY

Crystal variant: Nothing to change in this section!

5. **U67, U68, U69 and U70 Opto-isolators** (PC817).
   Crystal less variant: These devices are not really required unless you specifically want to use them. In this case, if they are present, then pin 4 (labelled as PWM) of J4/J5 can only be used as outputs. The output side of the opto-isolator is connected the VCC pins on J3/4/5 through pull-up resistors R2550 and R2556, so make sure anything connected to pin 4 can handle the voltage on J3/4/5 which is 12V.

   Pin 3 (labelled as SPEED) of J4/J5 can only be used as inputs. The thing is, this will provide to you 2 digital inputs, and 2 digital outputs, all opto-isolated. If you don't want to have this, you can avoid the soldering.

   If they are missing, then they will need to be fitted, but by default has 33 Ohm resistors on R2553, R2552, R2554 and R2557. This bypasses the missing opto-isolators and resistors for PWM and SPEED for fan connectors.

   Due to no isolation be careful if applying more than the maximum allowed 1.2 to 3.3V of FPGA I/O set voltage. In addition, pin 3 on J3 and J5 (labelled as speed) is connected to 3.3V via pull-up resistors R1344 and R1361, so using logic levels less than 3.3V should probably be avoided on those pins. Alternatively, just remove those resistors.

   NOTE: As you may see on the schematic, the resistor labelled as 3555 for J5 connector, near de opto-isolator U70, in the PCB is R2555.

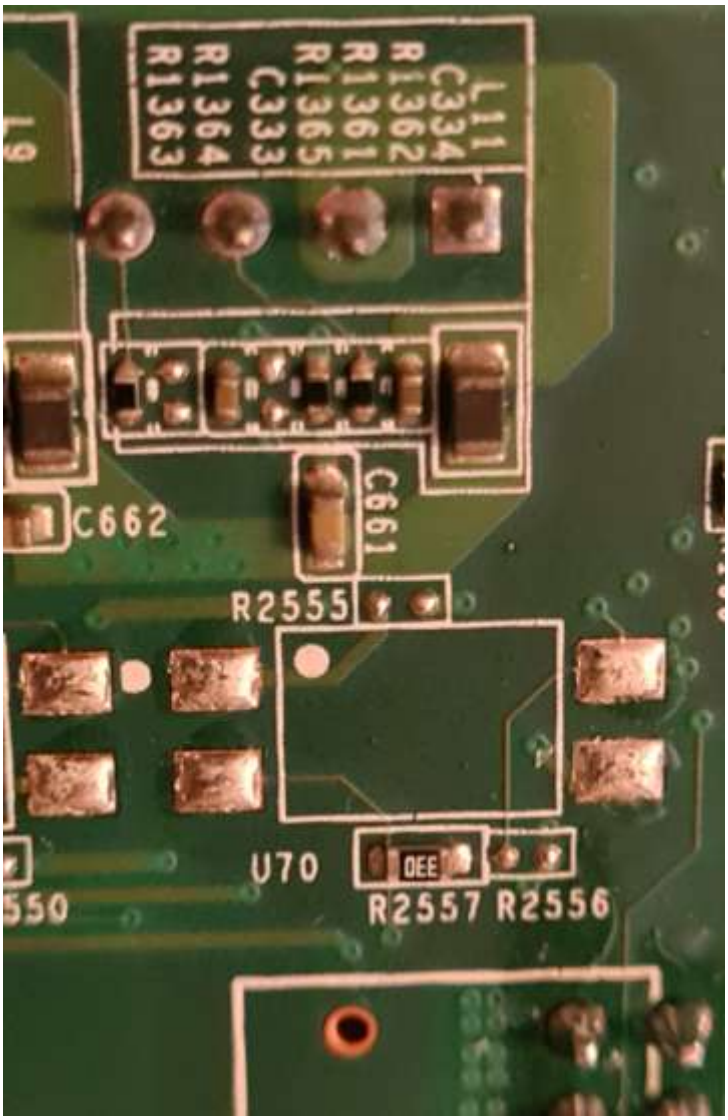


Figure 24: Marking differences R2555 crystal less variant

MODIFICATION STEPS: In the crystal less variant, there are no opto-isolators. So, I followed the schematic checking against the PCB to draw the circuit.

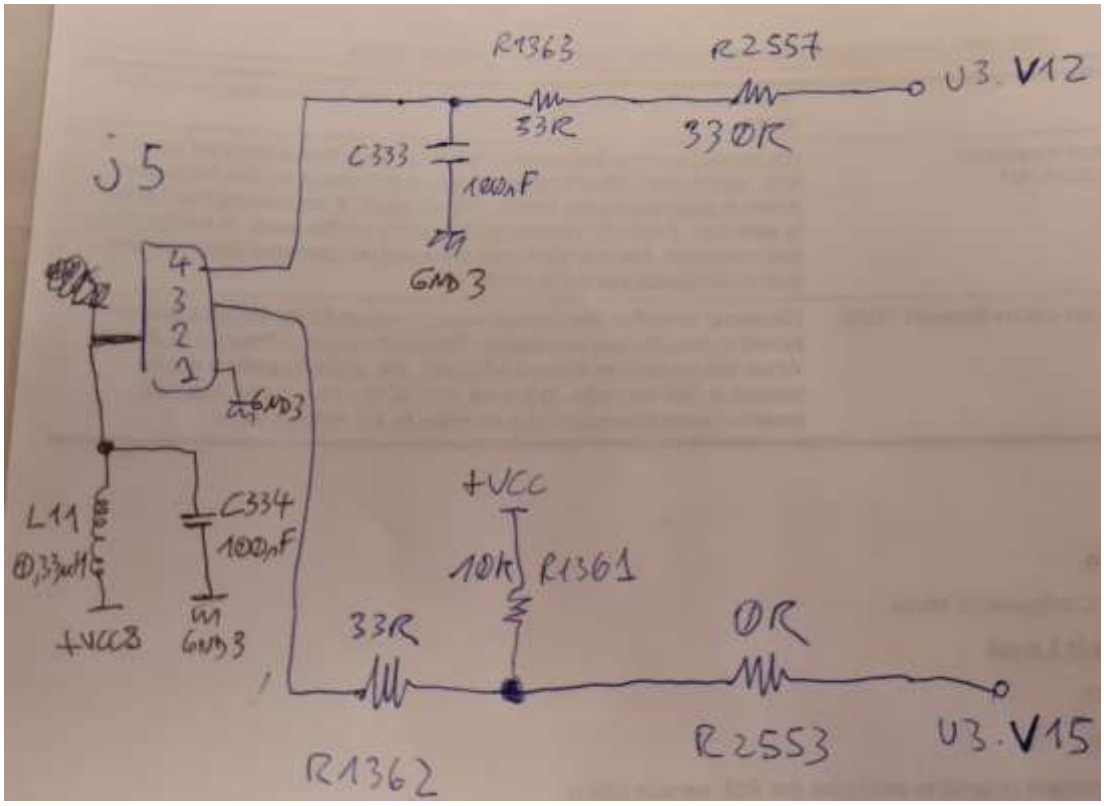


Figure 25: J5 equivalent schematic crystal-less variant

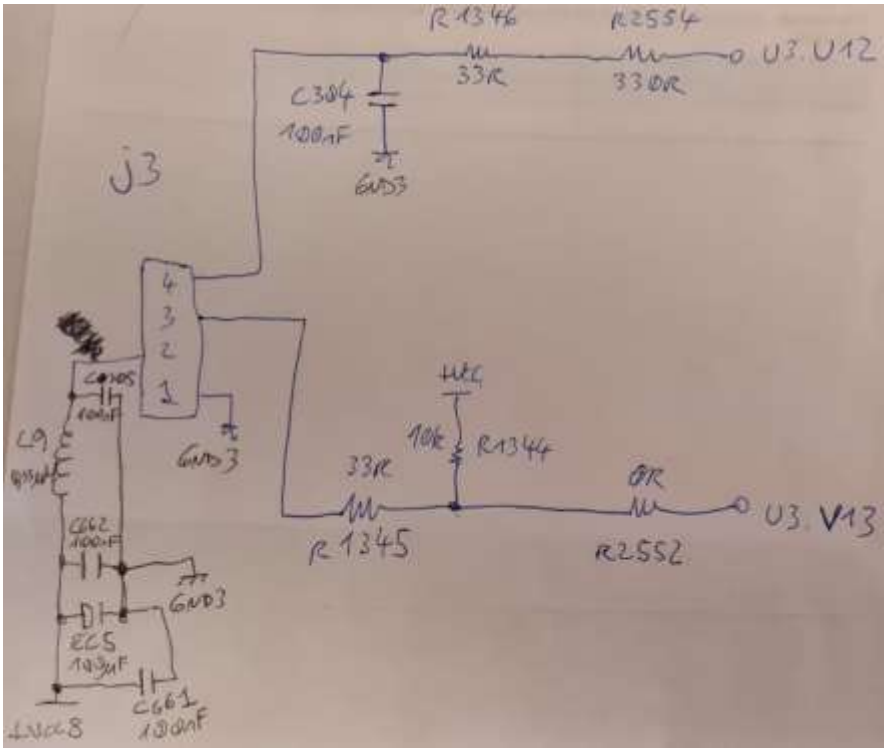


Figure 26: J3 equivalent schematic crystal-less variant

As a lot of 0402 components are required to weld the opto-isolators, I decided to keep in that way and use opto-isolators outside the board.
Crystal variant: In this board the opto-isolators are present so there is nothing to change. Just let's follow the schematic to check the circuit:

NOTE: As you may see on the schematic, the resistor labelled as 3555 for J5 connector, near de opto-isolator U70, in the PCB is R2555.

Figure 27: Marking differences R2555 on the crystal variant

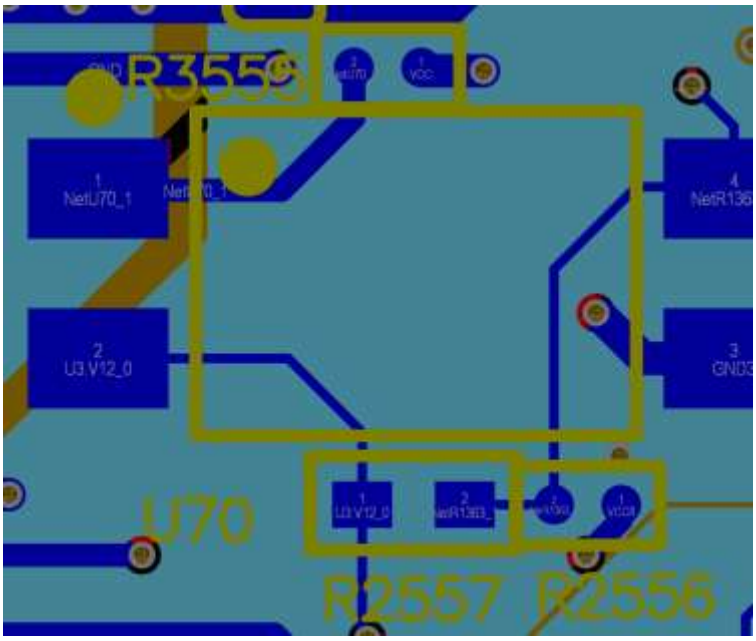


Figure 28: Marking differences R3555 crystal variant

NOTE 2: As you may see on the schematic, the resistor labelled as 2350 for J3 connector, near de opto-isolator U69, in the PCB is R2550.

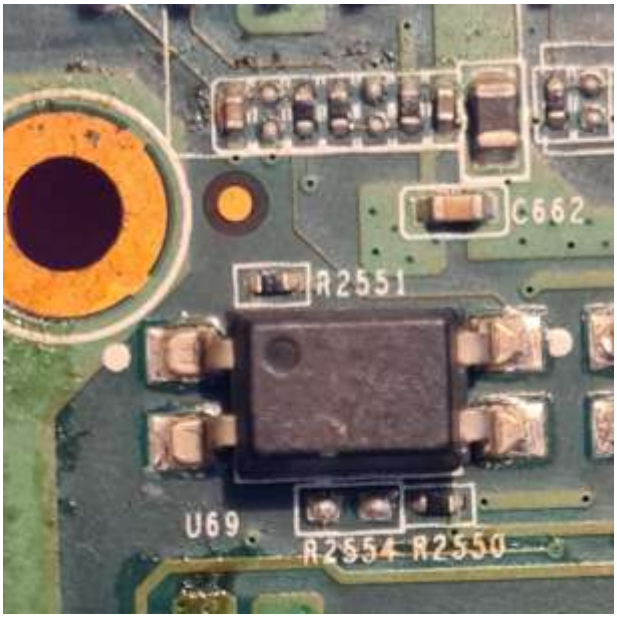


Figure 29: Marking differences R2550 crystal variant

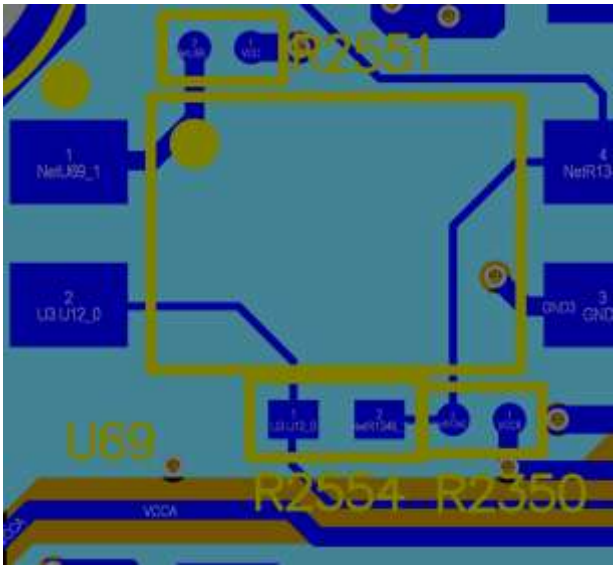


Figure 30: Marking differences R2350 crystal variant

The schematic with the components present on the PCB results:

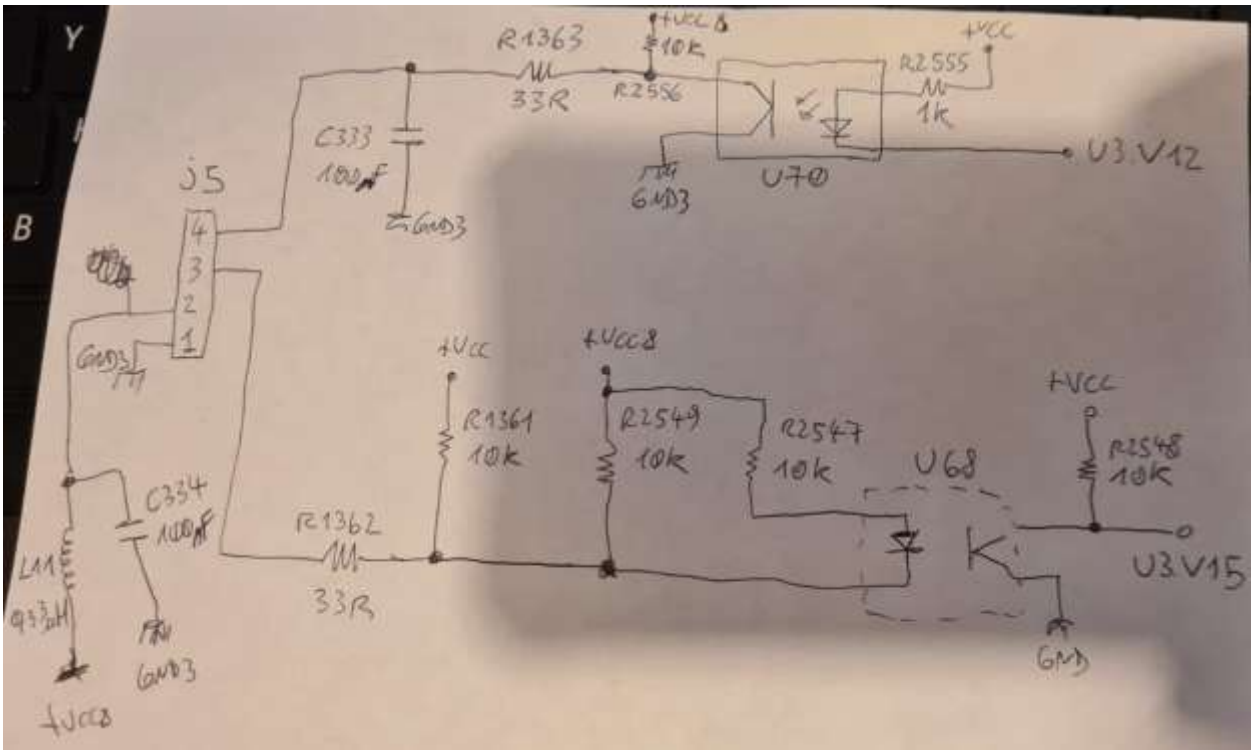


Figure 31: J5 equivalent schematic crystal variant

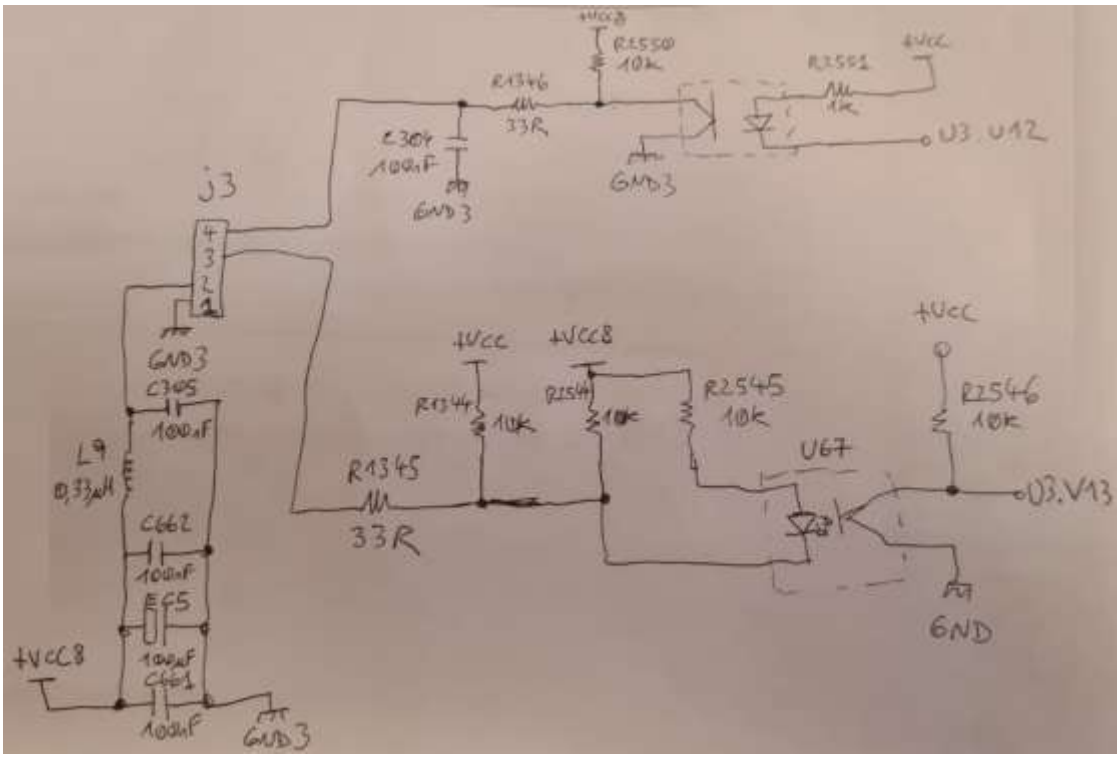


Figure 32: J3 equivalent schematic crystal variant

6. **Diode D24.** Many boards are supplied without this part. This part is not essential as the board can be powered via the three 2mm pitch data connectors if you decide to not solder it. However, the J4 Molex connector is preferred by me as it provides additional filtering and should be an easier connector to source. One of the other diodes (like D23) can be moved to D24 if the DATA3 is not going to be used to provide power. This will leave DATA3 without the possibility to provide power. You also can bridge the diode, the only caveat is, don't forget to connect power in the right way, if you mix VCC and GND you will burn something.

   MODIFICATION STEPS: Just weld the diode in the same direction than the others one.

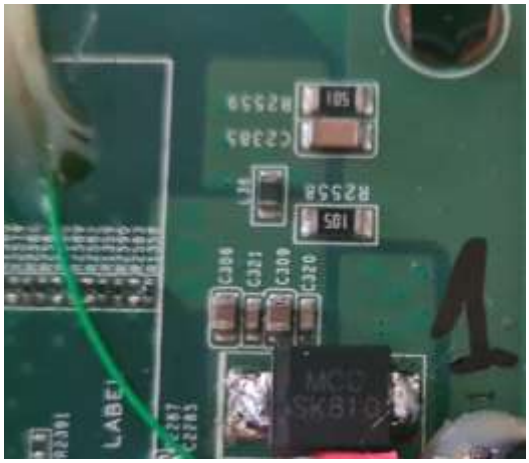


Figure 33: SK810 diode

7. **BOOT select resistors.** There are 4 resistors in charge of select booting.

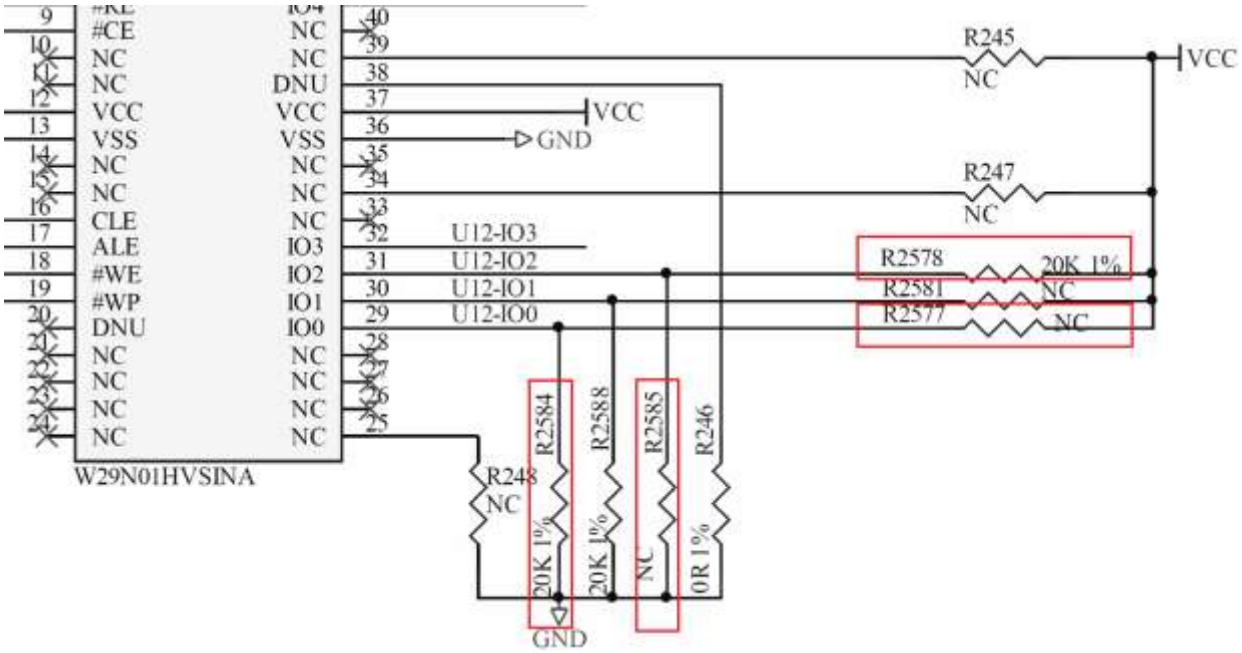


Figure 34: SD boot required modifications

This is the truth table for those pins:

Boot device	IO 0 (MIO [5])	IO 2 (MIO [4])
JTAG	0	0
NAND	0	1
QSPI	1	0

uSD	1	1

Table 2: Boot mode selection

So, let's analyze the possible cases.
Fixed JTAG: R present in R2584 (pull-down) and not present in R2577 (pull-up). R present in R2585 (pull-down) and not present in R2578 (pull-up).
Fixed NAND: R present in R2584 (pull-down) and not present in R2577 (pull-up). R not present in R2585 (pull-down) and present in R2578 (pull-up).
Fixed uSD: R not present in R2584 (pull-down) and present in R2577 (pull-up). R not present in R2585 (pull-down) and present in R2578 (pull-up).
Fixed QSPI: R not present in R2584 (pull-down) and present in R2577 (pull-up). R present in R2585 (pull-down) and not present in R2578 (pull-up).

For reference:

Table 6-4: **Boot Mode MIO Strapping Pins**

Pin-signal / Mode	MIO[8]	MIO[7]	MIO[6]	MIO[5]	MIO[4]	MIO[3]	MIO[2]
	VMODE[1]	VMODE[0]	BOOT_MODE[4]	BOOT_MODE[0]	BOOT_MODE[2]	BOOT_MODE[1]	BOOT_MODE[3]
Boot Devices							
JTAG Boot Mode; cascaded is most common(1)				0	0	0	**JTAG Chain Routing(2)** 0: Cascade mode 1: Independent mode
NOR Boot(3)				0	0	1	
NAND				0	1	0	
Quad-SPI(3)				1	0	0	
SD Card				1	1	0	
Mode for all 3 PLLs							
PLL Enabled			0	Hardware waits for PLL to lock, then executes BootROM.			
PLL Bypassed			1	Allows for a wide PS_CLK frequency range.			
MIO Bank Voltage(4)							
	Bank 1	**Bank 0**	Voltage Bank 0 includes MIO pins 0 thru 15. Voltage Bank 1 includes MIO pins 16 thru 53.				
2.5 V, 3.3 V	0	0					
1.8 V	1	1					

Notes:
1. JTAG cascaded mode is most common and is the assumed mode in all the references to JTAG mode except where noted.
2. For secure mode, JTAG is not enabled and MIO[2] is ignored.
3. The Quad-SPI and NOR boot modes support execute-in-place (this support is always non-secure)
4. Voltage Banks 0 and 1 must be set to the same value when an interface spans across these voltage banks. Examples include NOR, 16-bit NAND, and a wide TPIU test port. Other interface configuration may also span the two banks.

Figure 35: Boot mode selection Xilinx reference

The boot process of ZYNQ SOC is mainly based on ARM. After power-on, the hardware reads the IO port of the PS port to determine whether to boot from QSPI, NAND, uSD card or JTAG. The board is booted from NAND by default, I think, the resistor is on R2584 by default, which

indicates boot from NAND on an unmodified board. For software debugging and downloading NAND, it must be booted from JTAG.

Let's see the physical locations on the board:

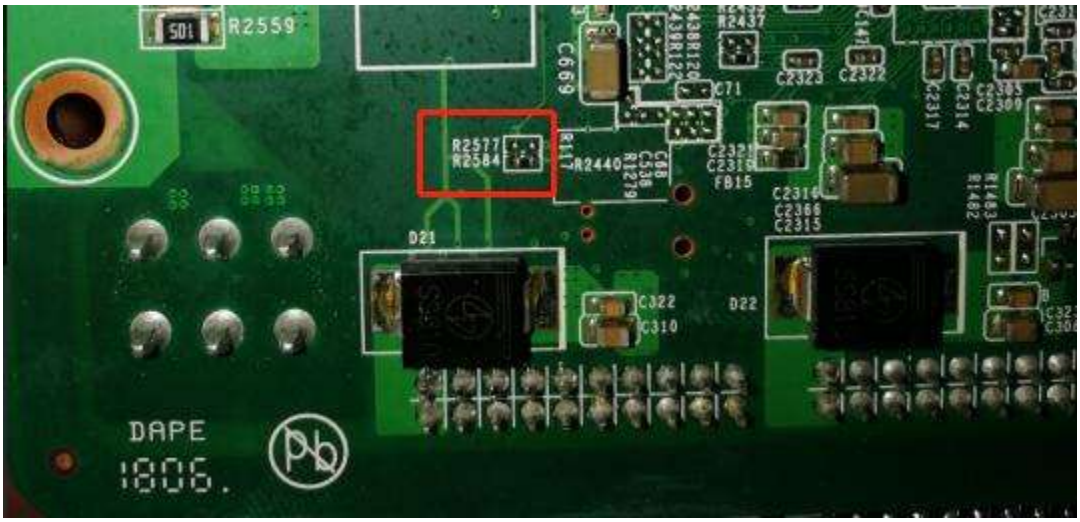


Figure 36: IO 0 resistors

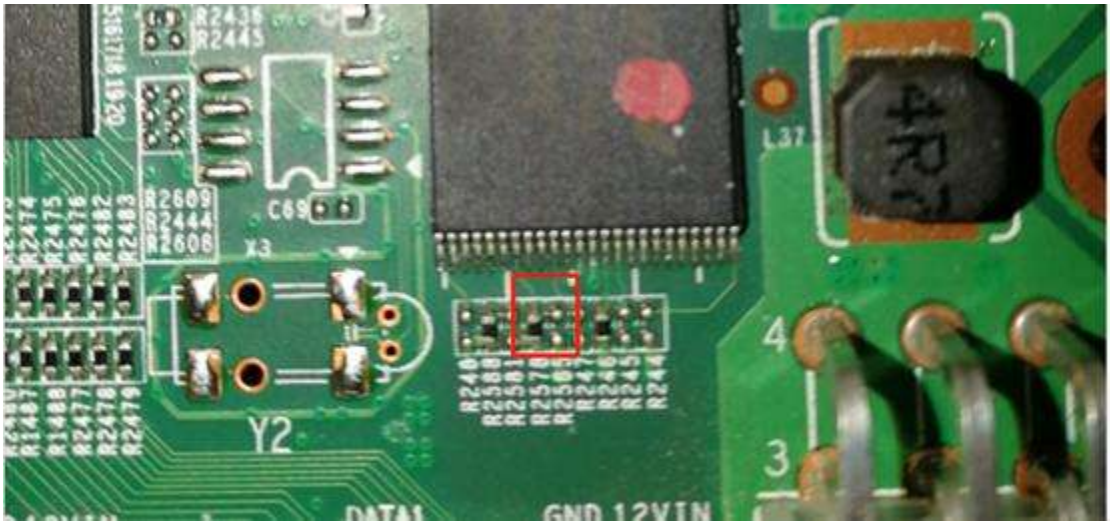


Figure 37: IO 2 resistors

Other solutions: It will be very problematic to change resistors every time I need to change between JTAG debugging and NAND booting. So, adding a switch can be a good solution, so the quick way is to give up the uSD card, leaving the resistor on R2584 position, then disconnect the end of the R2578 resistor connected to VCC, and connect it to a small switch (SPDT), so that the JTAG mode and the NAND boot mode can be controlled by the switch. IO2 is connected to 20k and then grounded or high.

IO 2 ---- 20K ---- SWITCH SPDT ---- GND or VCC.

Figure 38: switch example

This solution it's OK but you can do the same with the other resistor and have all the boot modes available. Another idea could be to put a dip switch.

MODIFICATION STEPS: I want to have all the boot modes available, so I took the idea from that user but with both IO's.

IO 2 ---- 20K ---- SWITCH SPDT ---- GND or VCC.

IO 0 ---- 20K ---- SWITCH SPDT ---- GND or VCC.

This is the process of welding on 0402 pads. Please be extra careful or you will ruin your board. Proposed circuit:

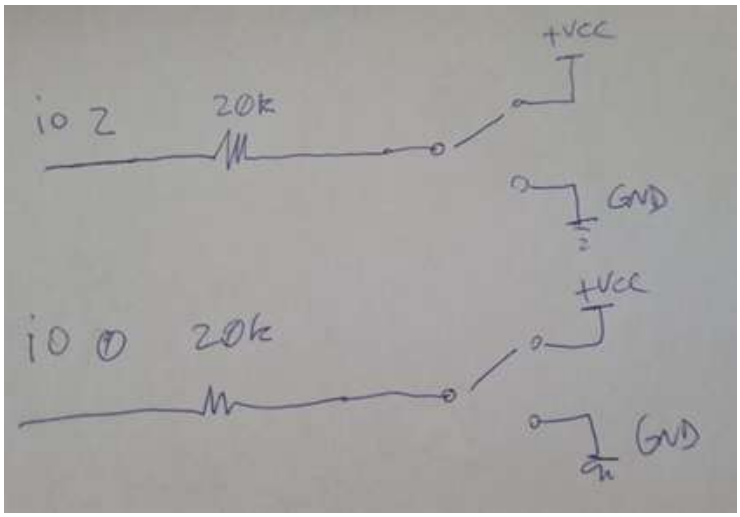


Figure 39: IO0 and IO2 schematic

On the physical part:

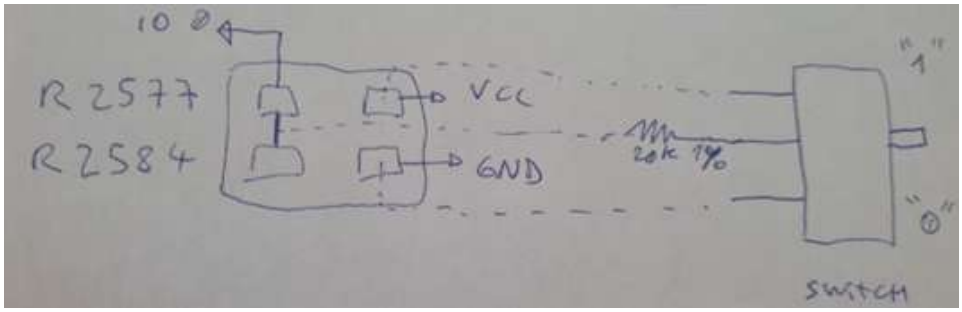


Figure 40: Physical wiring of IO0

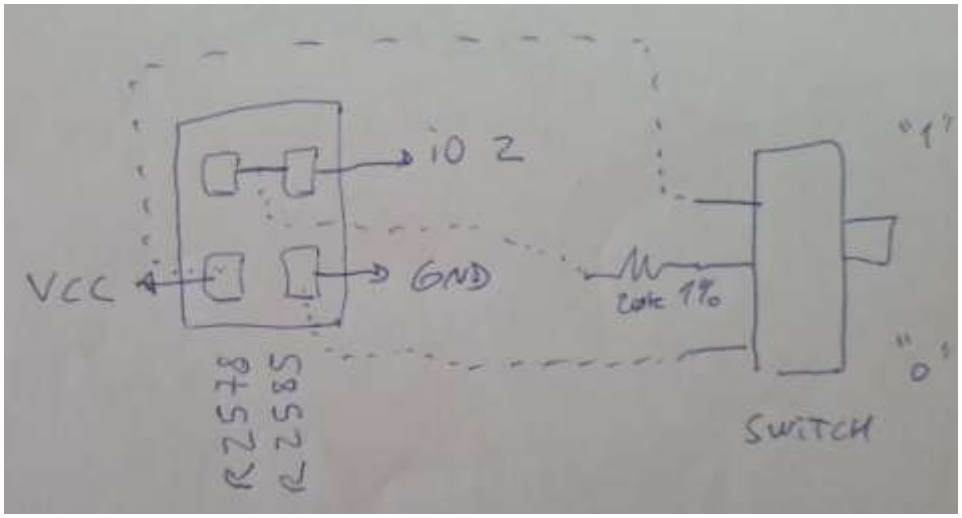


Figure 41: Physical wiring of IO2

Mod for IO 0 step by step:

Remove the resistor on R2577 or R2548 depending on your configuration and weld a wire wrap (both pads are the same net). This wire will go to the “center” of the SPDT switch, but with a 20K resistor.

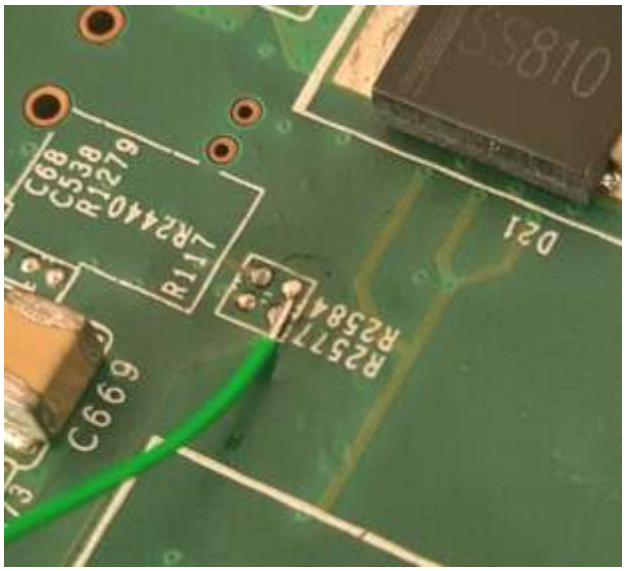


Figure 42: IO 0 modification step 1 of 4

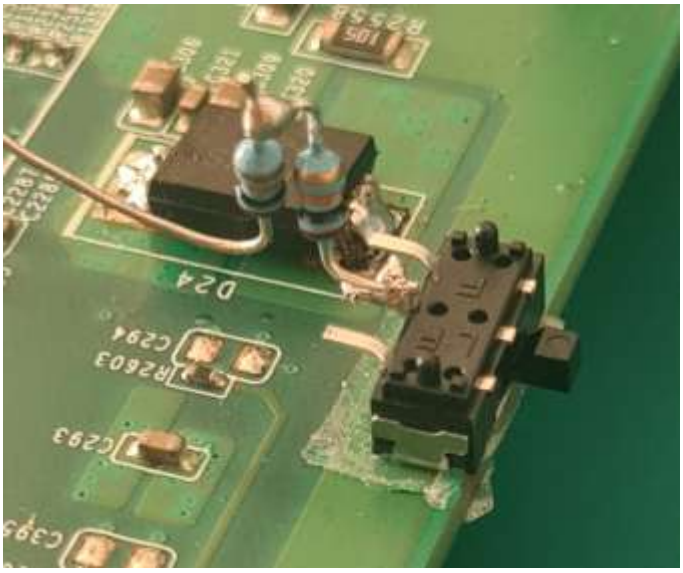


Figure 43: IO 0 modification step 2 of 4

So, I took a switch, with a double sticky tape fixed (at least I have tried to) and weld two 10K resistor in series to the “center”. The end of the resistor, goes to the wire wrap.

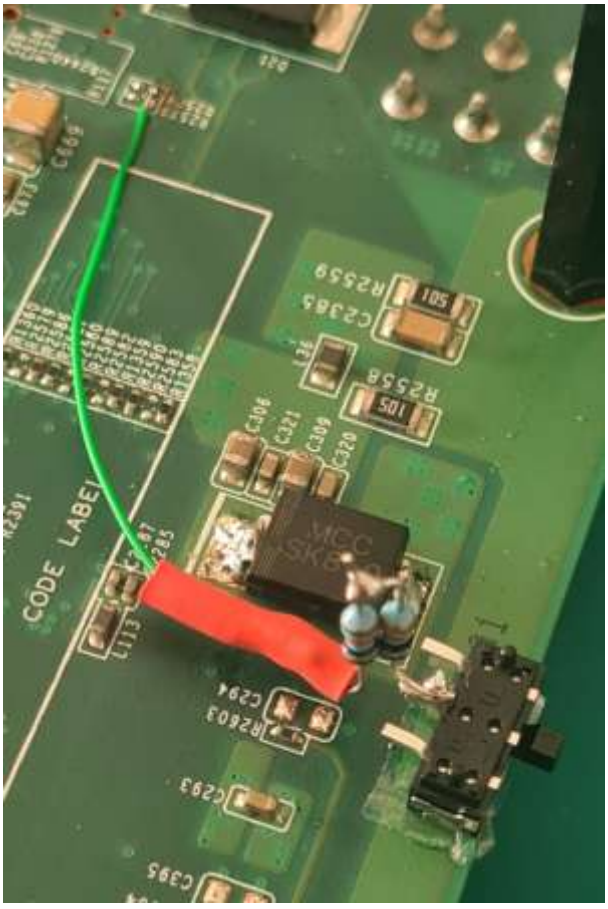


Figure 44: IO 0 modification step 3 of 4

Initially I welded VCC and GND on the resistor pads, but then I realized than GND is best on any other place, and took only VCC from the resistor’s pads. Then I use hot melt to “lock” the wire wraps against any movement. Finally, I took GND from the JTAG connector.

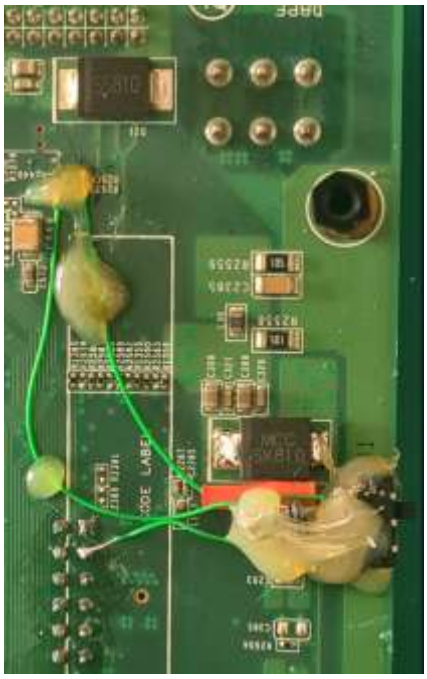


Figure 45: IO 0 modification step 4 of 4

Mod for IO 2 step by step: Remove R2578 or R2585 depending on your configuration and weld a wire wrap (both pads are the same net). This wire will go to the “center” of the SPDT switch, but with a 20K resistor.

NOTE: When I welded this part, I made a mistake. When I removed the R2585 resistor, I also removed R2588. So, I had to weld it again but then I produced some short with the pads on R2585 so I have to remove the holt melt and solve the problem. The SMD rework it’s not easy with 0402 components, so be careful working with these modifications.

Figure 46: IO 2 modification step 1 of 4

So, I took a switch, with a double sticky tape fixed (at least I have tried to) and weld two 10K resistor in series to the “center”. The end of the resistor, goes to the wire wrap.

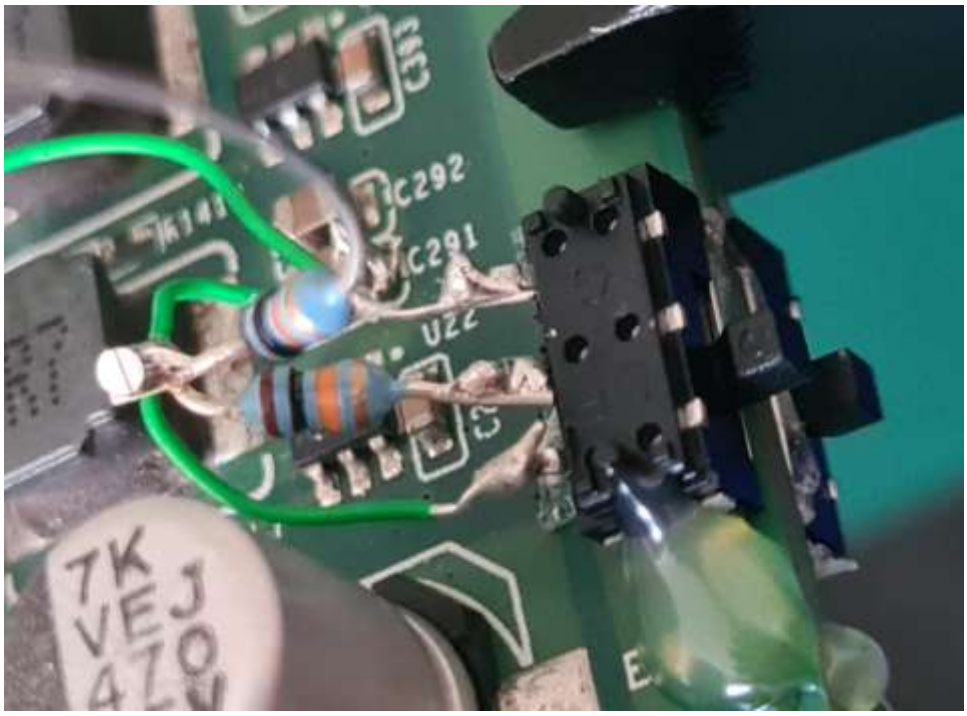


Figure 47: IO 2 modification step 2 of 4

I took GND from C291 capacitor:

Figure 48: IO 2 modification step 3 of 4

And VCC from the resistor pad and the hot melt to immobilize all. This is the result:

Figure 49: IO 2 modification step 4 of 4

Final result:

Figure 50: IO 0 and 2 final result

Off course, as it's can't be in another way, I mixed the orientations of "1" and "0" so instead to have the "0" down and "1" up, they are mixed. So, one "1" is up, and the other "1" is down. So, I wrote with a marker what is each position. I will pay more attention next time I suppose…

8. **SWITCH S3**. Can be added if you short out missing R2649 zero Ohm Resistor or weld a 0R. Or you can just link right leg of SW2 to right leg of SW3 if looking from front of PCB. (Leg closest to LAN socket). Please note that in the schematic this R show up as "R2641A", not "R2649" as in PCB. Also, a 1uF capacitor on C2410 is recommended.

MODIFICATION STEPS: Followed the schematic to check what is missing on my PCB:

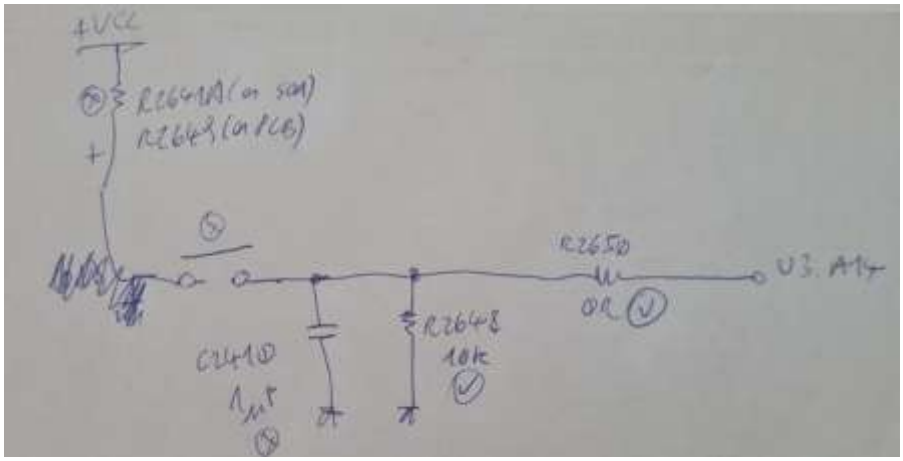


Figure 51: S3 switch schematic

Then I welded the switch and shorted the resistor with a wire wrap.

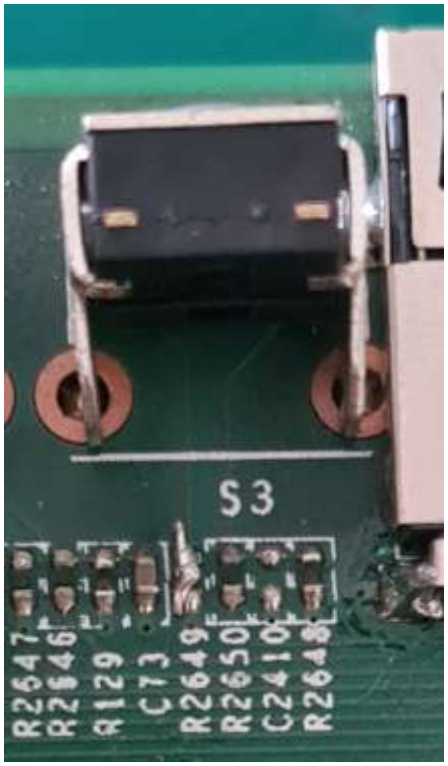


Figure 52: S3 switch welded

Then I solder the filter capacitor on the back

Figure 53: S3 switch capacitor

Final result:

Figure 54: S3 switch final result

9. **X5 XTAL**. The default PL part of the motherboard is not soldered, there are two ways to use the PL logic part:
   1. Use the PS part to share the clock to the PL.
   2. The PL part solders a single crystal oscillator including a 3225 active crystal oscillator (recommended 50Mhz, 3225 size); one 22R 0402 resistor at R1372, and a 0603 1.2uH inductor at L29 (on bottom of PCB). An active crystal oscillator is enough, and there is a PLL inside the FPGA, which can perform frequency multiplication.

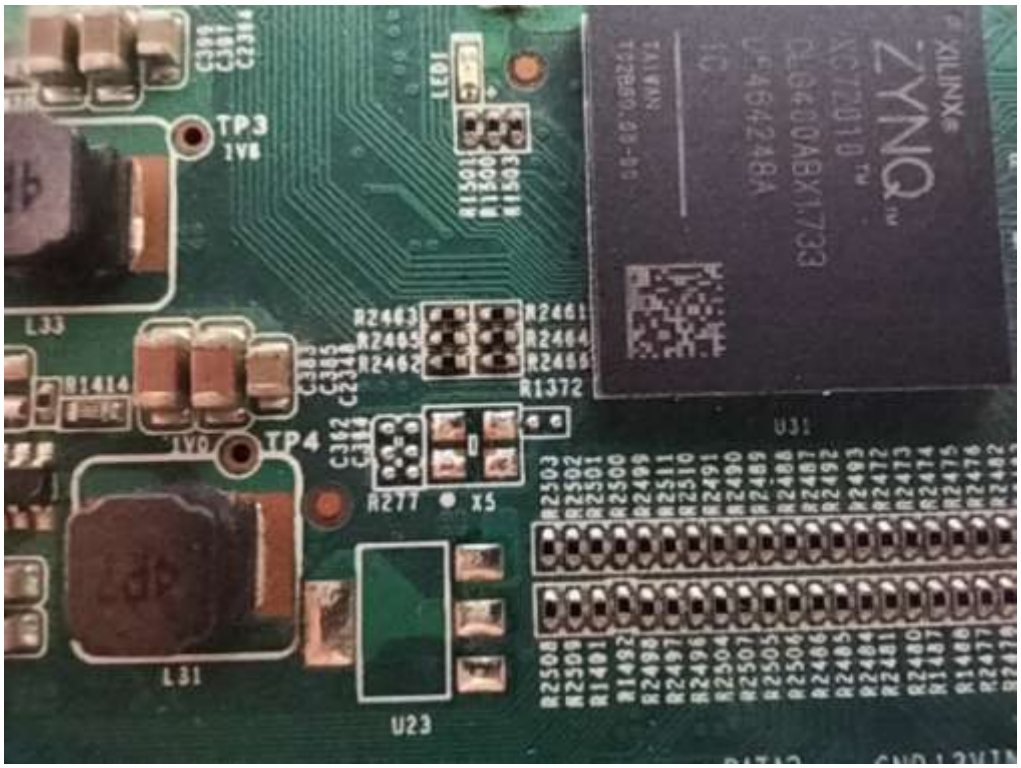


Figure 55: X5 xtal location

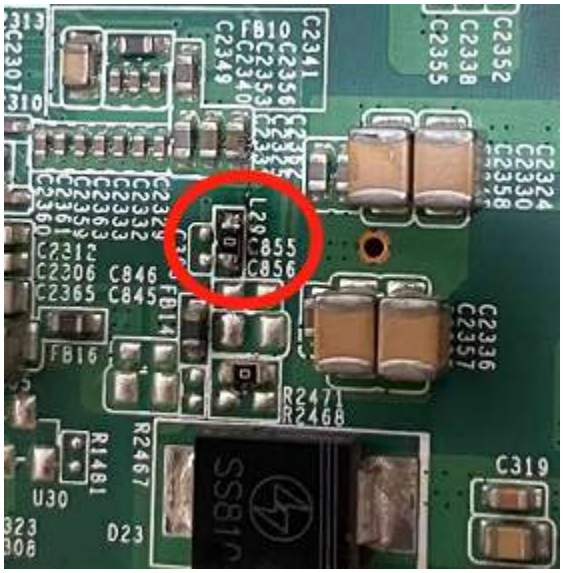


Figure 56: L29 on the back

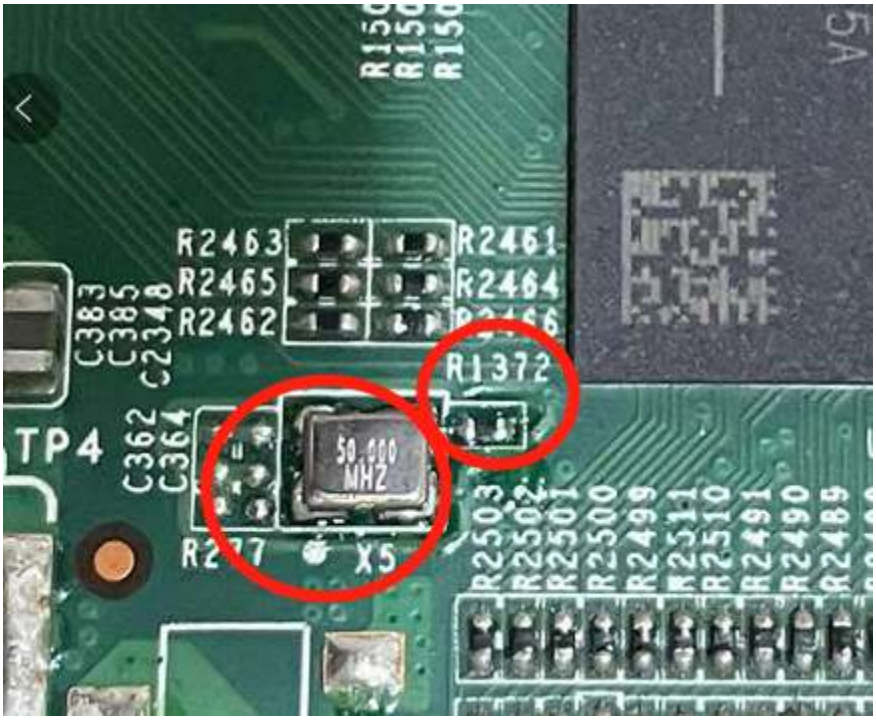


Figure 57: X5 welding

X5 welding 3225 50M active crystal oscillator, R1372 welding 0402 22R resistance.

MODIFICATION STEPS: I welded a 1.2uH inductor on L29 because it's 0603 so it's easy to weld.

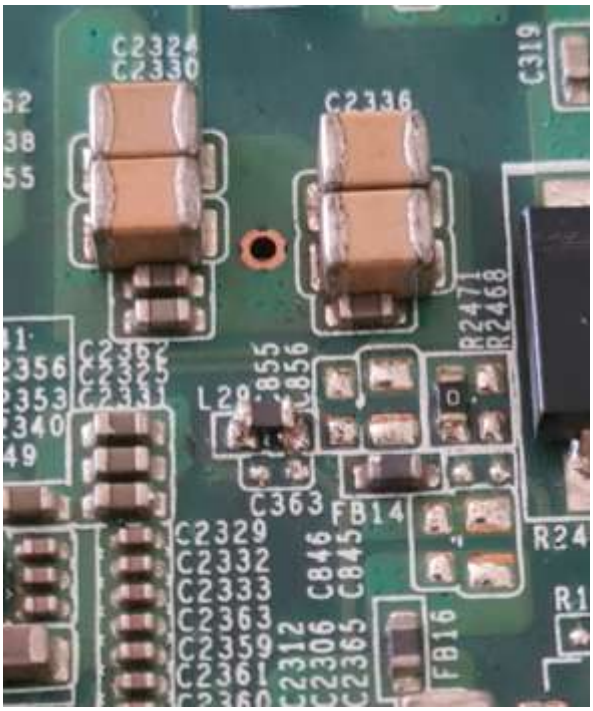


Figure 58: L29 welding

And then weld the resistor and the crystal:

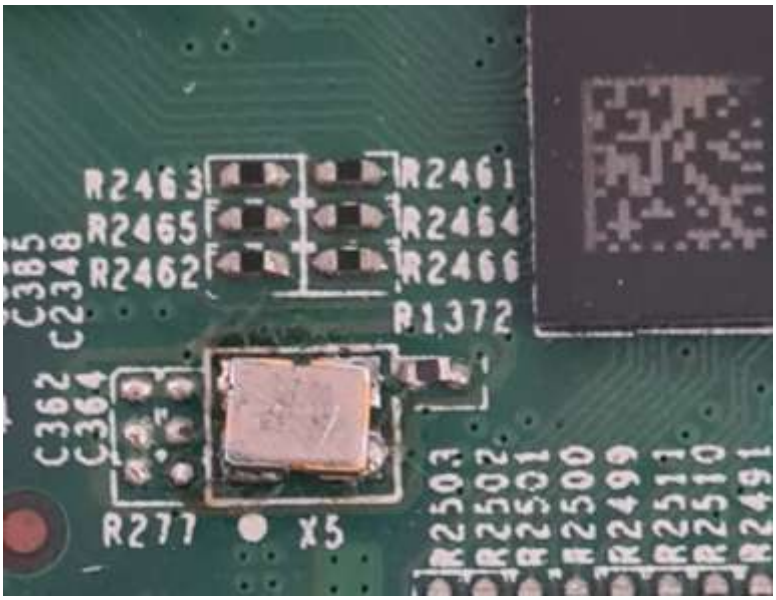


Figure 59: X5 and resistor welding

I haven't tried yet to weld C362 and C364 capacitors. If I discover problems due to this, I will try to weld them.

10. **Implementing reset via board supervisor chip.** There is no reset button on the board, but there is a power supervisor chip, available for a button. All that is needed is a button normal open switching to ground. I haven't tried it yet but it looks as if there is just enough room to solder one on the board.
    We only must to remove R2336 and solder there one end of the reset push button. The other end we need to put on GND.

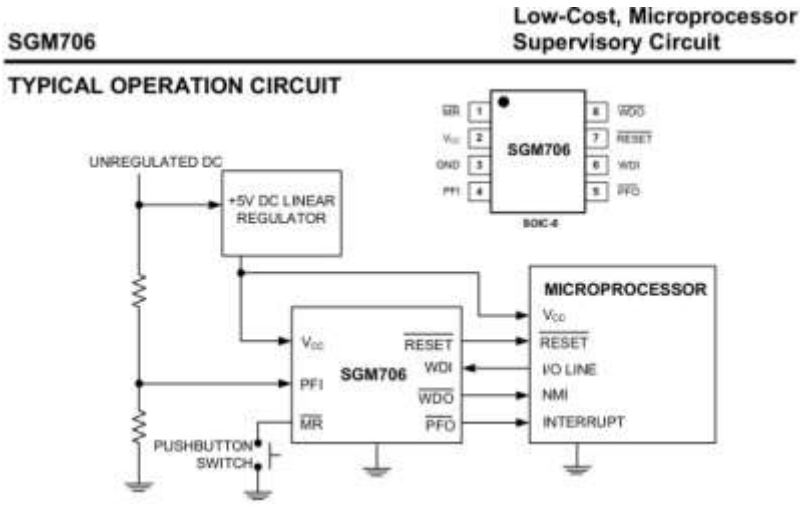


Figure 60: Supervisor datasheet

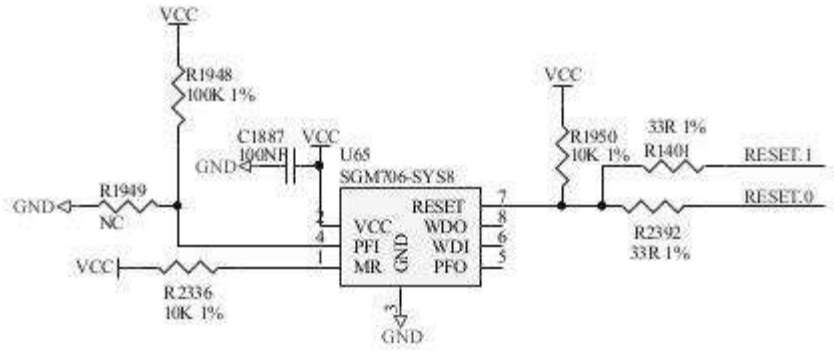


Figure 61: Supervisor schematic

Figure 62: Example of button placement

MODIFICATION STEPS: Remove R2336:

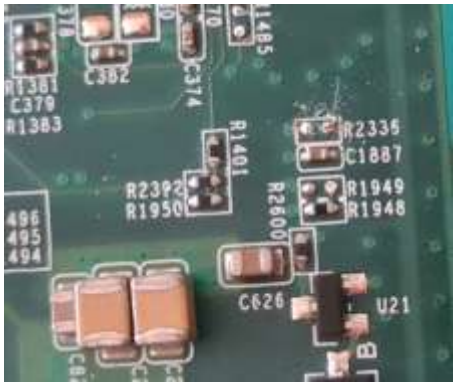


Figure 63: Reset welding 1 of 4

Weld one side of the switch to the pin 1 of U65:

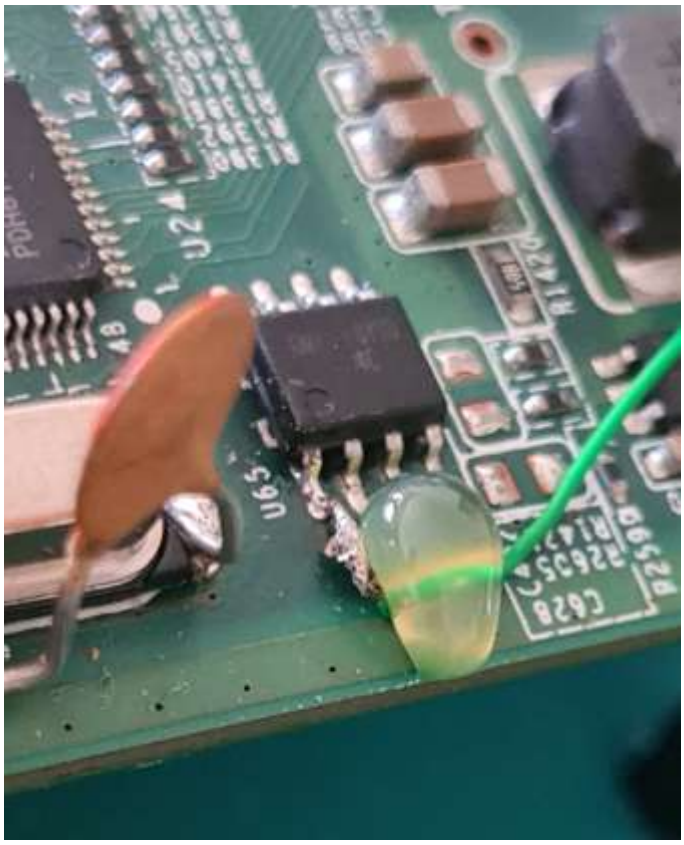


Figure 64: Reset welding 2 of 4

The other end to GND on unpopulated IC U23:

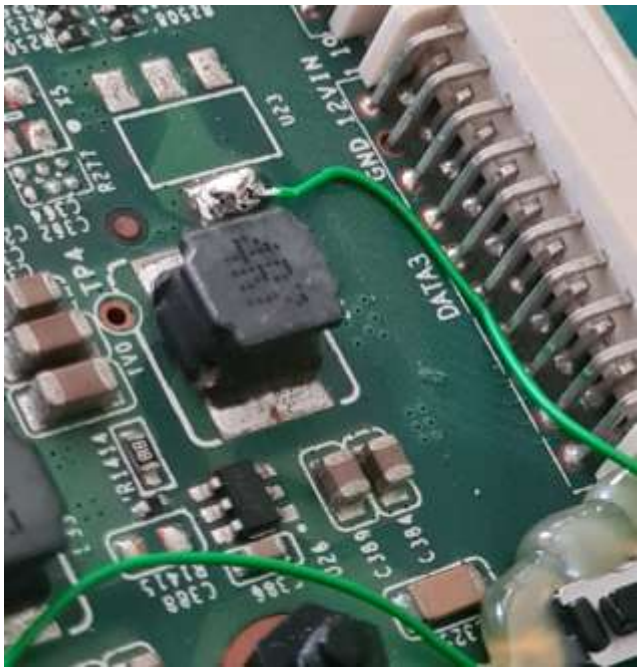


Figure 65: Reset welding 3 of 4

And then with hot melt, place the switch in some place.

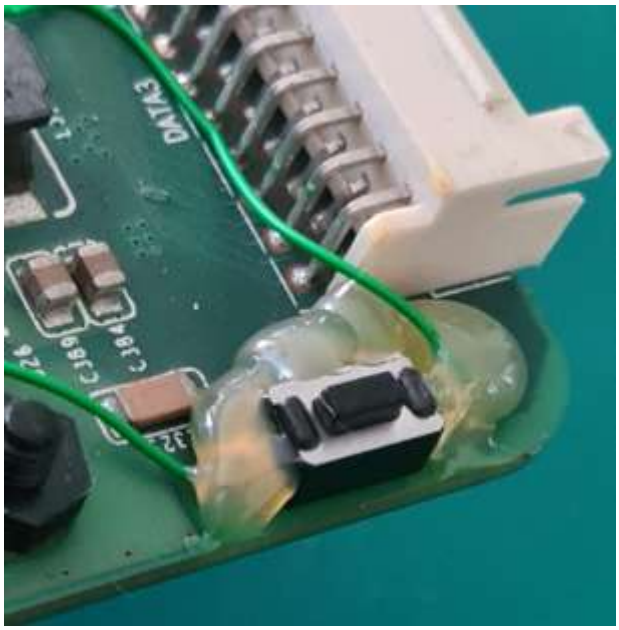


Figure 66: Reset welding 4 of 4

11. **RTC circuit.**

Also, I have an idea what the X3 chip + Y2 crystal might be used for - based on the Y2 footprint which allows cylindrical crystals, it could be an RTC. X3 connects to A15 & D13, these are I2C, the chip pinout matches DS1307 RTC. The fact that the RTC looks to be powered from Vcc on pin 3, this will limit its usefulness, it should have a battery. The PCF8523 SOIC-8 can use 1V to 5V5 supply.

So, exploring this possibility, we can stick a battery holder over there. In order to get this to work, we should populate C538, C68, R1279, R120, R122, R2438 and R2439. My implementation of the circuit does not connect the interrupt pin. But there is a way to map it if you want the functionality. You can populate R2240 with 1K resistor, departing from C71 pad (the other end of C71 is GND) and going to the and of R2240 with a wire (the other end of R2240 is VCC). Then you can weld a wire on R117 (the end who has continuity with R2240) and wire it to R1485 (XOUT0) which was used to provide clock to the Ethernet IC, but we take the choice to populate with a crystal. In that way we will not have that GPIO lost, and we can have the interrupt function available.

As this require a lot of components, all of them 0402, the risk is too high if you are not a high experienced welder. So, I decided to buy a PCF8523 module and expose I2C pins.

MODIFICATION STEPS: I choose to expose the I2C port with pins to use it later, on a RTC module, or another I2C I want. Also, I exposed XOUT.0 from the R1485 removed for the ethernet IC crystal. Just wire wrapped the pads and welded into a SMD straight male 2.54mm pins.

Studied were to weld:

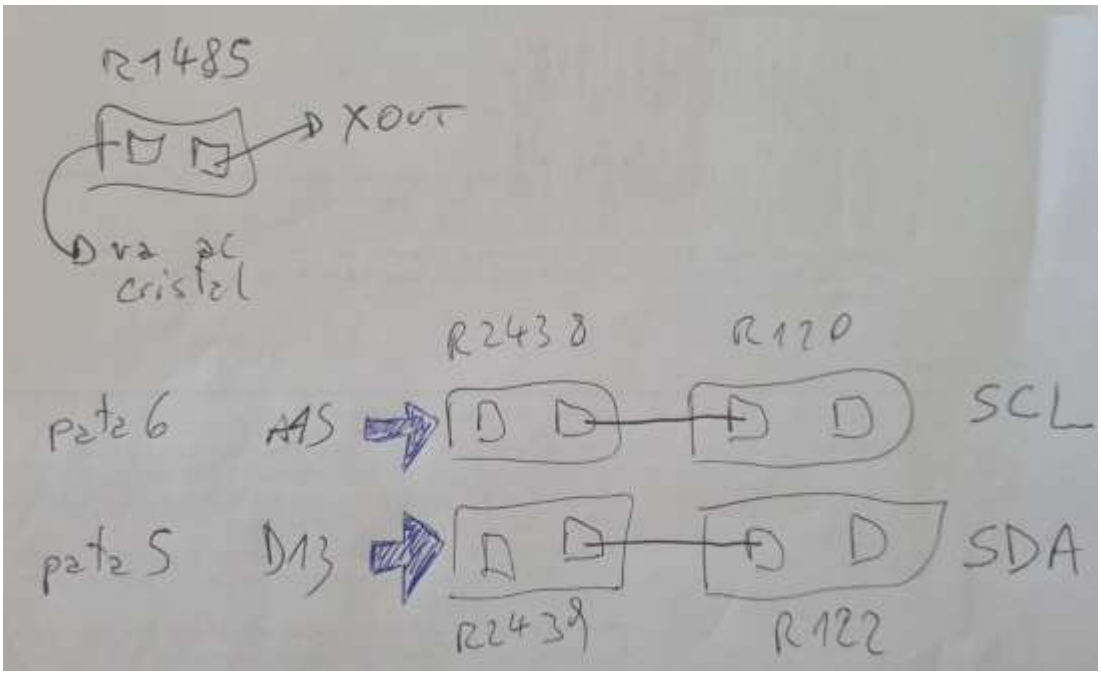


Figure 67: Physical footprint RTC

So, I took SMD pins

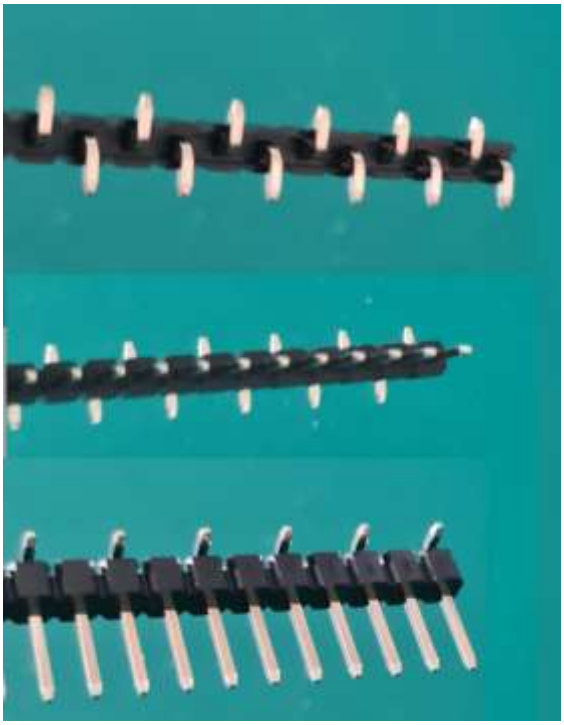


Figure 68: SMD header example

And welded 3 wire wraps.

Figure 69: Welded SMD pins to wirewrap

Then, the XOUT 0 goes to the right pad of the R1485 resistor seeing it with R1485 on the top:

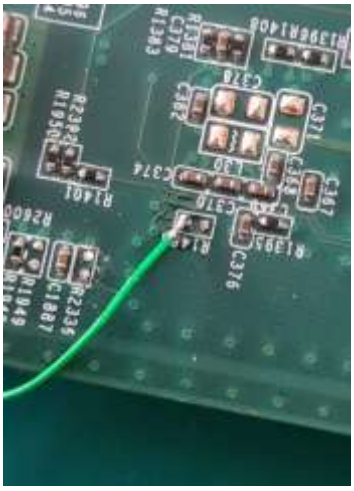


Figure 70: R1485 wiring

Then, the other two wire wraps go to R2438 and R2439.

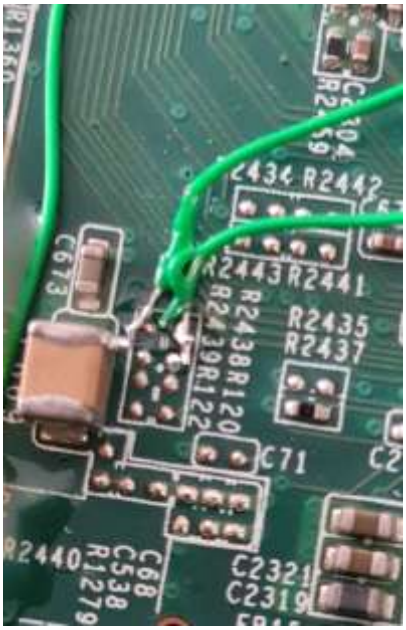


Figure 71: R2438 wiring

Then secure everything with hot melt

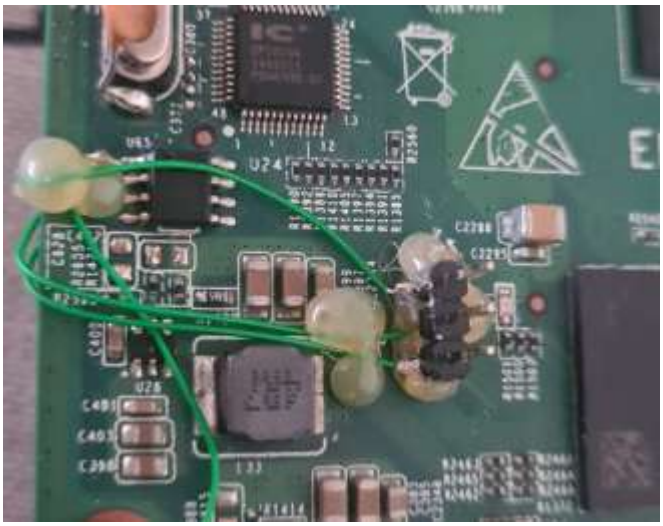


Figure 72: I2C pin placement

Resulting pinout

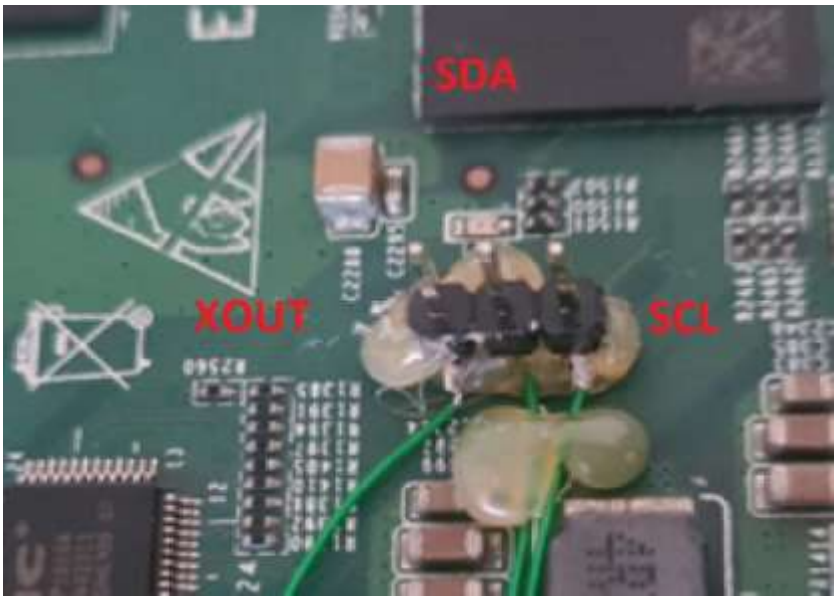


Figure 73: I2C pinout

12. **Solution to use the XADC facility.**

I believe that there will be problems if anyone wanted to use the XADC inputs on the Zynq. The details for this are in Xilinx User Guide UG480. The problem is that there is no power connection to the VCCADC pin (J9) on the Zynq. This looks to be a design error or oversight. To allow this to work a **clean** connection to a 1.8V supply should be connected to C327 on the pin nearest C2350.

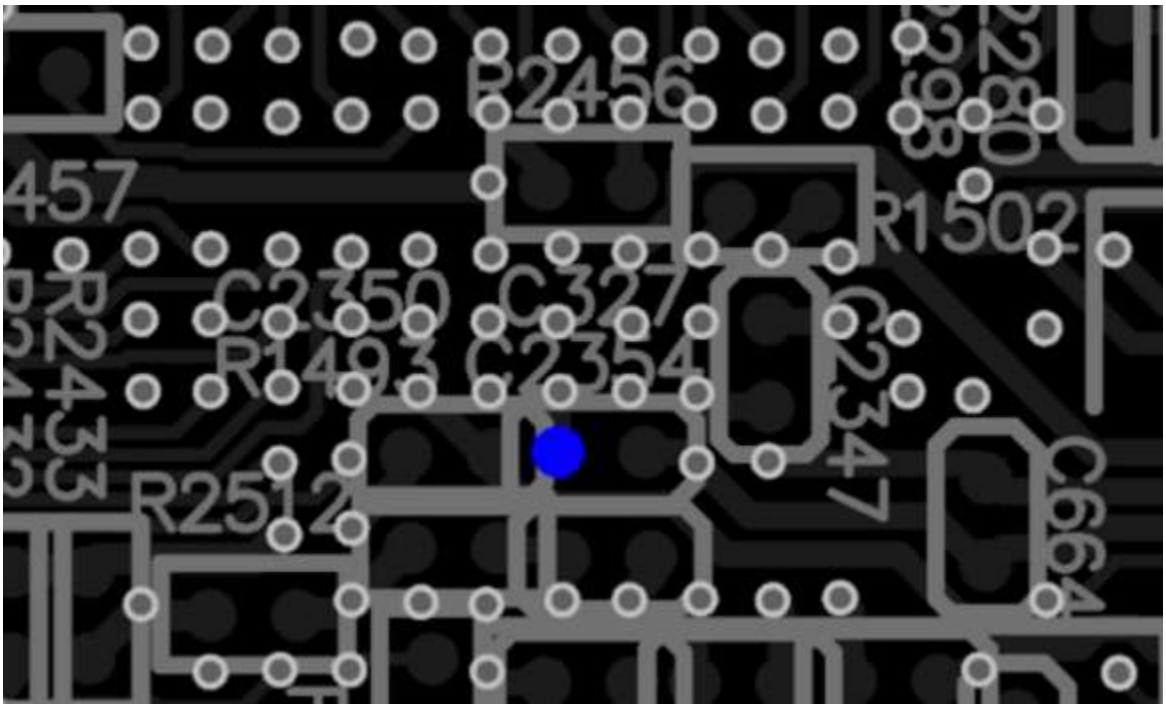


Figure 74: Pad location for XADC

Analytical drawings found VCCADC_0 missed 1.8V:

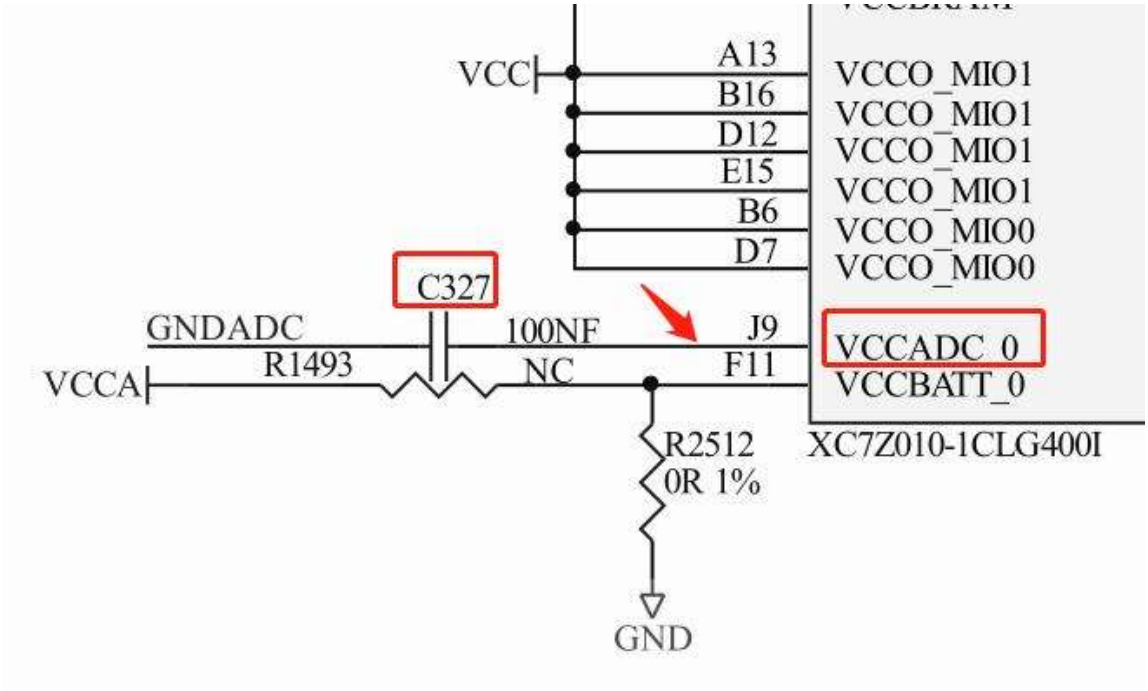


Figure 75: XADC schematics

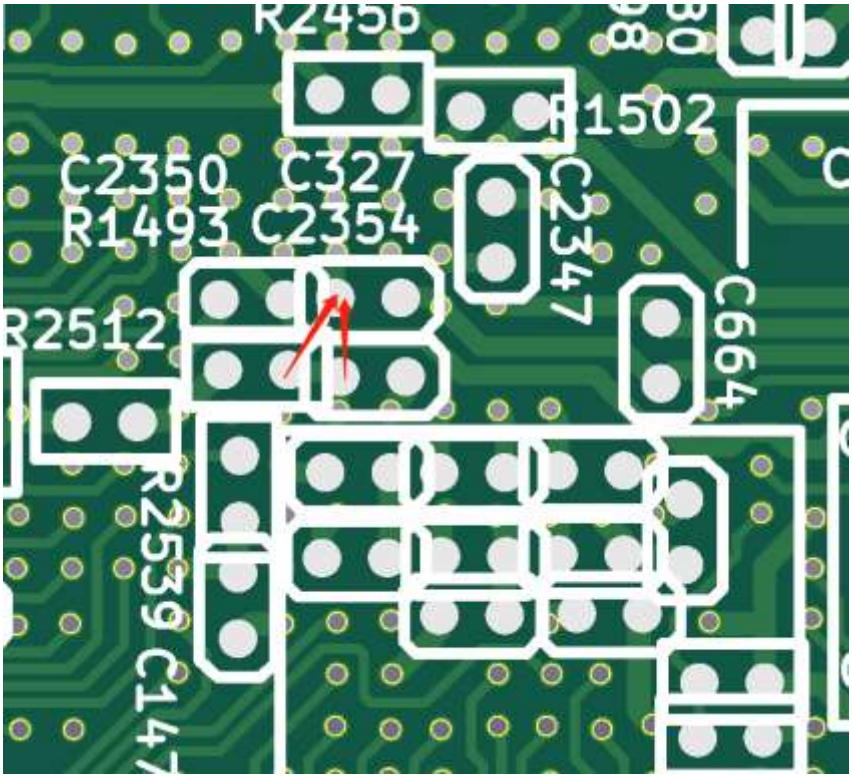


Figure 76: PCB pad location

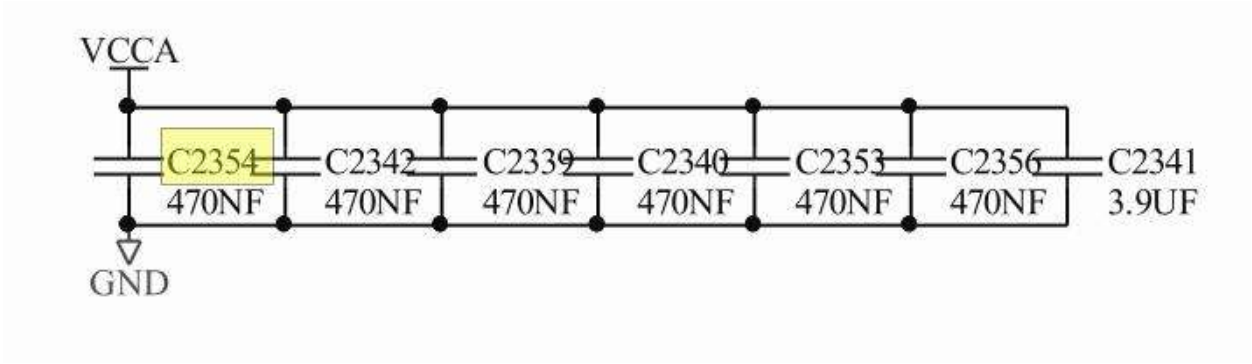


Figure 77: Near power line to use for XADC

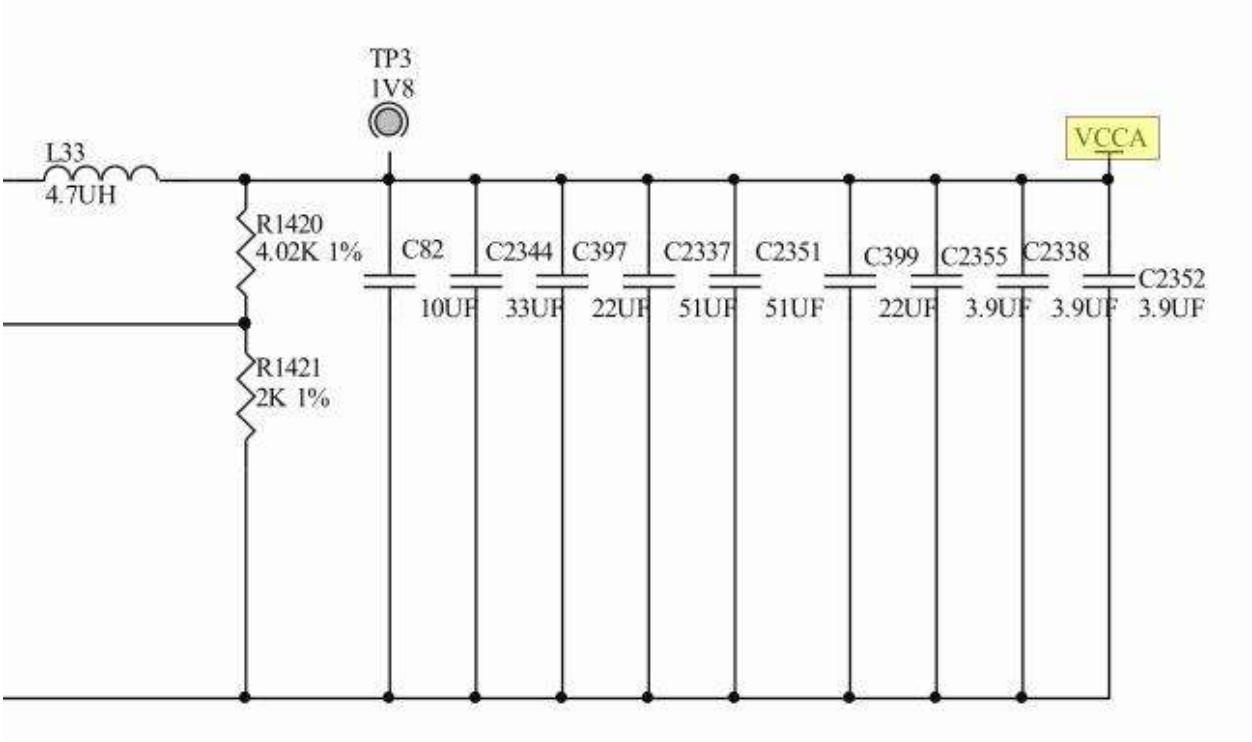


Figure 78: Near power line to use for XADC 2

MODIFICATION STEPS: Weld across C327 and C2354 with a wire wrap.

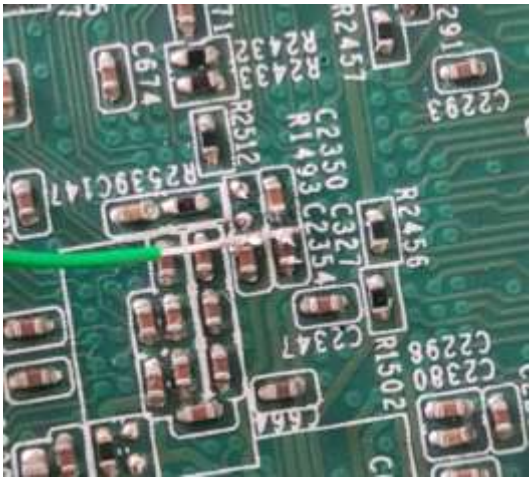


Figure 79: XADC welding 1 of 2

Then cut the wire wrap.

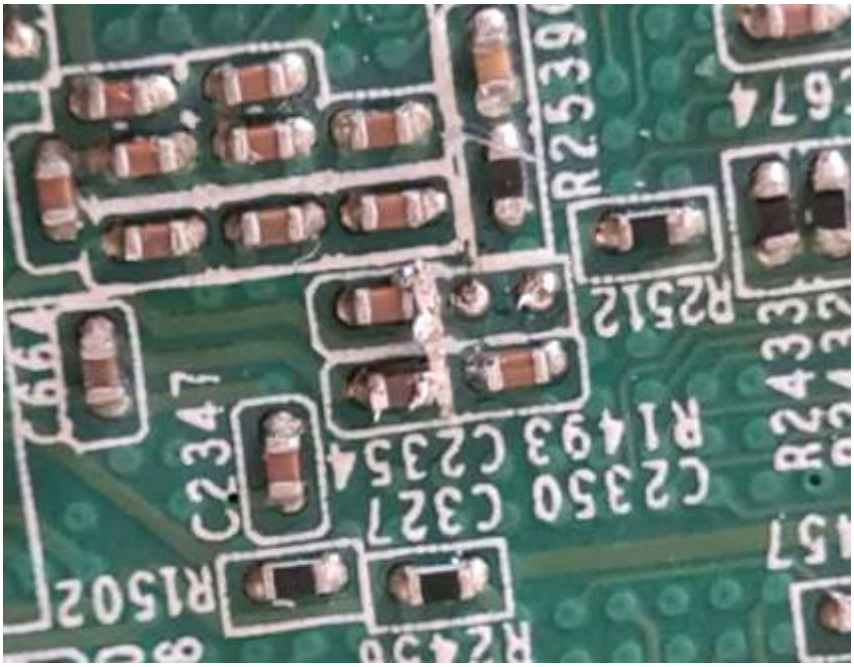


Figure 80: XADC welding 2 of 2

In addition, there will be no power to the VREFP pin (L9). This is not required as an internal reference can be used. Alternatively, a 1.25V reference voltage could be supplied to C664 on the pin furthest from the data connectors, but this will require to sacrifice a I/O pin. As there is no 1.25V on the board, I decided to use the internal reference.

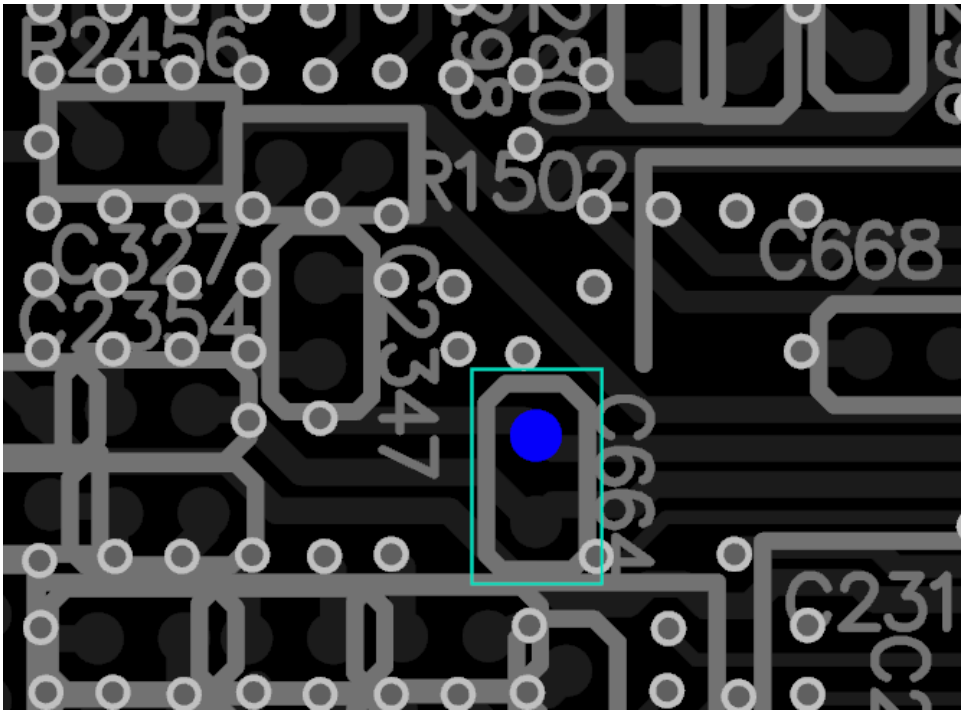


Figure 81: L9 missing pin

Interfaces pinout

This section will provide information about the connectors and the pinout related to the external interfaces.

1. **J7 Serial interface pins**.

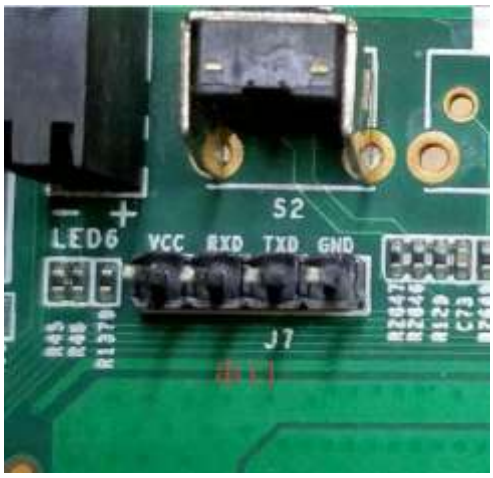


Figure 82: J7 serial interface pinout

Matching connector: Any 2.54mm dupont connector.

2. **J8 JTAG interface pins**.

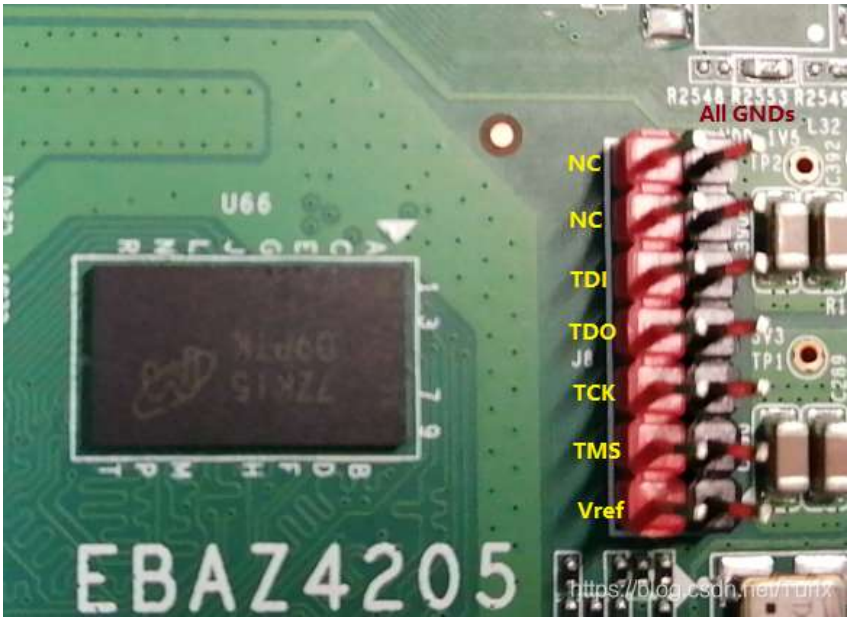


Figure 83: J8 JTAG pinout

Matching connector: Any 2.54mm dupont connector or 2x7 header.

3. Connector DATA1

Figure 84: Data 1 connector

Pin	Signal Name	Zynq pin
1	VCC-DATA1	N/A
2	VCC-DATA1	N/A
3	GND	N/A
4	GND	N/A
5	DATA1-5	A20
6	DATA1-6	H16
7	DATA1-7	B19
8	DATA1-8	B20
9	DATA1-9	C20
10	NC	N/A
11	DATA1-11	H17
12	GND	N/A
13	DATA1-13	D20
14	DATA1-14	D18
15	DATA1-15	H18
16	DATA1-16	D19
17	DATA1-17	F20
18	DATA1-18	E19
19	DATA1-19	F19
20	DATA1-20	K17

Table 3: Data 1 pinout

Matching connector: Any 2mm female header.

4. Connector DATA2

Figure 85: Data 2 connector

Pin	Signal Name	Zynq pin
1	VCC-DATA2	N/A
2	VCC-DATA2	N/A
3	GND	N/A
4	GND	N/A
5	DATA2-5	G20
6	DATA2-6	J18
7	DATA2-7	G19
8	DATA2-8	H20
9	DATA2-9	J19
10	NC	N/A
11	DATA2-11	K18
12	GND	N/A
13	DATA2-13	K19
14	DATA2-14	J20
15	DATA2-15	L16
16	DATA2-16	L19
17	DATA2-17	M18
18	DATA2-18	L20
19	DATA2-19	M20
20	DATA2-20	L17

Table 4: Data 2 pinout

5. Connector DATA3

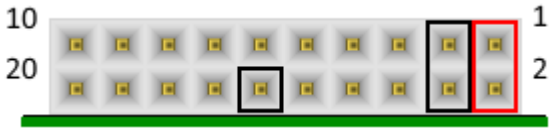


Figure 86: Data 3 connector

Pin	Signal Name	Zynq pin
1	VCC-DATA3	N/A
2	VCC-DATA3	N/A
3	GND	N/A

4	GND	N/A
5	DATA3-5	M19
6	DATA3-6	N20
7	DATA3-7	P18
8	DATA3-8	M17
9	DATA3-9	N17
10	NC	N/A
11	DATA3-11	P20
12	GND	N/A
13	DATA3-13	R18
14	DATA3-14	R19
15	DATA3-15	P19
16	DATA3-16	T20
17	DATA3-17	U20
18	DATA3-18	T19
19	DATA3-19	V20
20	DATA3-20	U19

Table 5: Data 3 pinout

NOTE: The connector follows a distribution called “PMOD” you can find more information about this in https://digilent.com/reference/pmod/start (Digilent site)

6. Connector J3

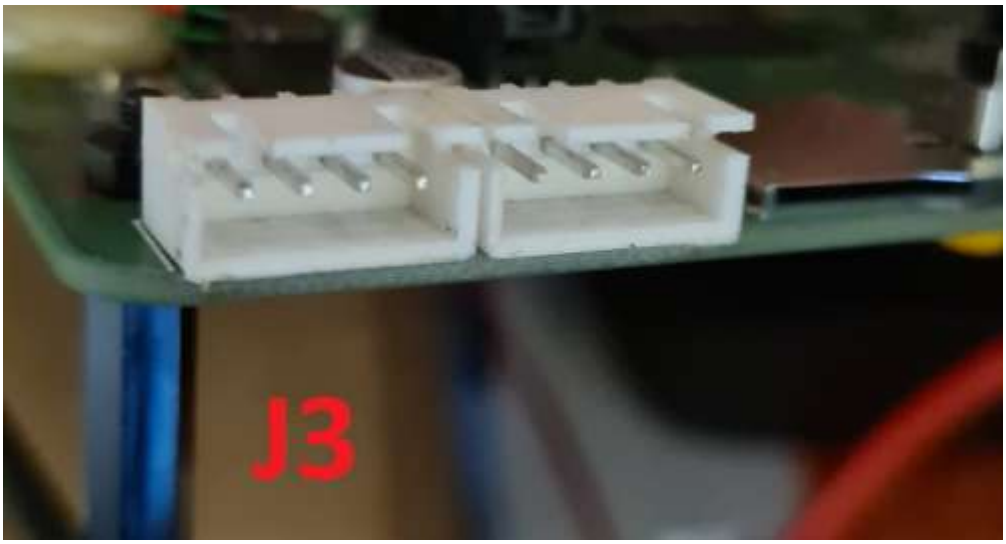


Figure 87: J3 connector

Pin	Signal Name	Zynq pin
1	GND	N/A
2	VCC	N/A
3	SPEED	V13
4	PWM	U12

Table 6: J3 connector pinout

Matching connector: Any 2.54mm JST 4 positions connector.

I used this wire (JST female to Dupont female) in order to be able to use the outputs as best as I can.

Figure 88: J3/J5 wire (JST to Dupont)

7. Connector J5

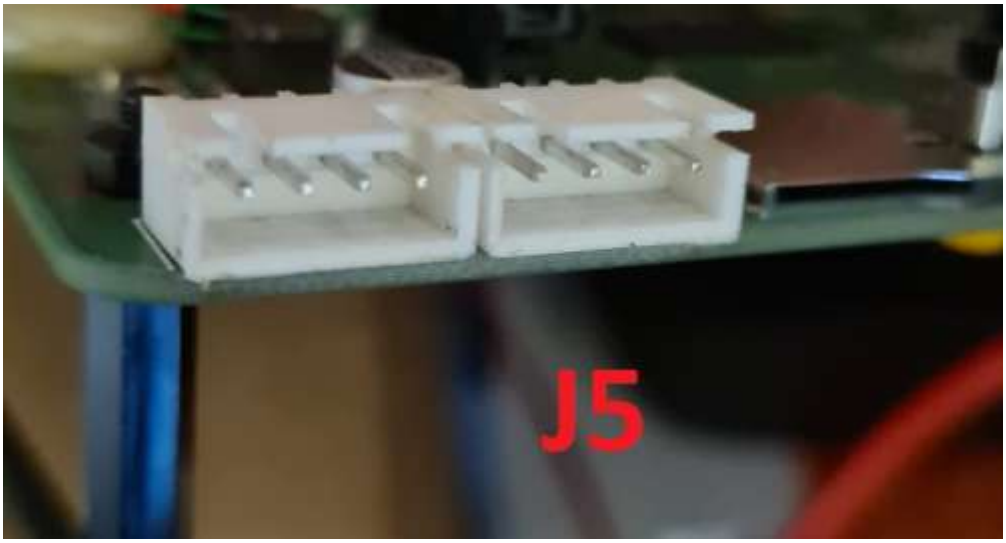


Figure 89: J5 connector

Pin	Signal Name	Zynq pin
1	GND	N/A
2	VCC	N/A
3	SPEED	V15
4	PWM	V12

Table 7: J5 connector pinout

Matching connector: Any 2.54mm JST 4 positions connector.

I used the same wire for J3 connector.

8. Connector J4

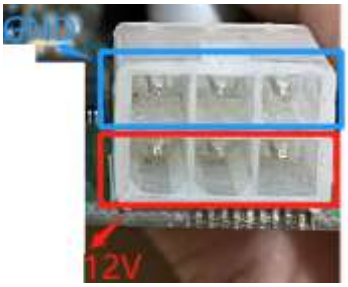


Figure 90: J4 connector

Pin	Signal Name

1	VCC
2	VCC
3	VCC
4	GND
5	GND
6	GND

Table 8: J4 connector pinout

Matching connector: Molex 0039012060 connector, with Molex 0039000038 socket crimp. (18-24 AWG)

I make this wire to use it.

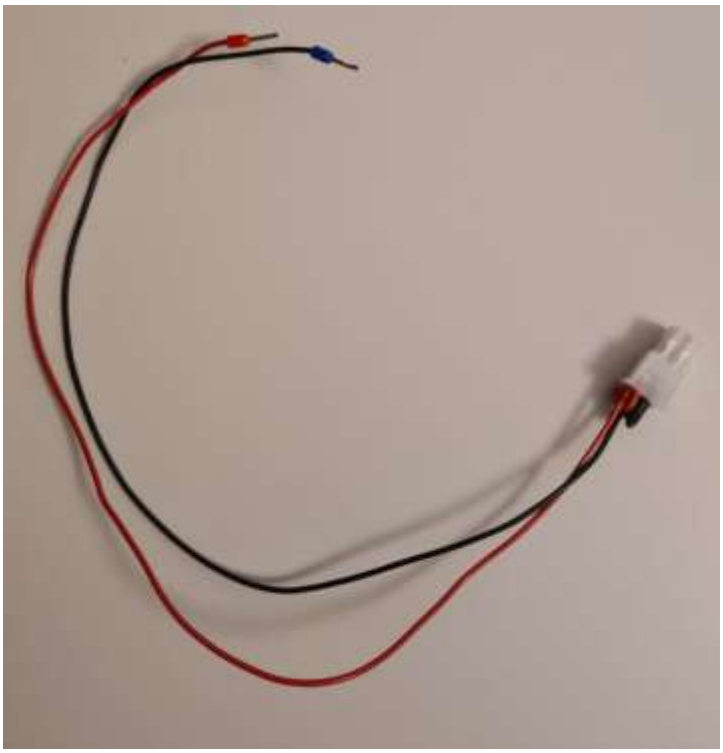


Figure 91: J4 wire (power)

Connector detail:

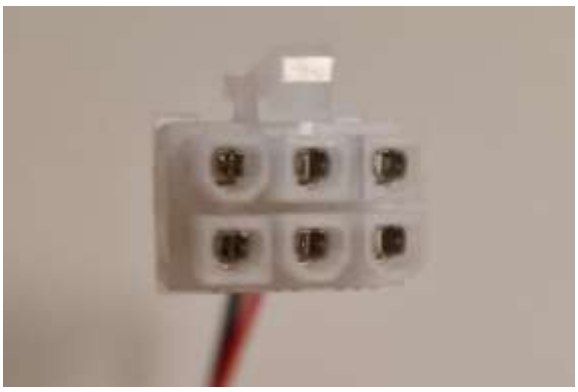


Figure 92: J4 matching connector

9. **LEDs**

There are 6 LEDs welded on the card, and they are:

1. Connected between VCC and GND, used to indicate the power supply LED4;
2. Connected to the DONE_0 pin of ZYNQ, used to indicate the LED1 that the ZYNQ configuration is completed;
3. Connect to the red and green indicator lights of W13 and W14 on ZYNQ (not corresponding yet) LED6;
4. 2 network port indicators connected to IP101GA.

There are only 2 available for customization.

First turn on

Depending on what you have ordered, usually have two options, the board as it is (with boot device NAND Flash) or with modifications so that the boot mode is set to SD-Card. I ordered both PCBs with SD-Card, UART and JTAG Header already mounted.

If the Schottky Diode D24 was not mounted (my case), you can only power the board using DATA1, DATA2 or DATA3 port. Voltage 5V-12V is OK, 400mA @ 12V is necessary. The pin distance of these three ports is 2.0mm, which is not common as 2.54mm.

If D24 was mounted, we can power the board using J3, J4 or J5. For convenience, J3 or J5 is recommended, which was originally for FAN power. I didn't verify J3 or J5, because I make this wire to power on the board.

Starting to test the hardware received (uSD version)

With the version of the PCB that I received; the booting is set by default to uSD. A simple USB to UART module connected to the pins on J7 (see above) will provide us a debugging interface. The configuration is 115200bps, 8 bit, no parity, stop bit, no flow control.

In order to quickly test the board, I followed these steps to flash a uSD card and test the board before doing any soldering.

[THANKS TO "RAINLY" FOR THIS INSTRUCTIONS]

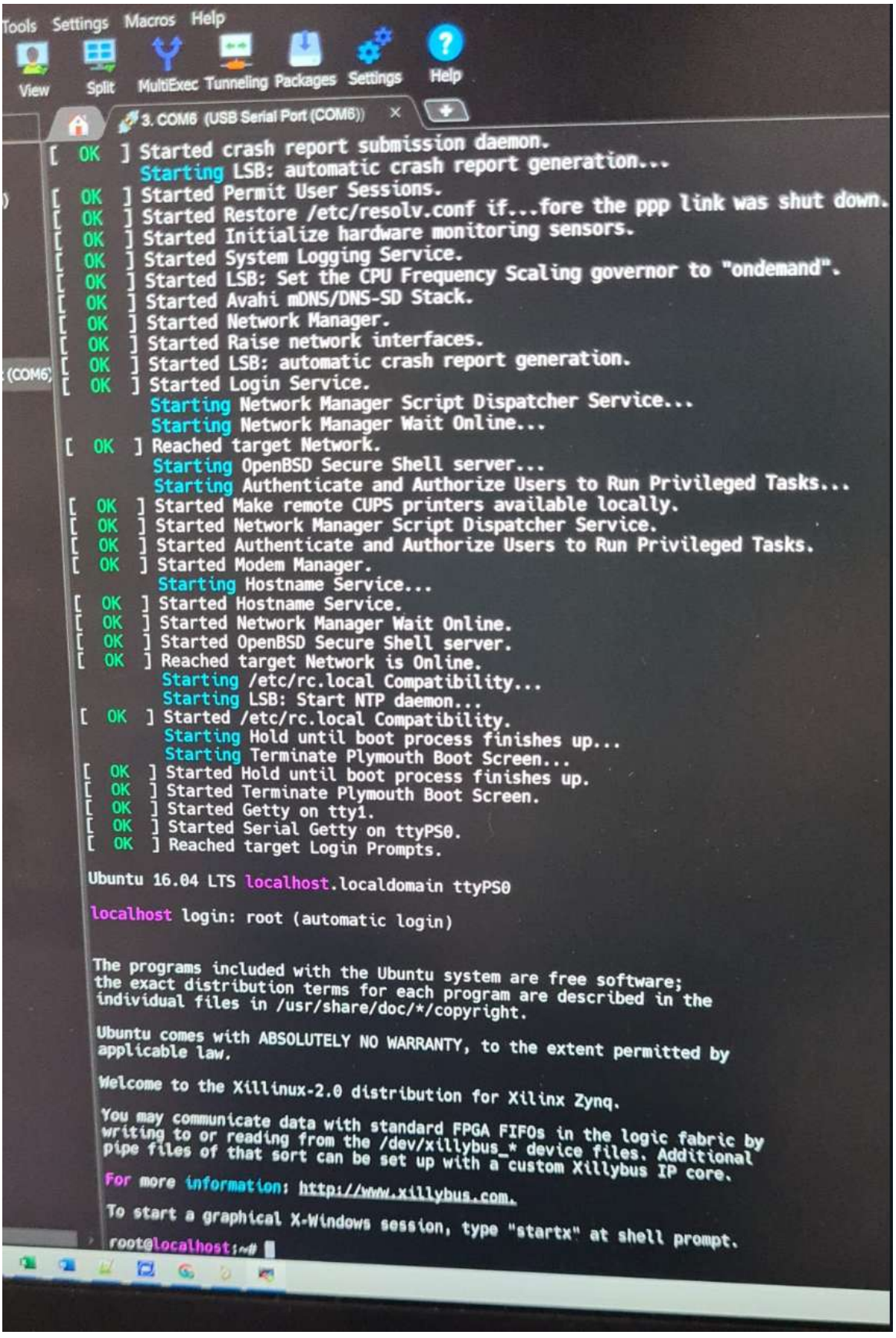
Instructions for using the image:

1. You have 3 files: fat32-file.7z, xillinux-2.0.img.gz and win32diskimage.

2. Extract fat32-file.7z and xillinux-2.0.img.gz.
3. Prepare a uSD card 8gb or above.
4. Use win32diskimagerto write xillinux-2.0.img on the uSD card.

5. Delete the files in the FAT partition and copy the files of fat32-file folder to the FAT partition.

Then power up the board and you should see the output on the serial console:


Tools Settings Macros Help
View Split MultiExec Tunneling Packages Settings Help
3. COM6 (USB Serial Port (COM6))
[OK] Started crash report submission daemon.
Starting LSB: automatic crash report generation...
[OK] Started Permit User Sessions.
[OK] Started Restore /etc/resolv.conf if...fore the ppp link was shut down.
[OK] Started Initialize hardware monitoring sensors.
[OK] Started System Logging Service.
[OK] Started LSB: Set the CPU Frequency Scaling governor to "ondemand".
[OK] Started Avahi mDNS/DNS-SD Stack.
[OK] Started Network Manager.
[OK] Started Raise network interfaces.
[OK] Started LSB: automatic crash report generation.
[OK] Started Login Service.
Starting Network Manager Script Dispatcher Service...
Starting Network Manager Wait Online...
[OK] Reached target Network.
Starting OpenBSD Secure Shell server...
Starting Authenticate and Authorize Users to Run Privileged Tasks...
[OK] Started Make remote CUPS printers available locally.
[OK] Started Network Manager Script Dispatcher Service.
[OK] Started Authenticate and Authorize Users to Run Privileged Tasks.
[OK] Started Modem Manager.
Starting Hostname Service...
[OK] Started Hostname Service.
[OK] Started Network Manager Wait Online.
[OK] Started OpenBSD Secure Shell server.
[OK] Reached target Network is Online.
Starting /etc/rc.local Compatibility...
Starting LSB: Start NTP daemon...
[OK] Started /etc/rc.local Compatibility.
Starting Hold until boot process finishes up...
Starting Terminate Plymouth Boot Screen...
[OK] Started Hold until boot process finishes up.
[OK] Started Terminate Plymouth Boot Screen.
[OK] Started Getty on tty1.
[OK] Started Serial Getty on ttyPS0.
[OK] Reached target Login Prompts.
Ubuntu 16.04 LTS localhost.localdomain ttyPS0
localhost login: root (automatic login)
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
Welcome to the Xillinux-2.0 distribution for Xilinx Zynq.
You may communicate data with standard FPGA FIFOs in the logic fabric by
writing to or reading from the /dev/xillybus_* device files. Additional
pipe files of that sort can be set up with a custom Xillybus IP core.
For more information: http://www.xillybus.com.
To start a graphical X-Windows session, type "startx" at shell prompt.
root@localhost:~#

In my case, ethernet it's not working, but I don't know why (Drivers/Xtal problem?). With this result, I consider the board tested.

Starting to test the hardware received (NAND version)

I have tried to start with NAND but all I got it's a message in the console stating "This is a SM driver example" and nothing happens. Maybe the provider deleted the original software.

However, I will copy paste the instructions given by xjtuecho on the github wiki page in case you have the original software:

Reset the root password of built-in linux

Use HyperTerminal or Putty to connect the TTL port, TXD RXD and GND is enough, VCC is not necessary. A simple USB to UART module connected to the pins on J7 to provide us a debugging interface. The configuration is 115200bps, 8 bit, no parity, stop bit, no flow control.

Power the Board, hit d to enter the u-boot shell. Now use the following commands to reset the root password.

```
setenv nandboot "echo Copying Linux from NAND flash to RAM... && nand info && run
nandroot;nand read 0x100000 0x2220000 0x300000 && fpga loadb 0 0x100000 0x300000 &&
nand read ${kernel_load_address} 0x300000 ${kernel_size} && nand read
${devicetree_load_address} 0x800000 ${devicetree_size}"
run nandboot
setenv bootargs 'console=ttyPS0,115200 root=/dev/mtdblock6 rootfstype=jffs2 noinitrd
rw rootwait reboot=cold,hard emergency init=/bin/sh'
bootm ${kernel_load_address} - ${devicetree_load_address} init=/bin/sh
passwd
```

Shut down the BTC miner program: After logging on with root, execute commands below to disable the BTC miner program.

```
mv /etc/rcS.d/S95cgminer.sh /etc/rcS.d/K95cgminer.sh
reboot
```

If you want to set an IP address, edit configuration file: /etc/network/interfaces, and add contents below.

```
auto eth0
iface eth0 inet static
address 192.168.1.205
netmask 255.255.255.0
gateway 192.168.1.1
dns-nameservers 192.168.1.1
```

Enable ethernet: ifup eth0. Check ethernet status: ethtool eth0

Now you can use ssh root@192.168.1.205 to connect the built-in linux. Of course, you can use other IP address according to your network configuration.

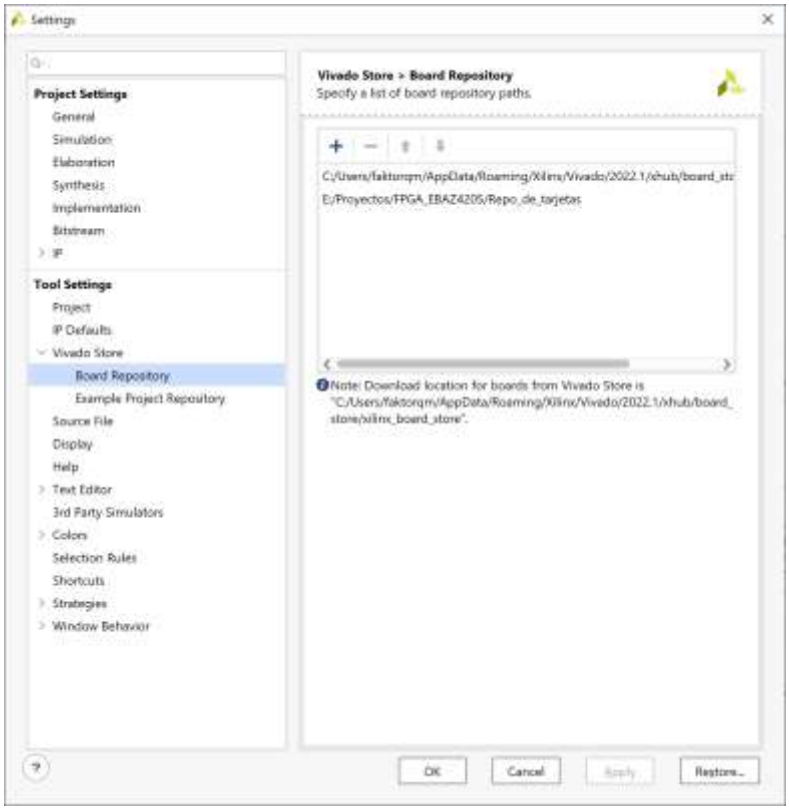
NOTE: Please check the JTAG section to establish a “hello world” program on the NAND generating the FSBL and all the required things.

Starting to test the hardware received (JTAG version)

Well, these steps are a little bit longer than the others one but will allow us to have a serial printing “hello world” project flashed via JTAG to the board to test. I will work on Vivado 2022.1.

Instructions to install board files:

1. Assuming you already have Vivado installed, we would need to install the board files before start. Download the files from author’s github repo: https://github.com/XyleMora/EBAZ4205/tree/main/Documents/Board%20files
2. Saved it on a folder that you can remember. (Remember to uncompress it if you downloaded compressed).
3. Open Vivado. Go to Tools -> Settings
4. Under the Tool settings sections, click on “Vivado Store”.
5. Then select “Board repository”. In the right panel you will see a default folder.
6. Click on “+” symbol to add a folder.
7. Select the folder where you put the downloaded files.



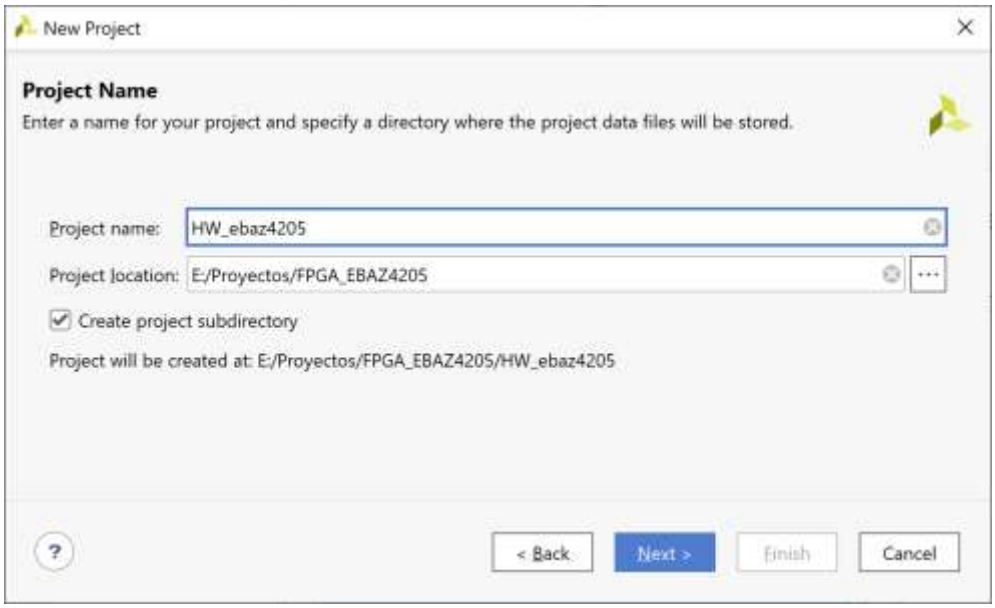
8. Click on Apply. Then on OK.

Instructions to create the project:

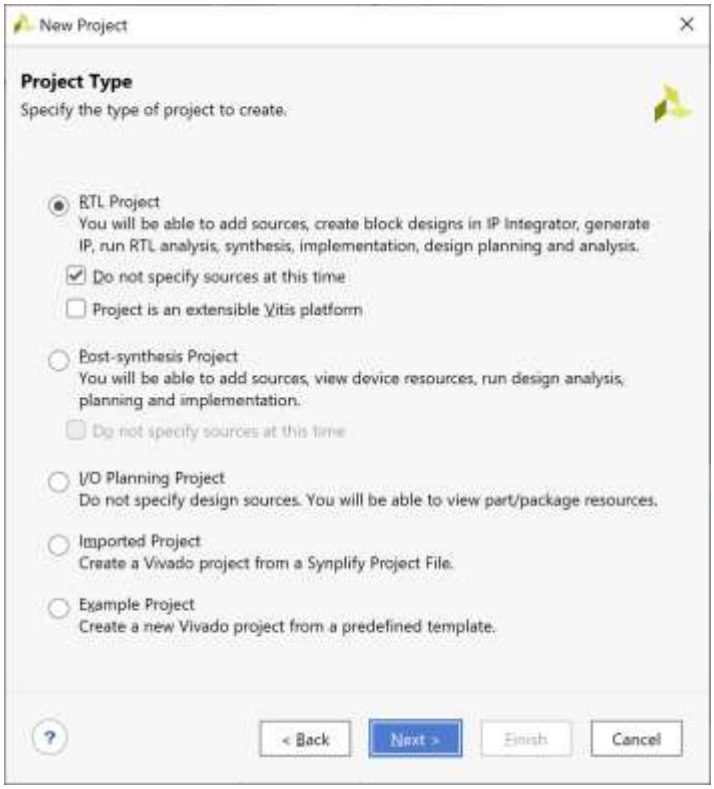
1. Start by opening Vivado and select “Create Project” to create a new project.

2. Click on “Next”

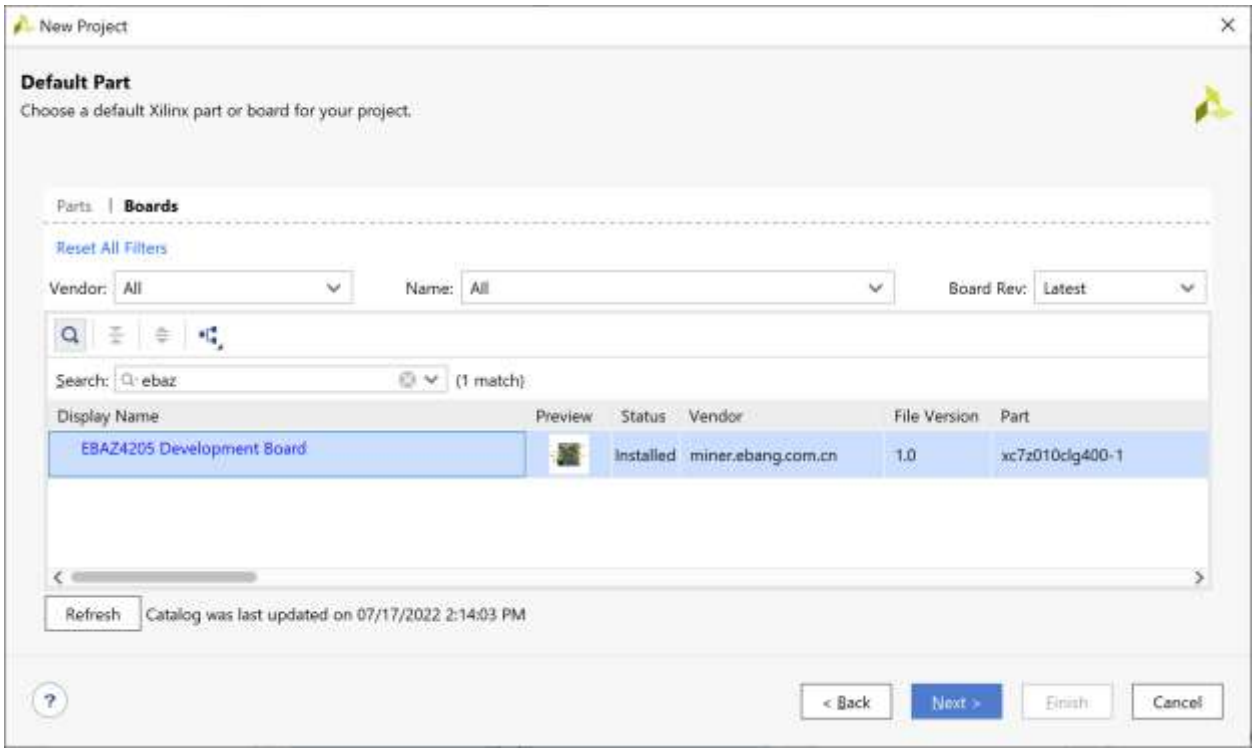
3. Select project name, I used HW_ebaz4205 and make sure to have “Create project subdirectory” selected



4. Select “RTL Project” as your project type and check “Do not specify sources at this time”. Also make sure to NOT click on “Project is an extensible Vitis platform”. This choice is intended for Linux platform only, does not allow bare metal projects. Using this option is more like “the old way”.

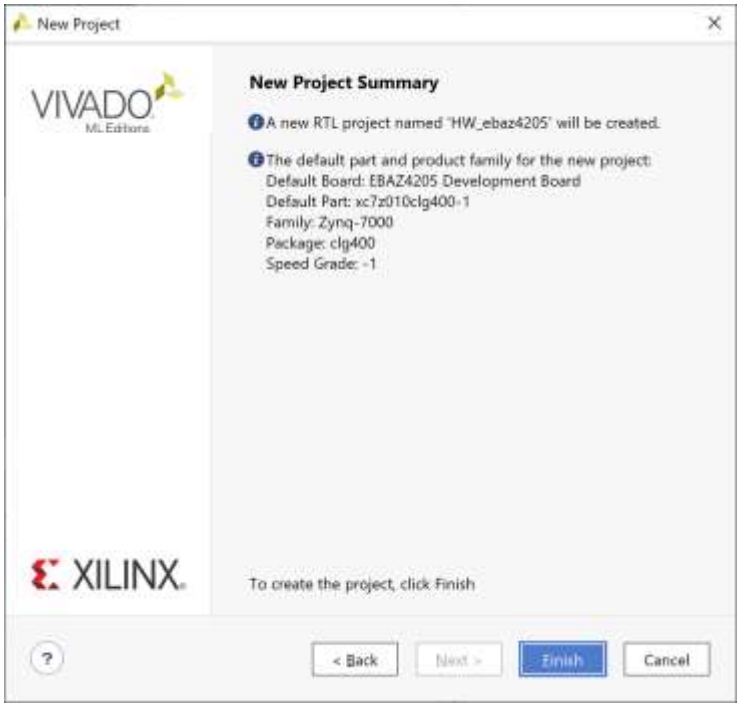


5. Select the board from the list and click next

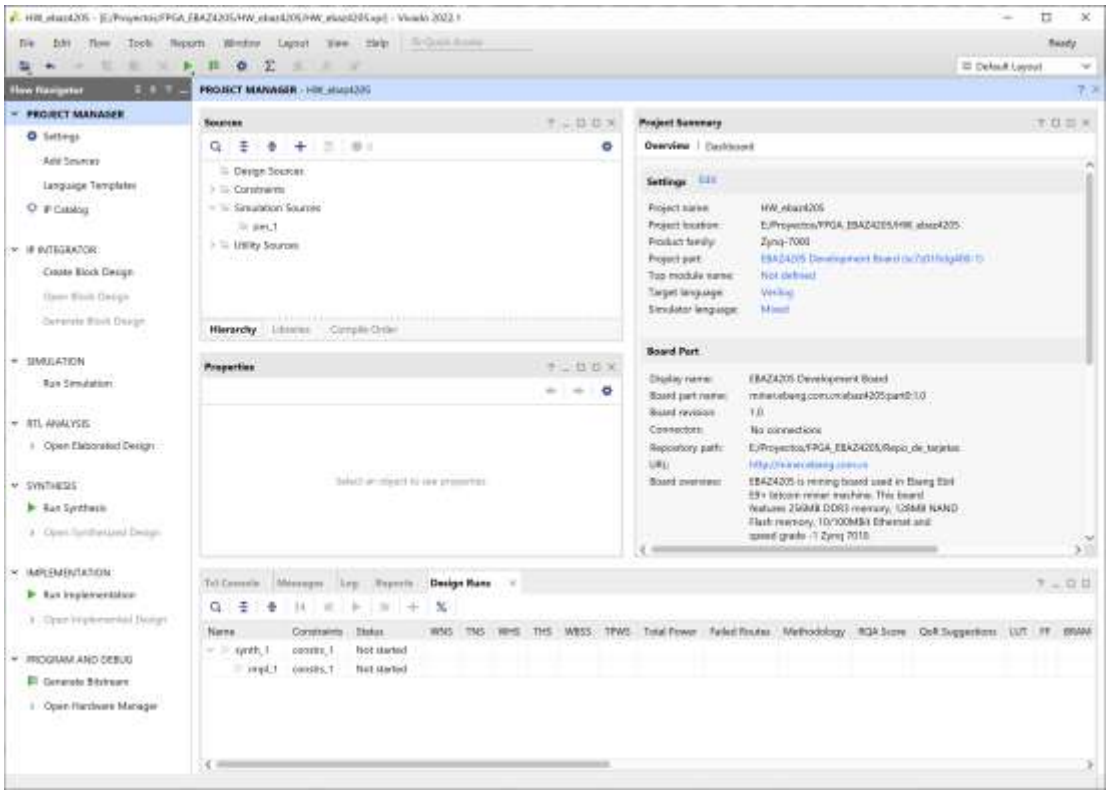


NOTE: Probably you would like to “refresh” the catalog. This will update from internet the board and part catalog.

6. Click on finish



7. You should see a window like this

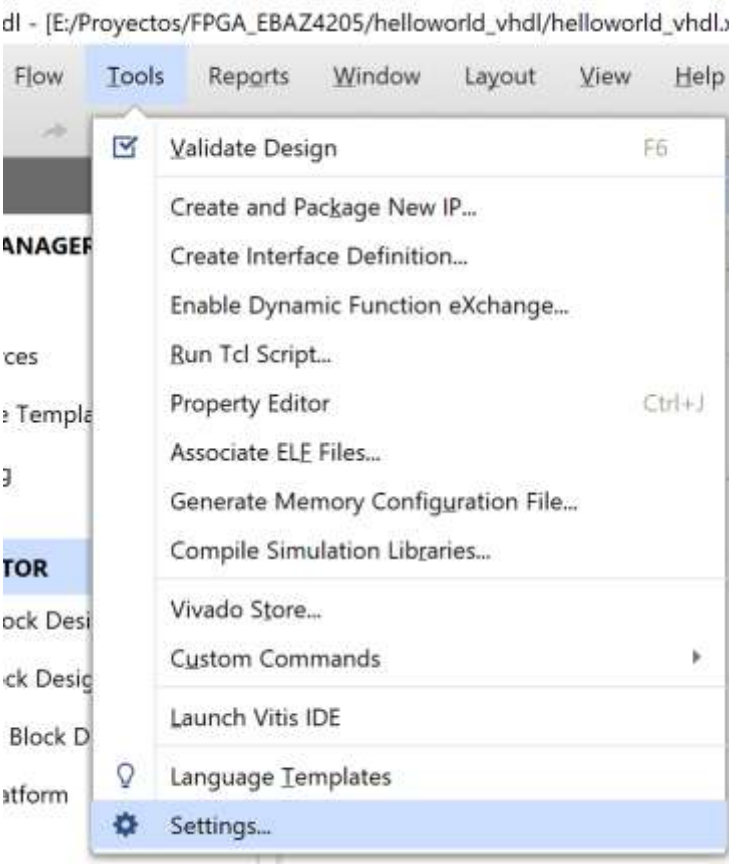


Instructions to begin:

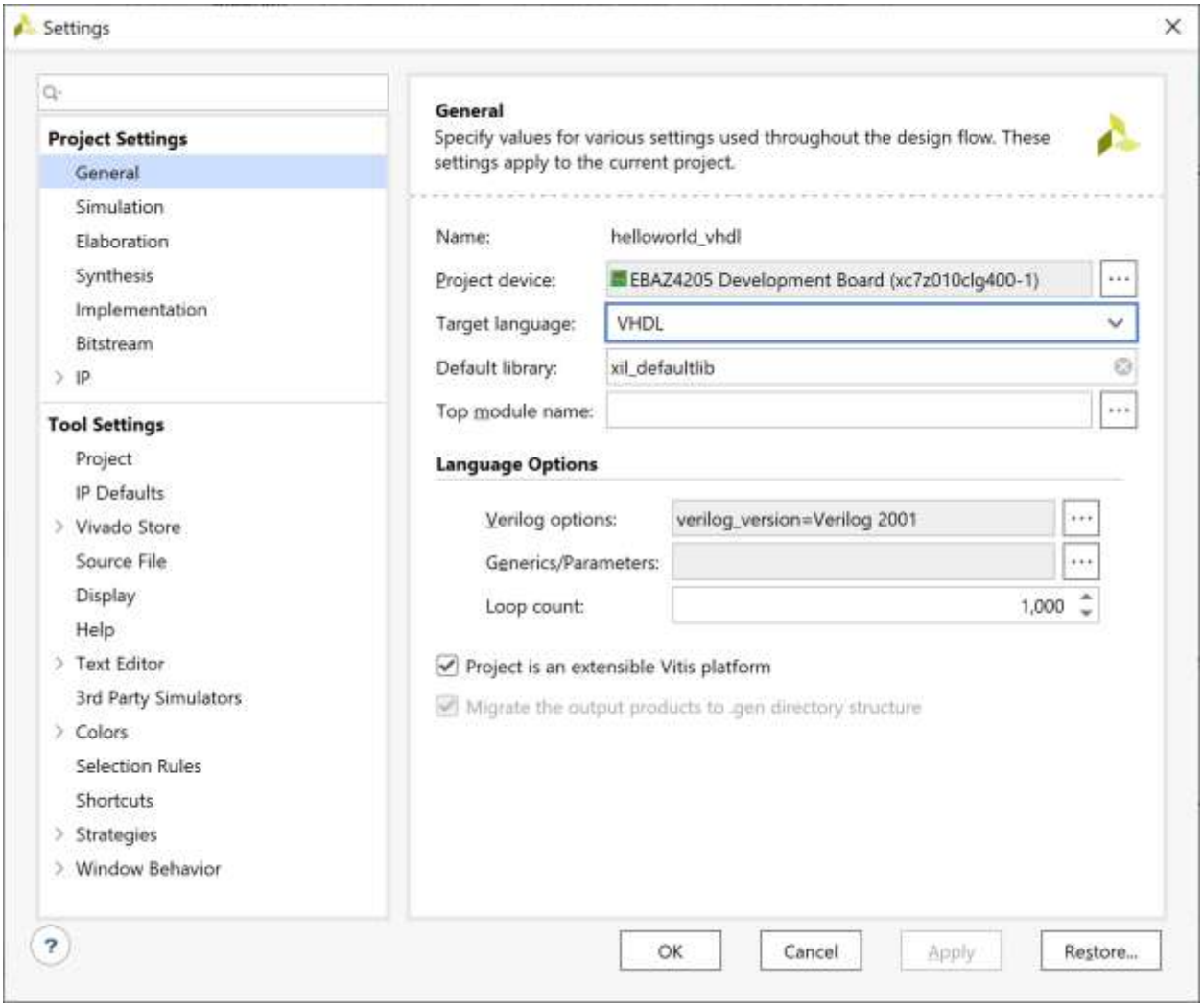
Create a Processing System with IP Integrator, Vivado development kit provides a graphical design and development tool - IP Integrator (IP Integrator), in which various functional modules (IP) can be easily inserted. It supports intelligent automatic connection of key IP interfaces, one-click IP subsystem generation, real-time DRC and other functions, which can help us quickly assemble complex systems and accelerate the design process.

Next, we will complete the construction of ZYNQ embedded system in IP integrator.

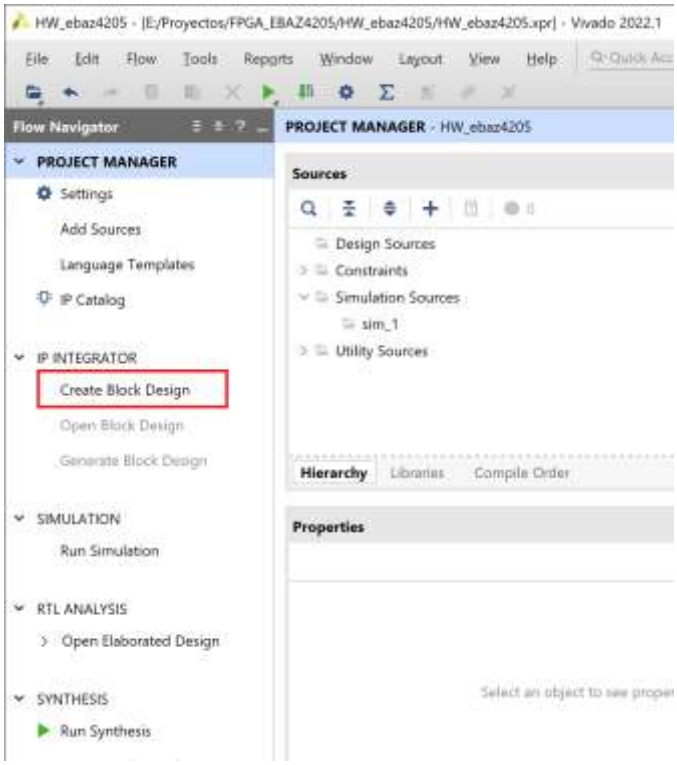
1. Click on Tools -> Settings…



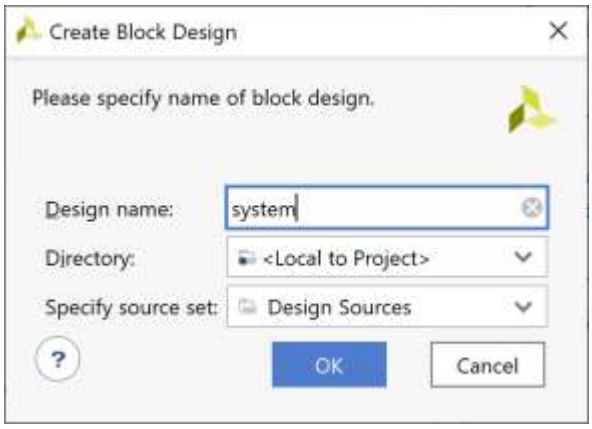
2. You will be presented with the General window, here I changed from Verilog to VHDL, because that's my favorite language. Press OK.



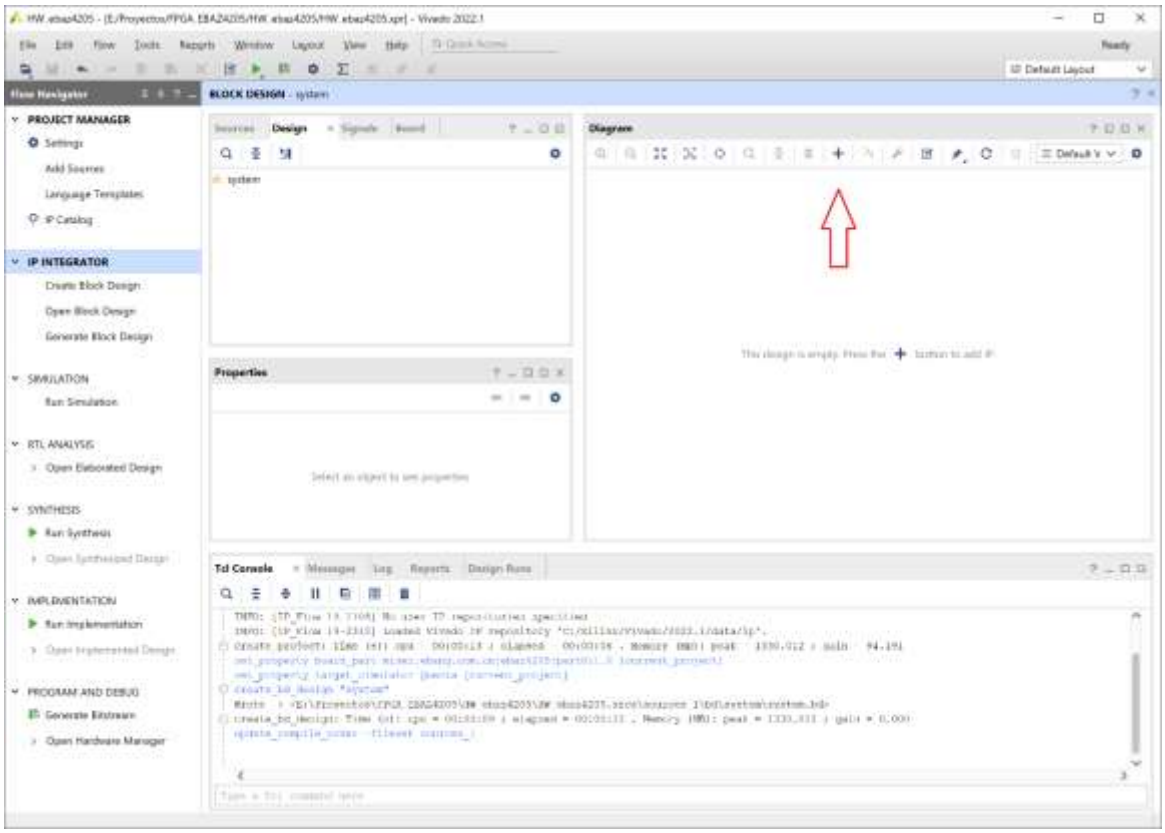
3. Click on the “Create Block Design” option under ‘IP INTEGRATOR” to start your design.



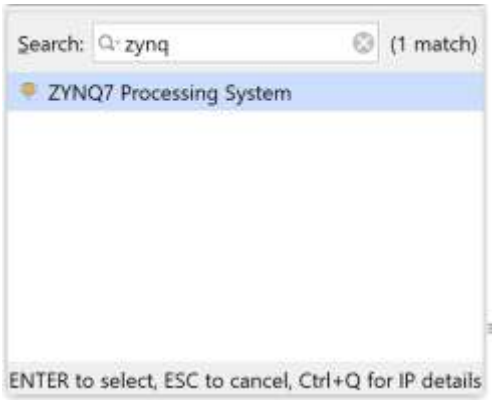
4. Give a name to your design (I will use “system”). Then click on “OK”.



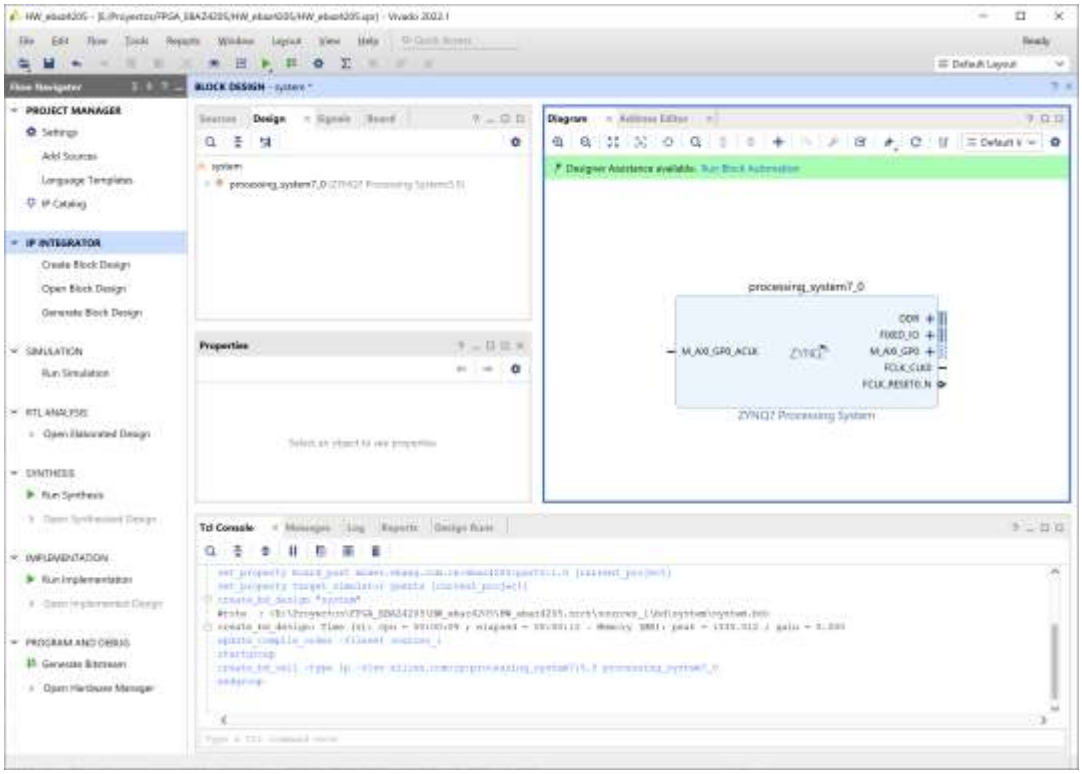
5. In the next window, click on “+”



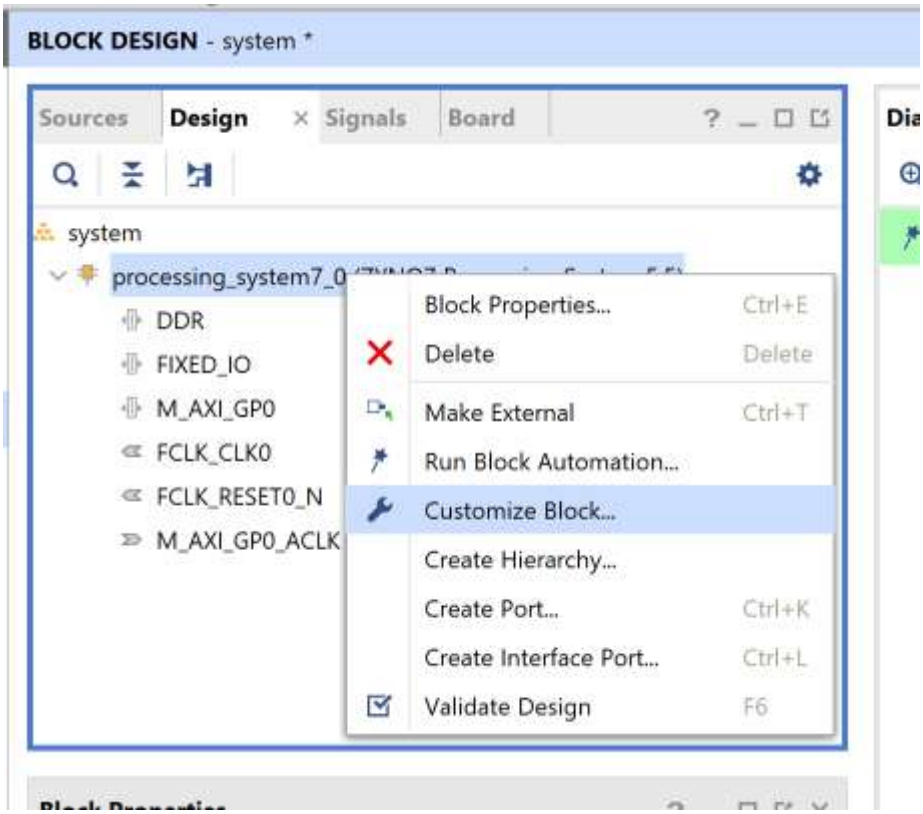
6. After opening the IP directory, type "zynq" in the search bar, and then double-click on "ZYNQ7 Processing System", to add it to the design.



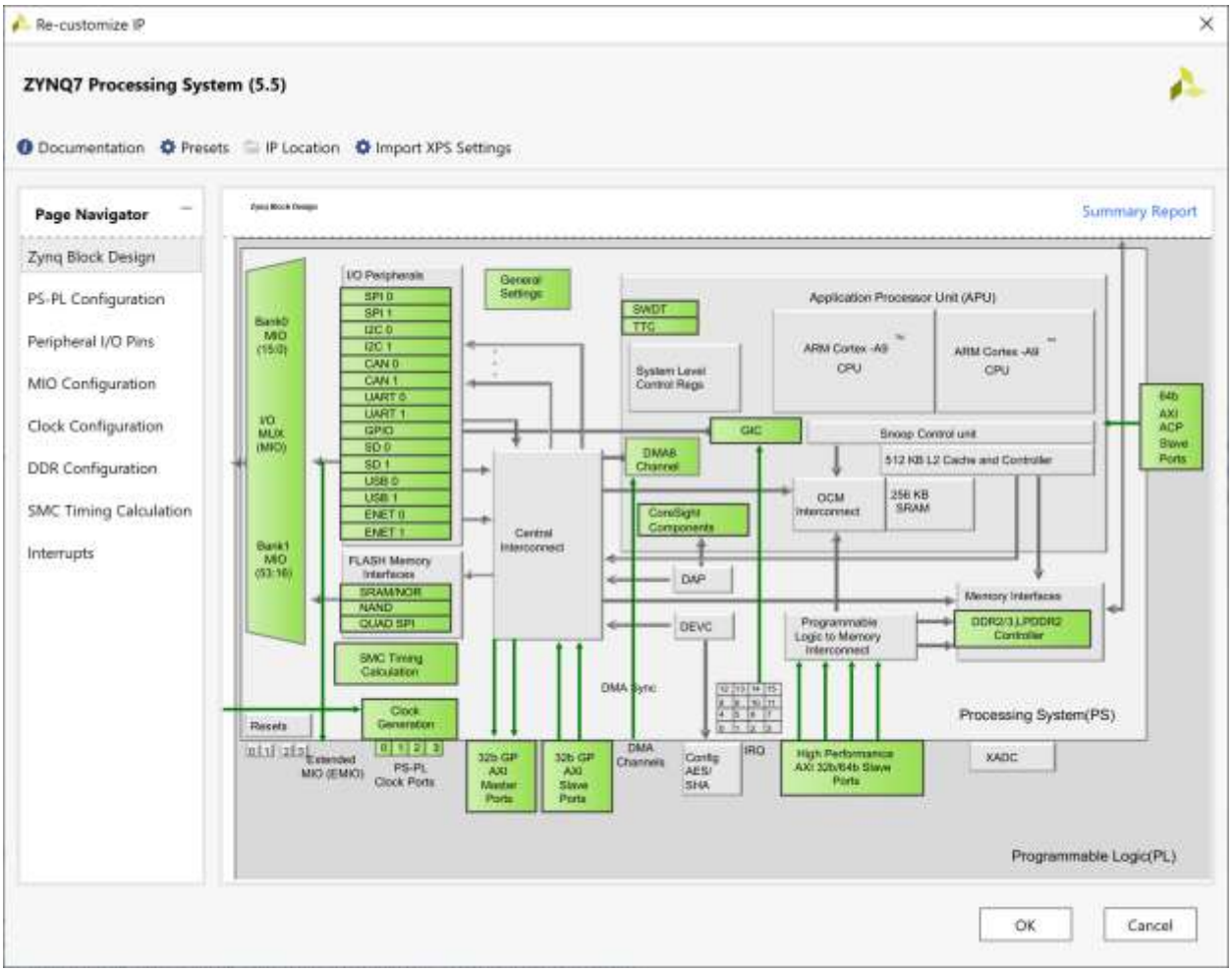
7. Then you should see this



8. Do right click on “processing_system7_0” and select “customize block”



9. Then you will see this window. The left side of the interface is the page navigation panel, and the right side is the configuration information panel.



10. Briefly introduction about each item on the left panel:

On the Zynq Block Design page, various configurable blocks of the Zynq processing system (PS) are displayed, in which the gray part is fixed, and the green part is configurable, which is configured according to the actual needs of the project. You can directly click various configurable blocks (highlighted in green) to enter the corresponding configuration page for configuration, or you can select the page navigation panel on the left to configure the system.

The PS-PL Configuration page enables configuration of PS-PL interfaces, including AXI, HP, and ACP bus interfaces.

The Peripheral IO Pins page allows you to select MIO/EMIO configurations for different I/O peripherals.

The MIO Configuration page can specifically configure MIO/EMIO for different I/O peripherals.

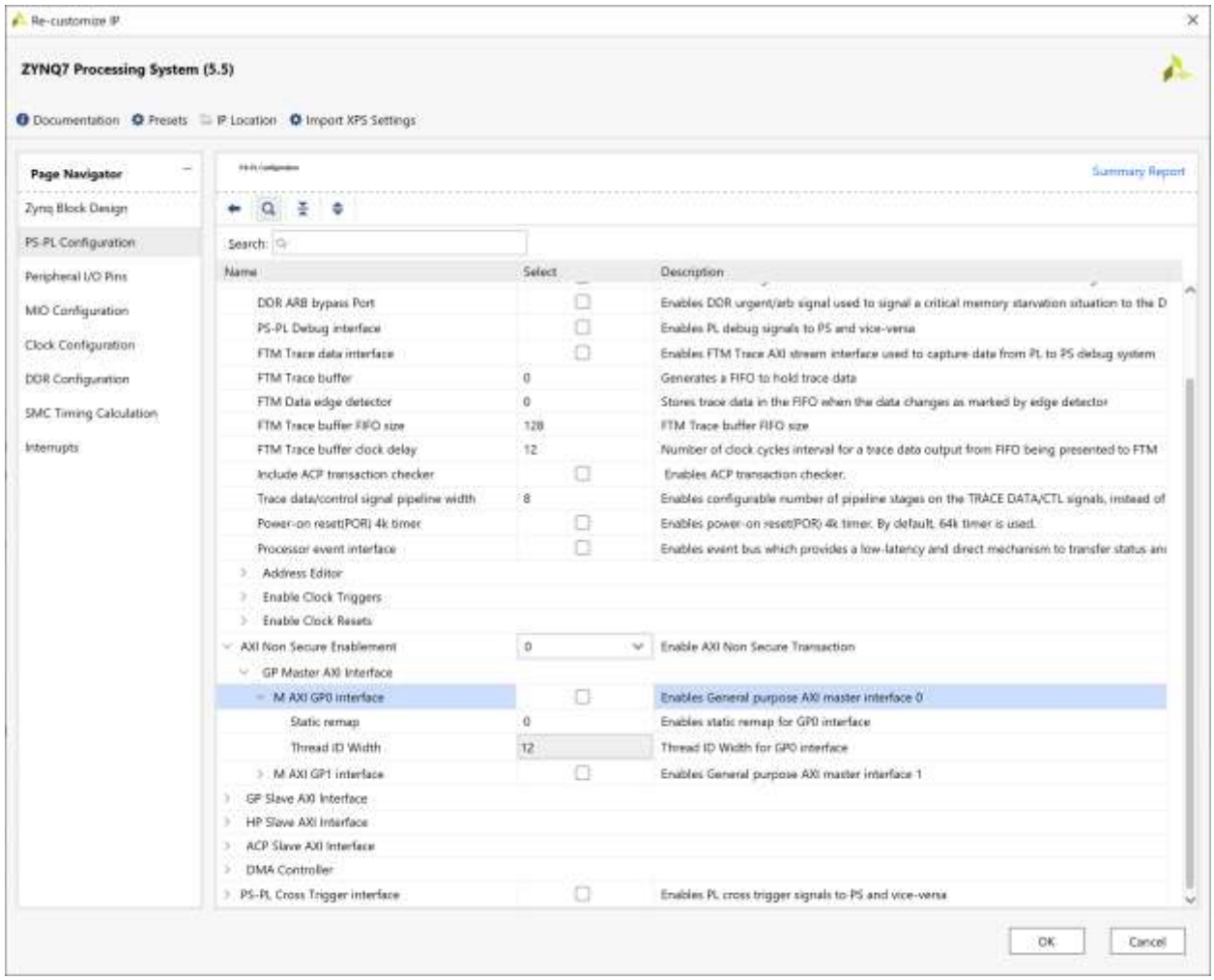
The Clock Configuration page is used to configure PS input clock, peripheral clock, DDR and CPU clock, etc.

The DDR Configuration page is used to set the DDR controller configuration information.

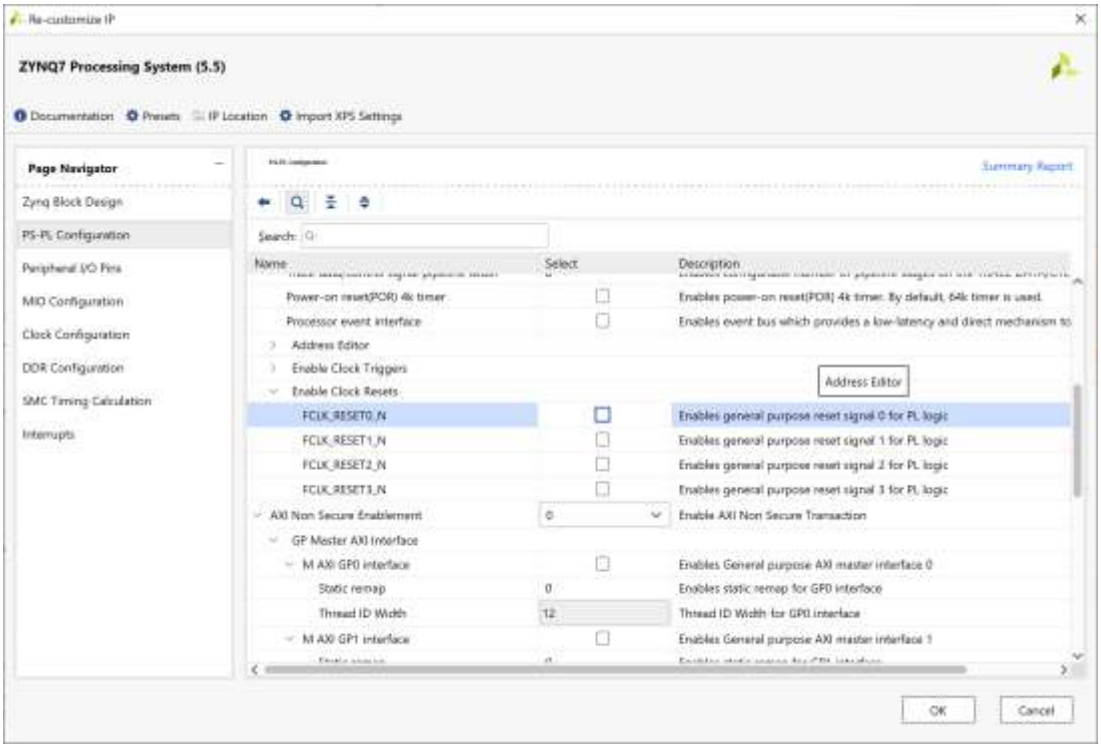
The SMC Timing Calculation page is used to perform SMC timing calculations.

The Interrupts page is used to configure PS-PL interrupt ports.

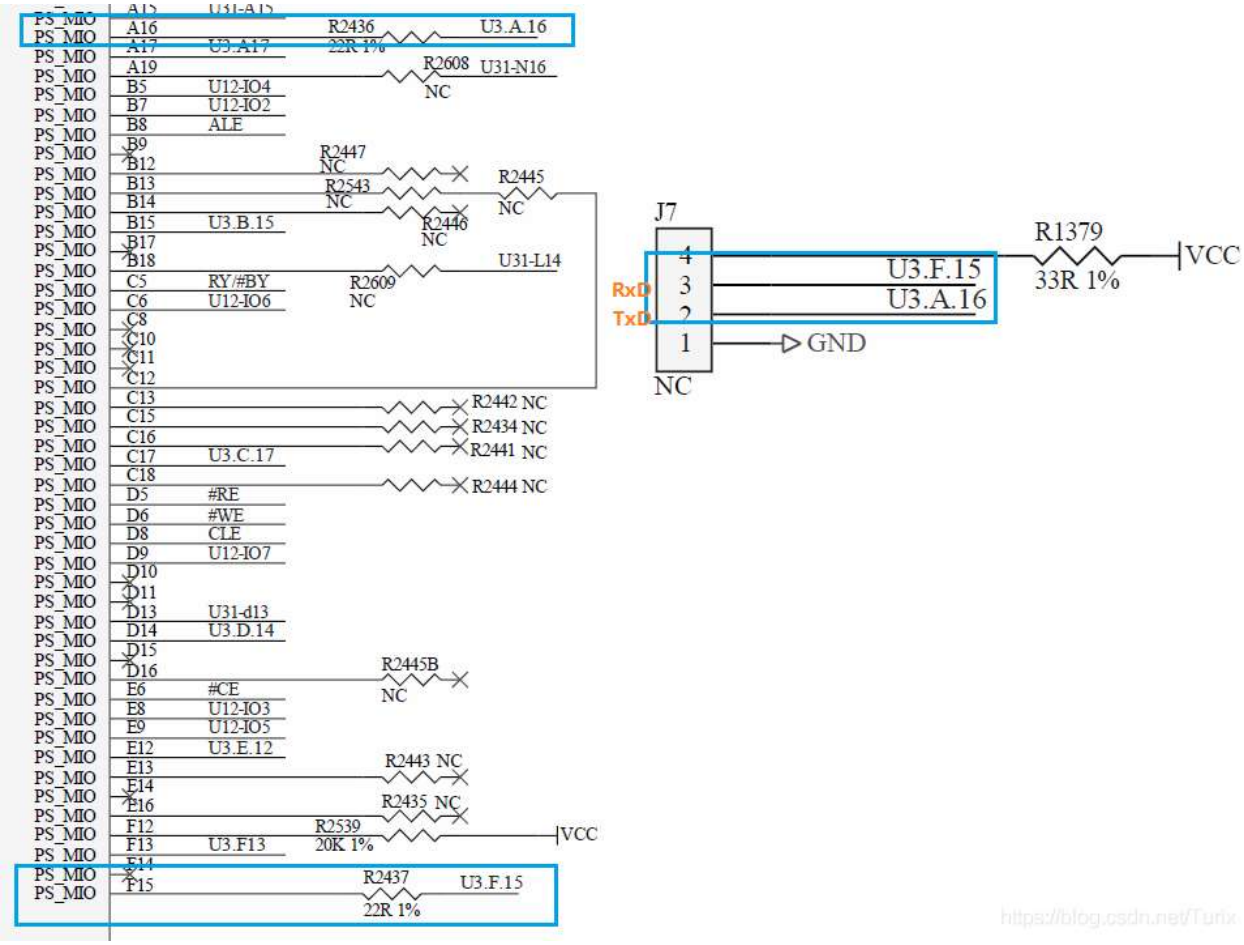
11. We need to disable AXI: To do that, we can go to PS-PL configuration on the left panel, then to AXI Non secure enablement and then to GP Master AXI interface. Then disable “M AXI GP0 interface”.



12. Also, unselect the "FCLK_RESET0_N" item in "General" -> "Enable Clock Resets";

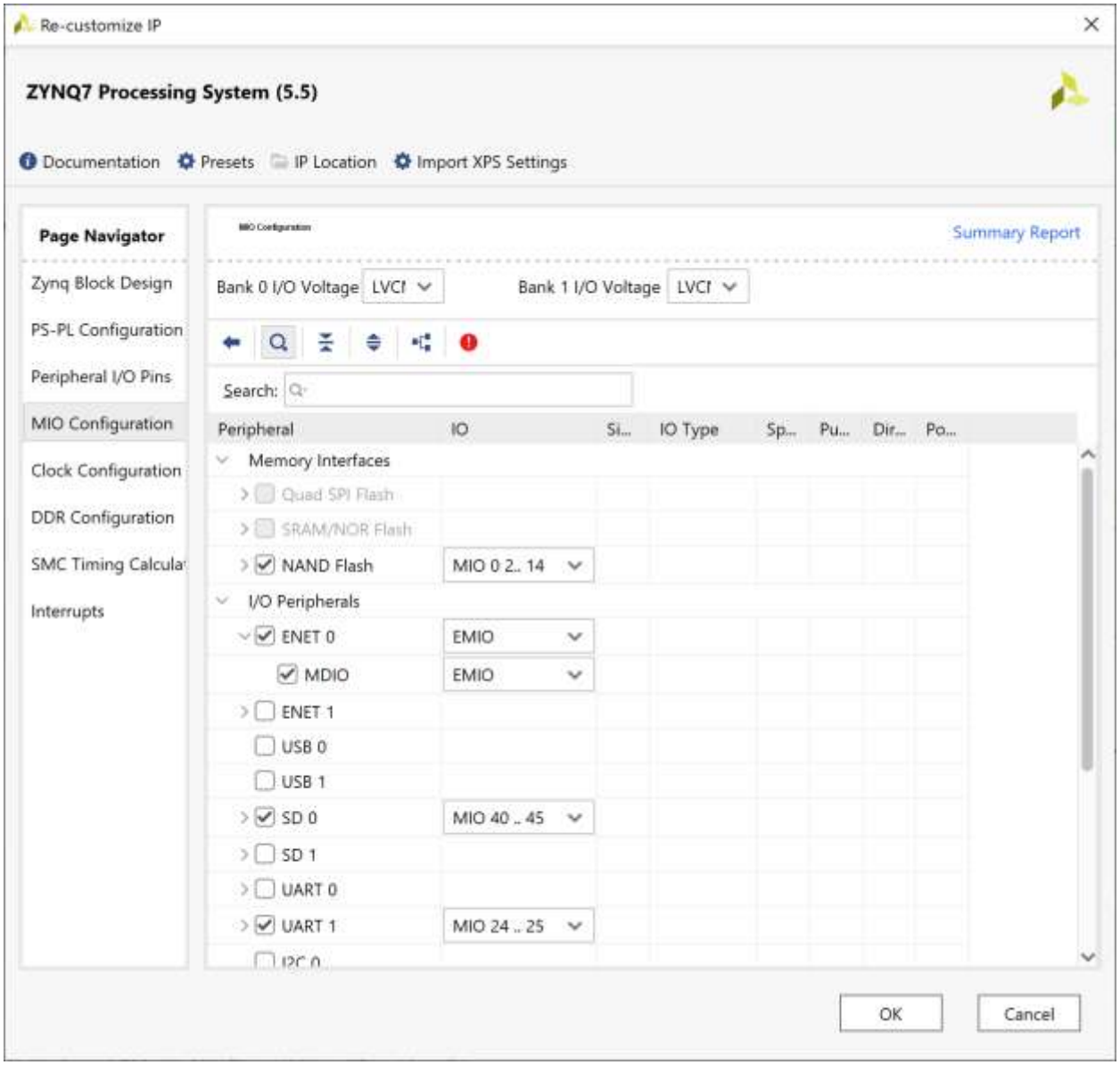


13. Let’s check where the UART port is connected, so let’s see the schematics

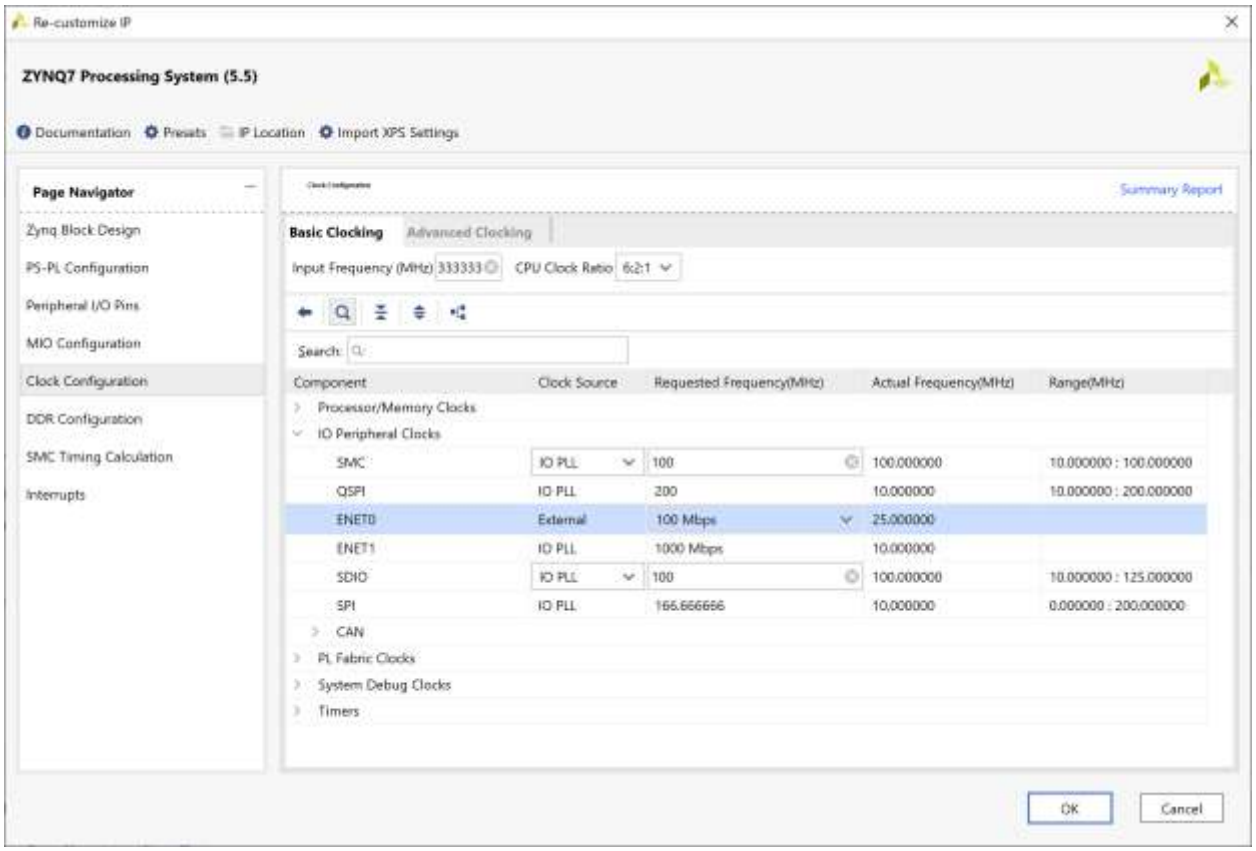


This is the pin circuit diagram of the PS (ARM) side, so we found it at MIO24 and MIO25, which is the multiplexing pin of serial port 1.

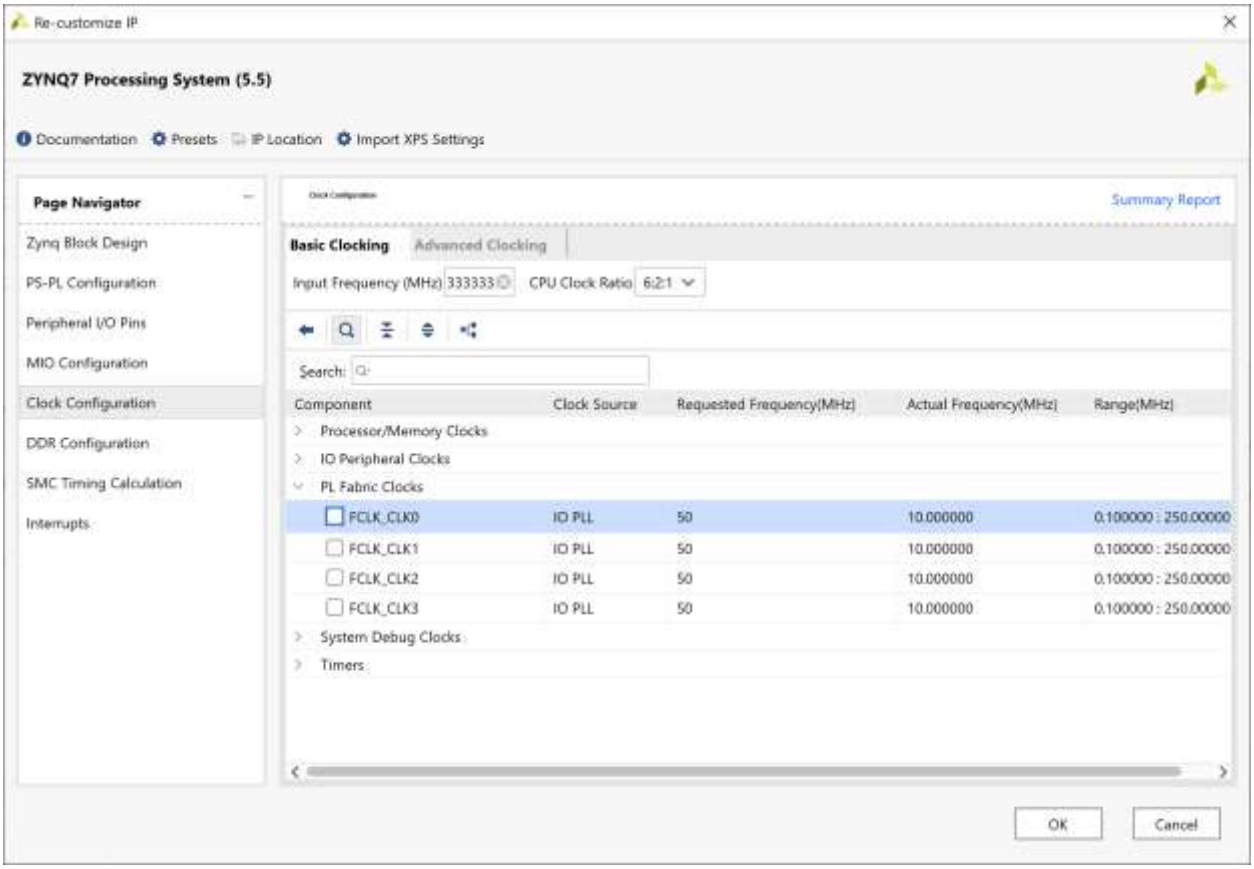
14. Once we have this, we have to go to “MIO configuration” on the left panel. Check “NAND Flash” under Memory interfaces section. Check “ENET 0” (remember to enable MDIO under this tab), “SD 0” and “UART 1”. Check all of them has the same pin configuration as in the figure.



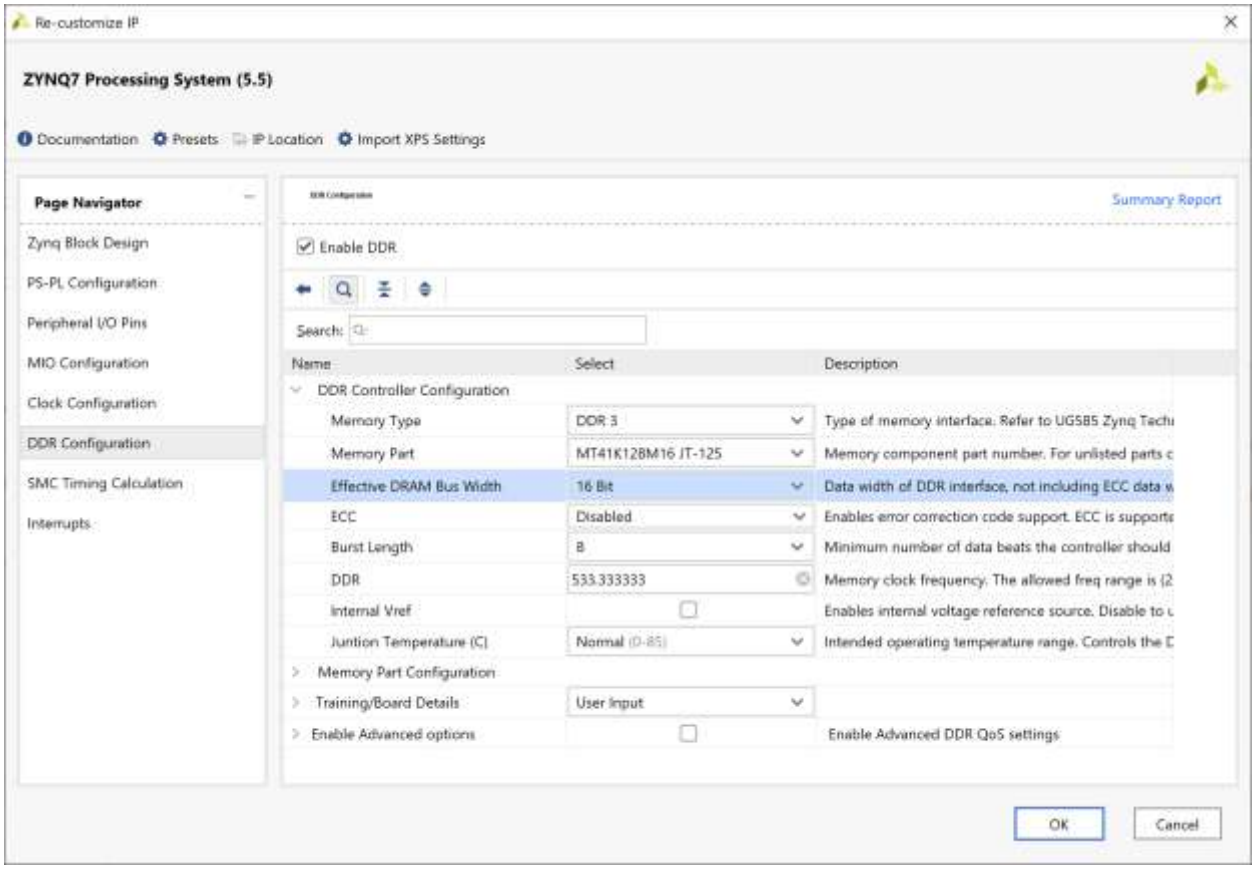
15. Clock settings. Set the ethernet clock. Change the network port to 100M:



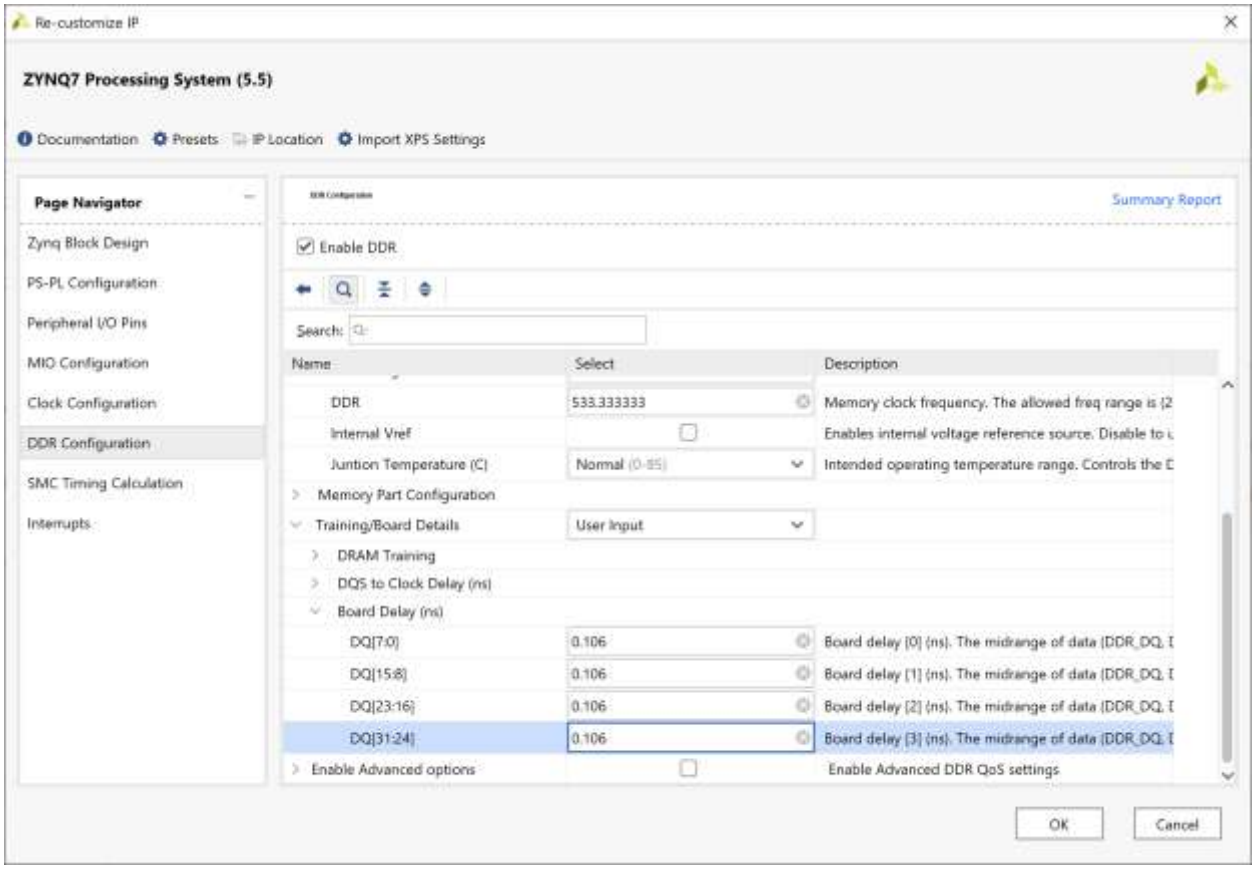
16. Under “PL Fabric Clocks” section, disable also FCLK_CLK0.



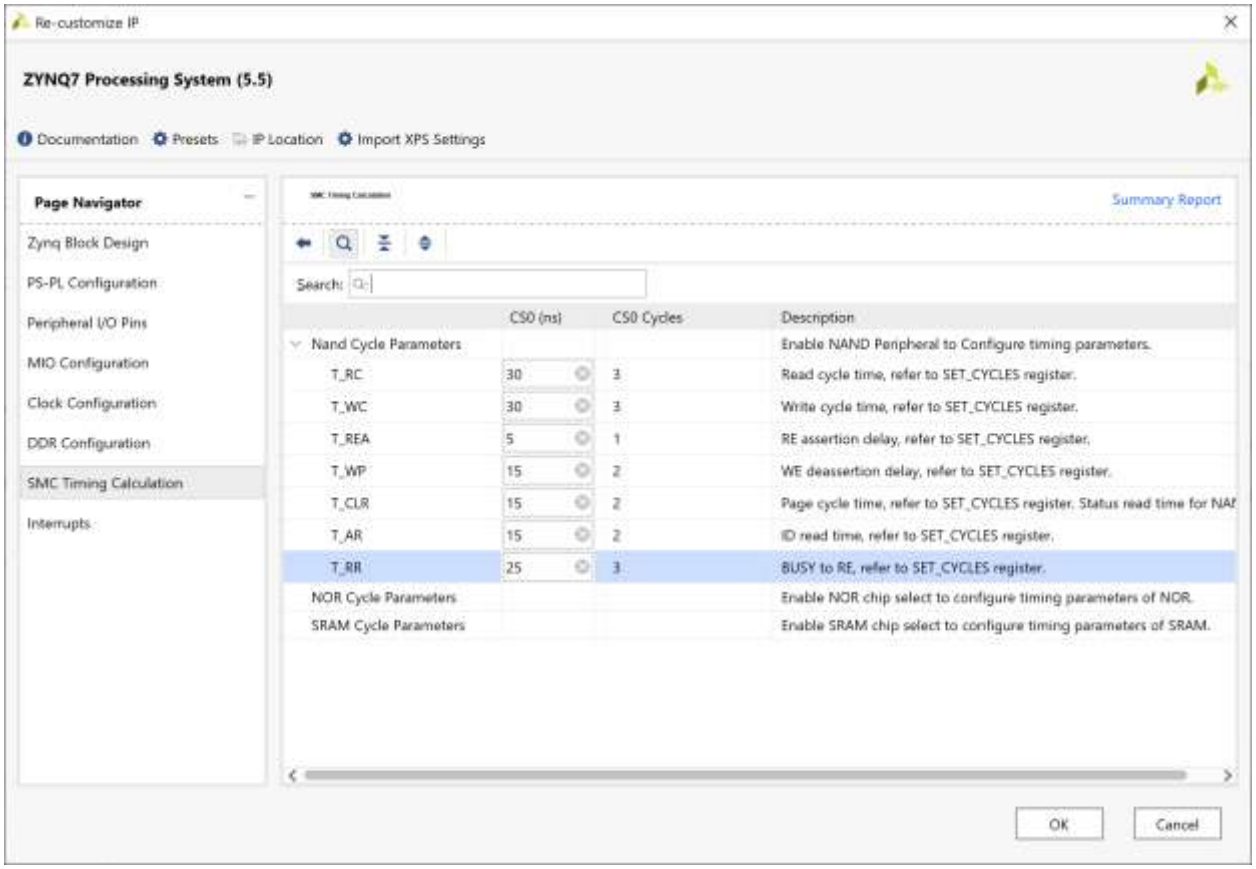
17. Now we are moving to DDR configuration on the left panel. We need to select the model and the bus width.
    The model compatible either you have Micron or Etron memory is MT41K128M16 JT-125.
    Then under “Effective DRAM bus width” we have to select 16 bits.



18. Now we need to click under “Training/Board details” and then into “Board delay” section. Once there all the four values have to be set with 0.106 (by default Vivado sets 0.25).

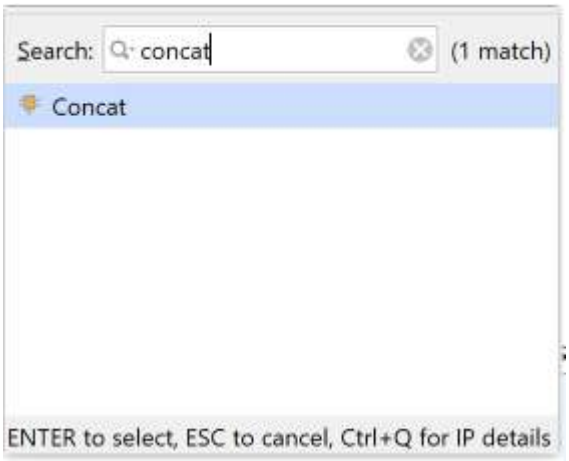


19. Now we are going to “SMC Timing calculation”, then on “nand cycle parameters” and modify the values to match the figure. (Only change the field CS0. The field CS0 cycles will automatically get updated by Vivado).

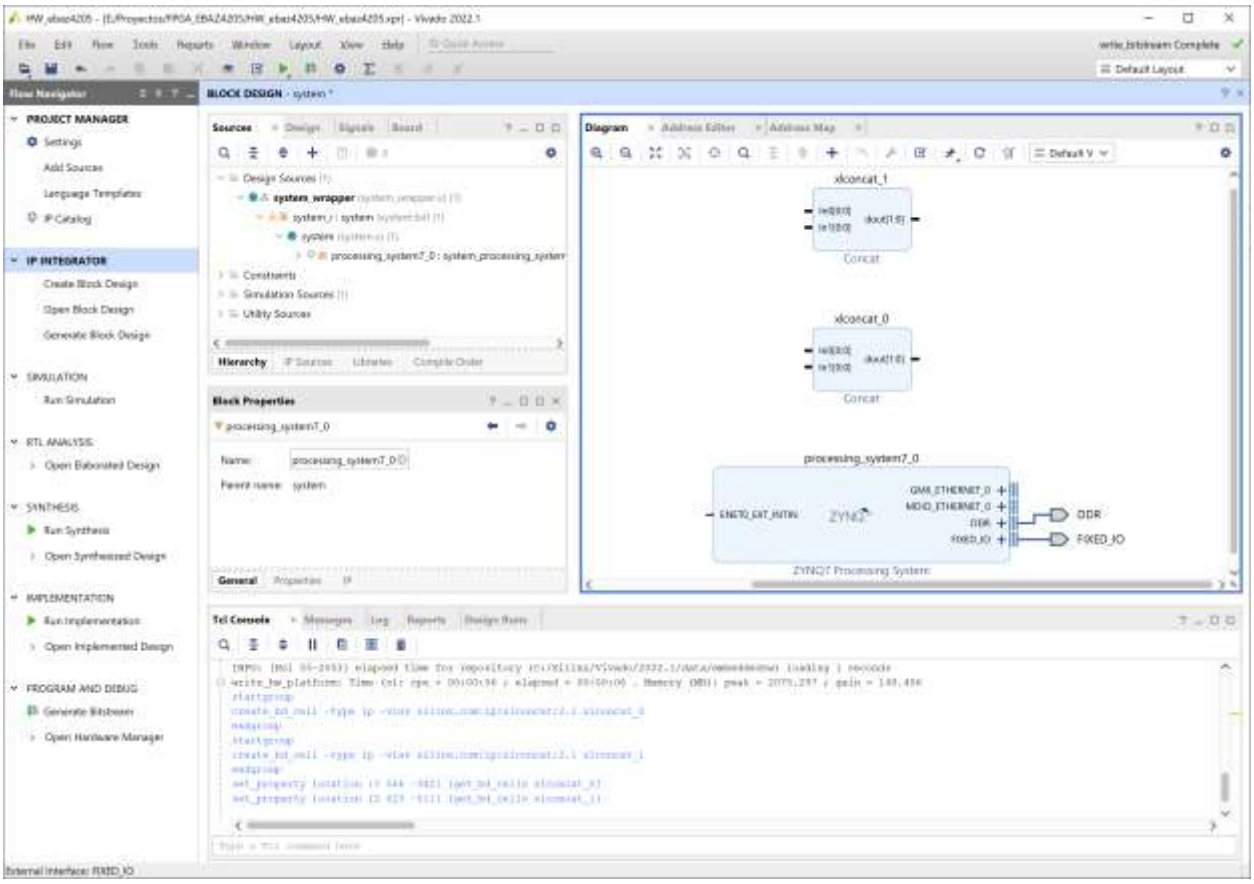


20. As the network port of the board is led out through EMIO, we need to manually assign the pins one by one. But this IP is a GMII Gigabit Ethernet port, TX RX has 8 bits, and the MII interface TX RX used by the 100M network card has only 4 bits. You must explicitly add two “concat” modules to convert 8 bits to 4 bits, otherwise it will be redundant. The pin is brought out but the IO port is not allocated, and an error will be reported when the bitstream is finally generated.

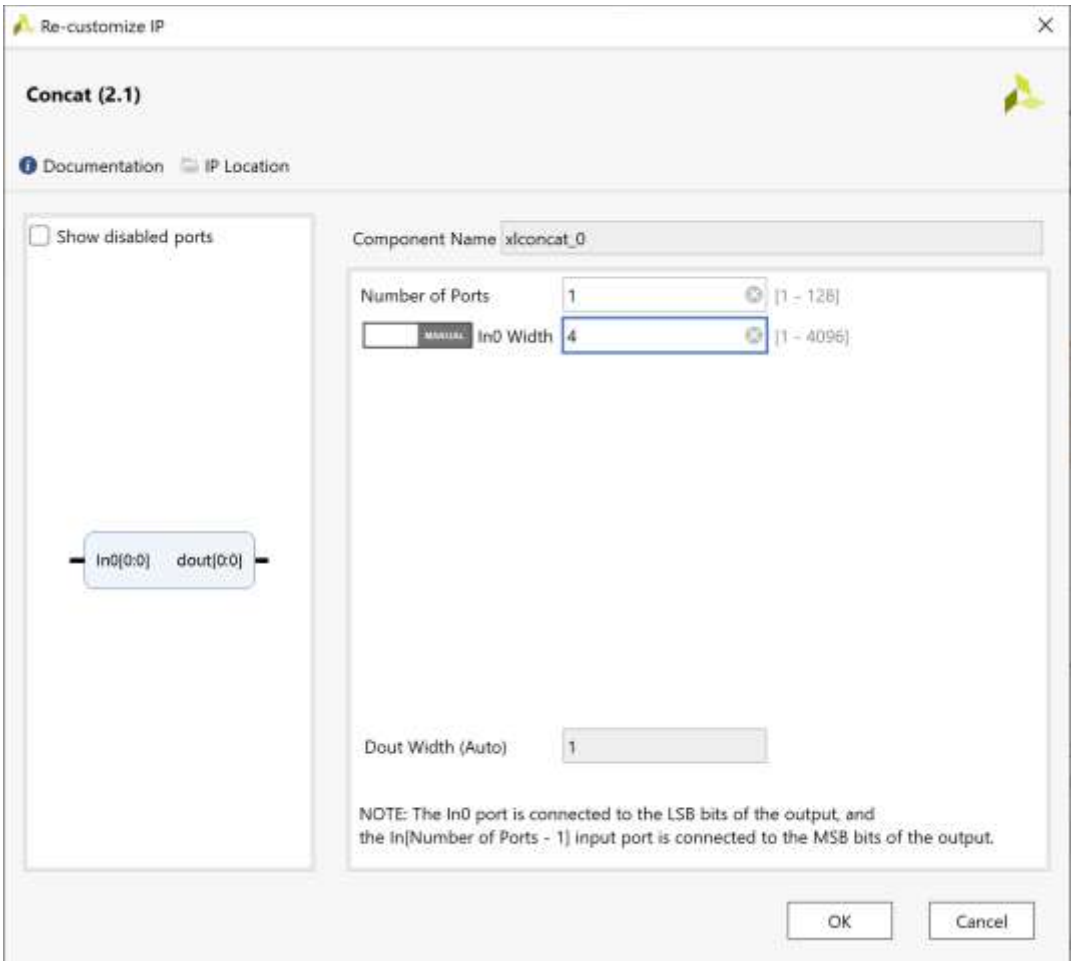
    Follow the steps 5 and 6 in order to add a new IP. On the search bar type “concat”.



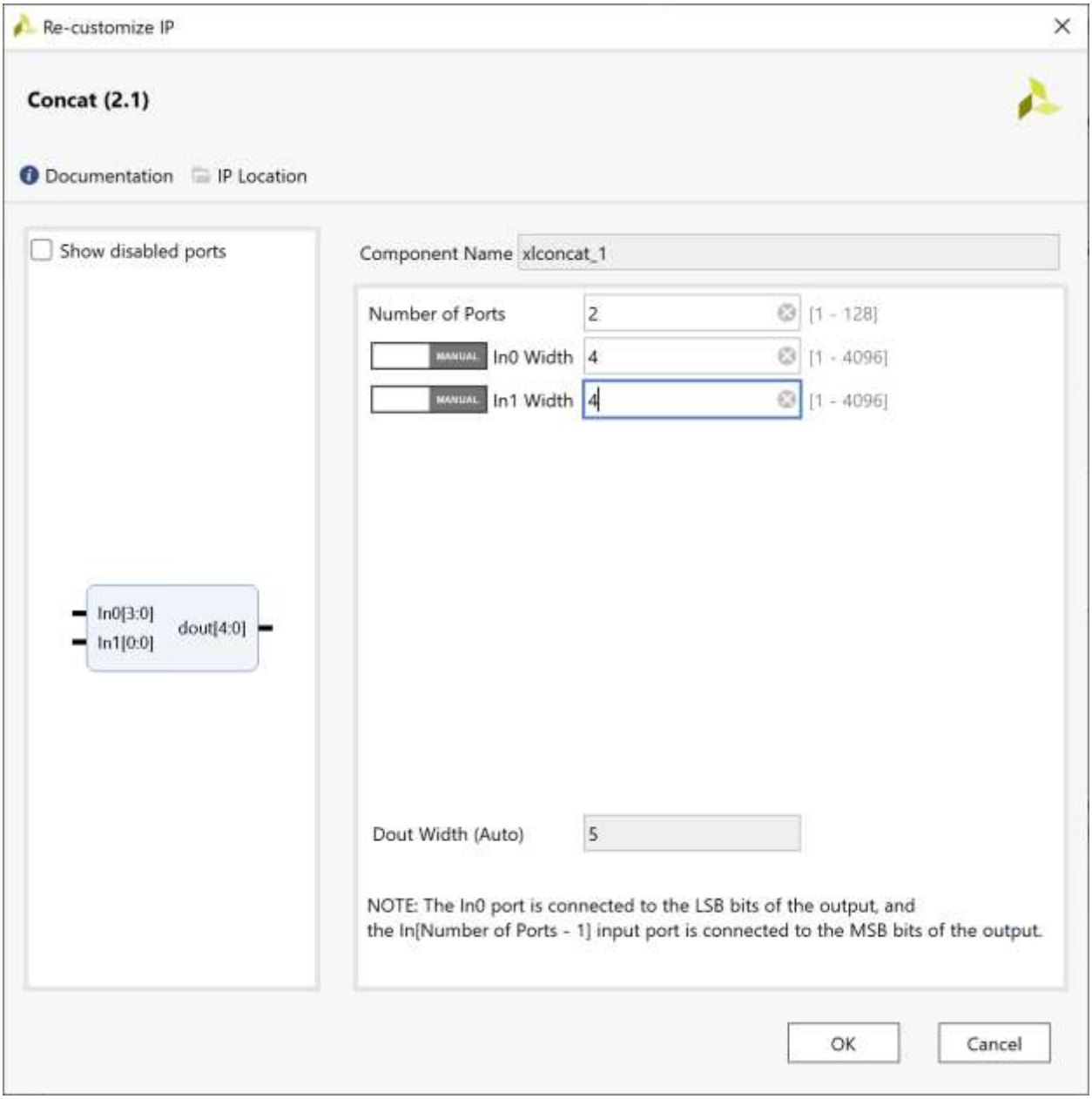
    Automatically, Vivado adds “xlconcat_0” and “xlconcat_1”.



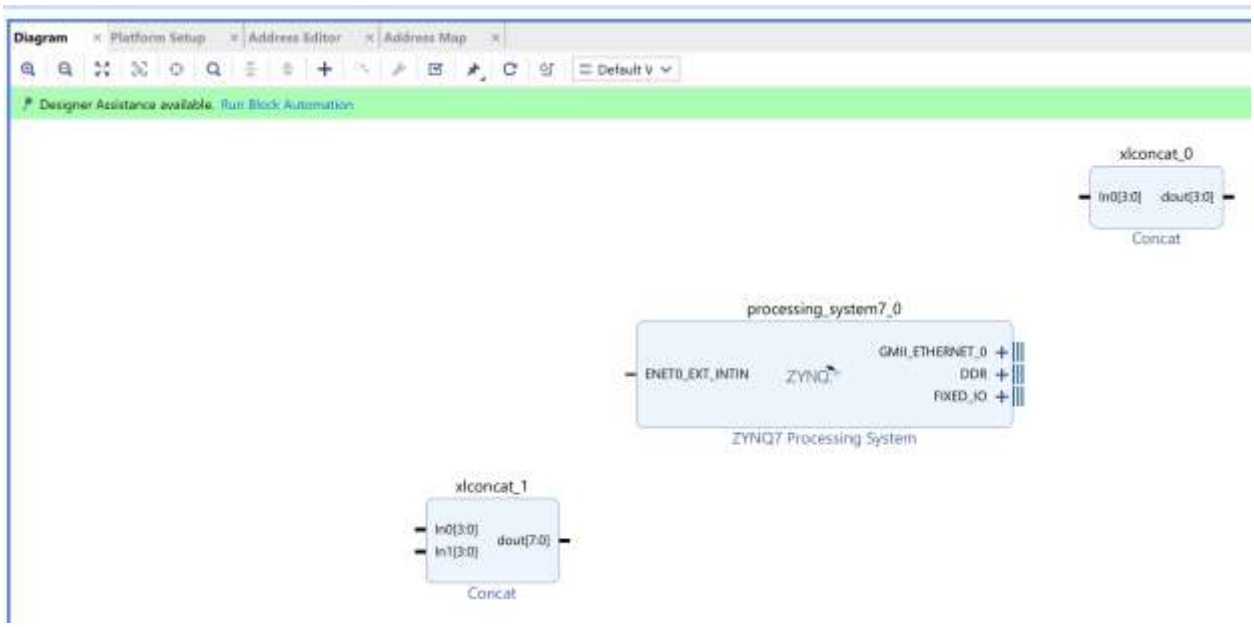
21. Double click to edit number of bits and IOs. The RX is 2-inputs, 4 bits each, the TX is 1 input, 4 bits. TX will be xlconcat_0 and RX will be xlconcat_1. Remember to set “auto” to “manual” on the slider. Then click OK.



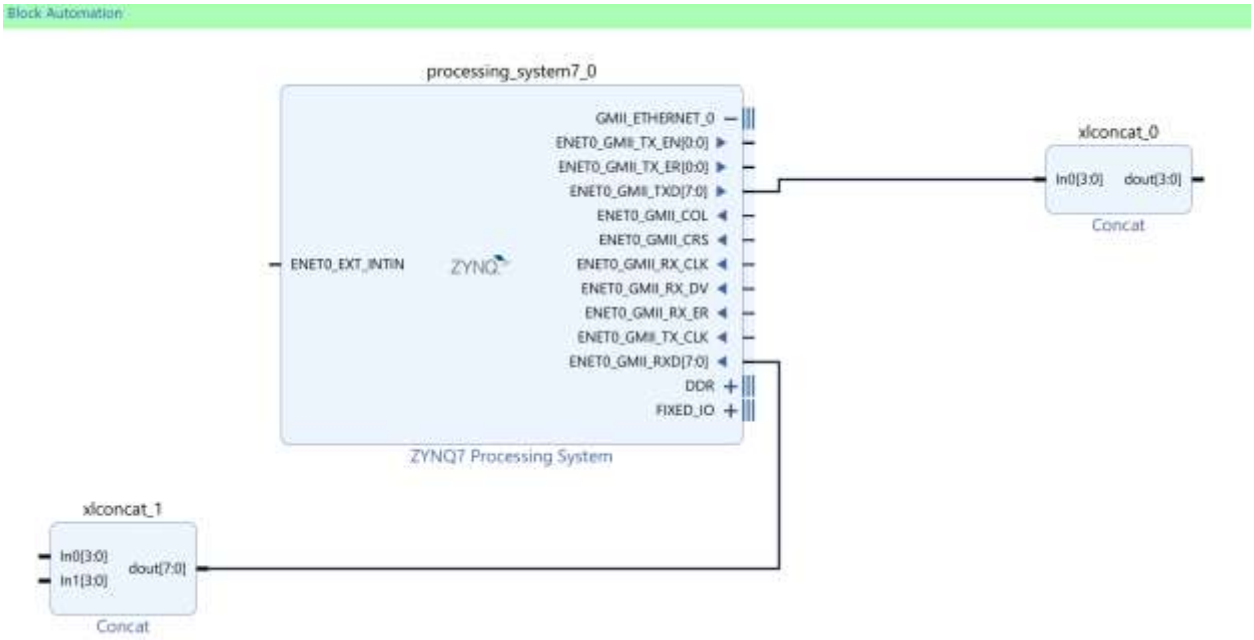
22. Now, with the RX and press OK



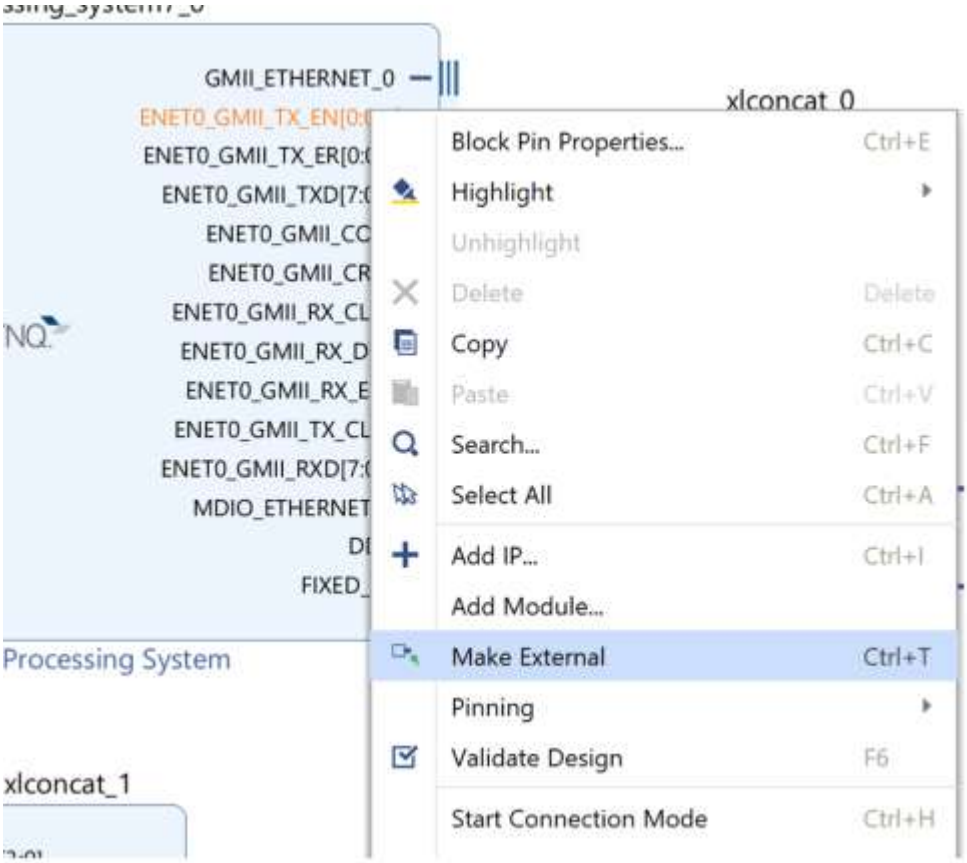
23. This is how it looks after these changes.



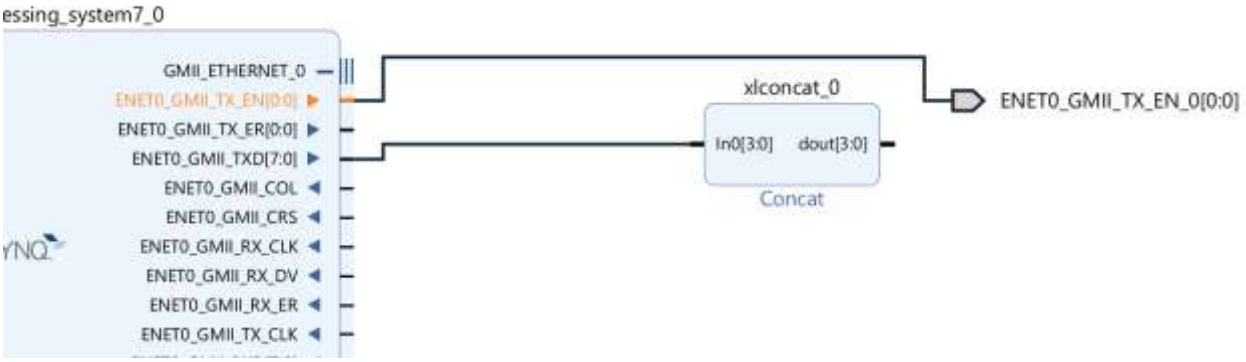
24. Now we need to create some connections. On the ZynQ7 block, click “+” next to GMII_ETHERNET0 to expand, connect block to *concat* blocks using a click and drag line. The connections will be the same as follows:



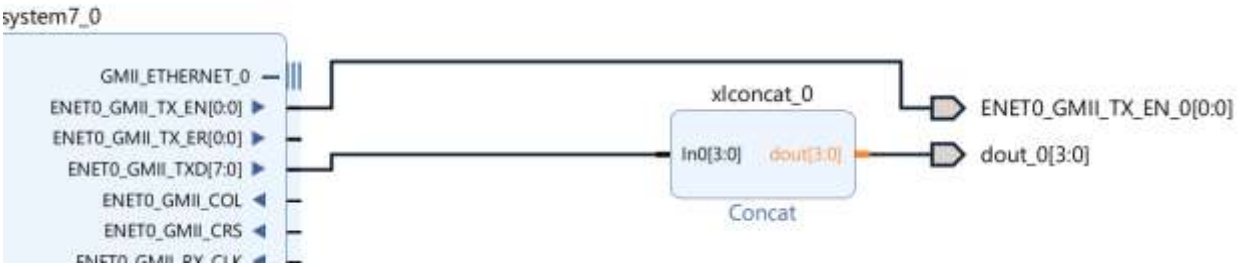
25. Do right click on ENET0_GMII_TX_EN[0:0] and select Make external to create a port.



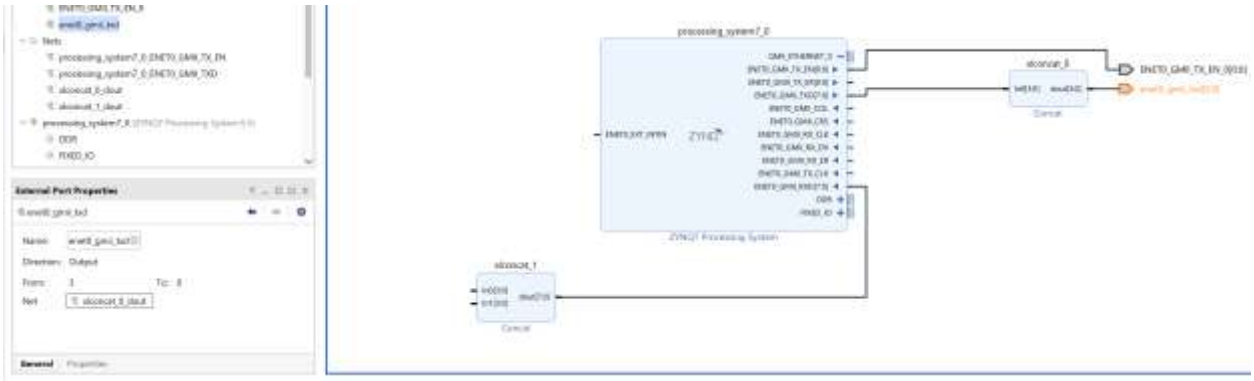
So, the result is the following:



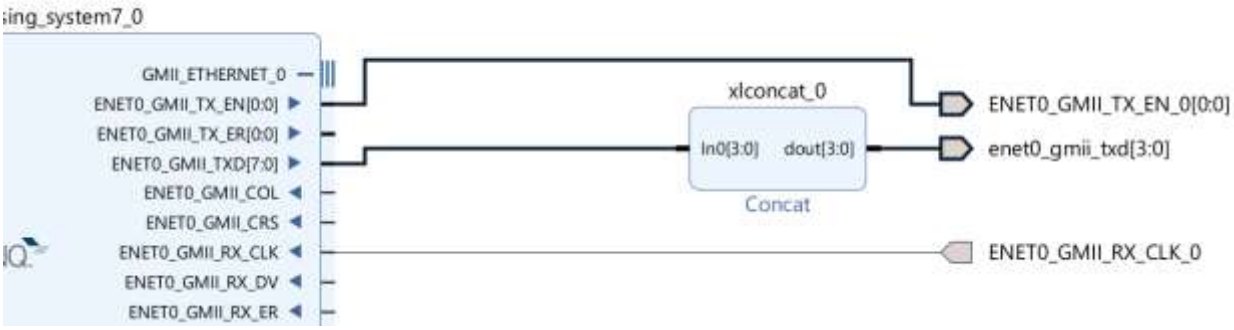
26. Now repeat the last step with dout[3:0] of xlconcat_0 block.



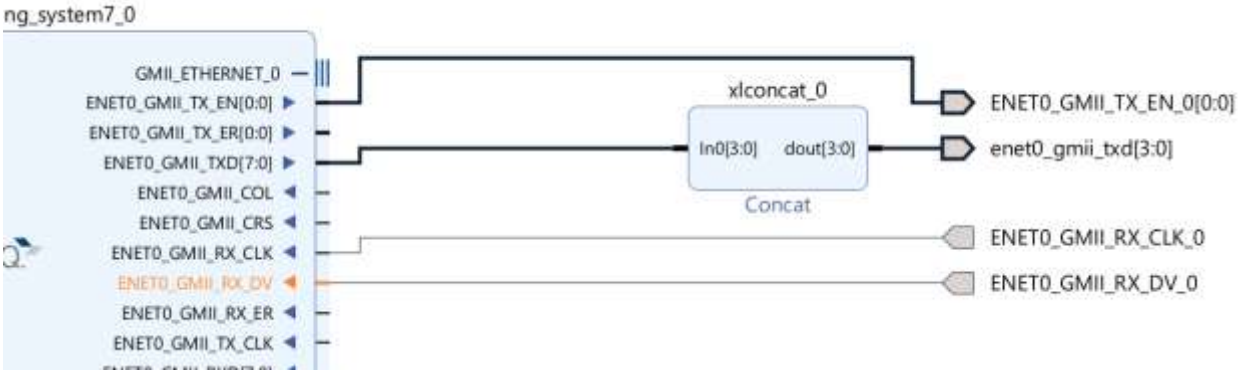
27. Do click on recently created “dout_0[3:0]” in order to change the name to “enet0_gmii_txd”. The name can be changed under the “External port properties” box.



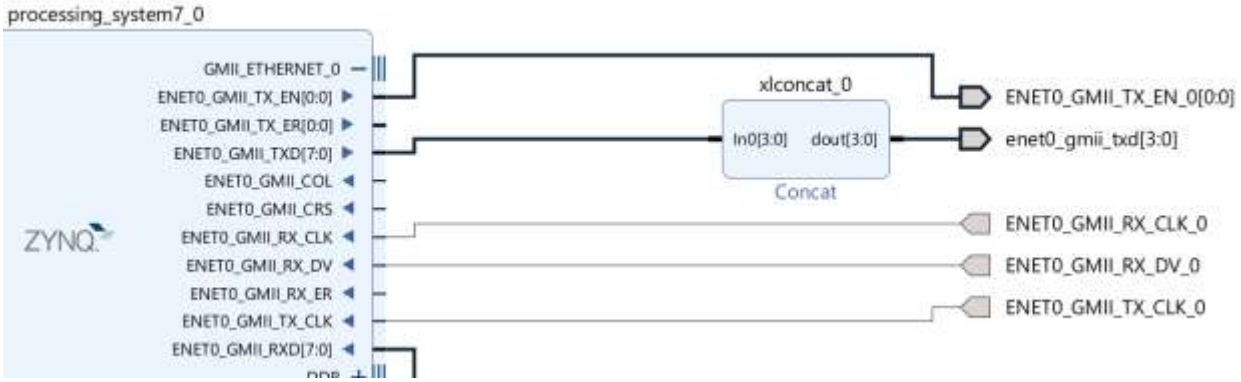
28. Now repeat step 25 with ENET0_GMII_RX_CLK



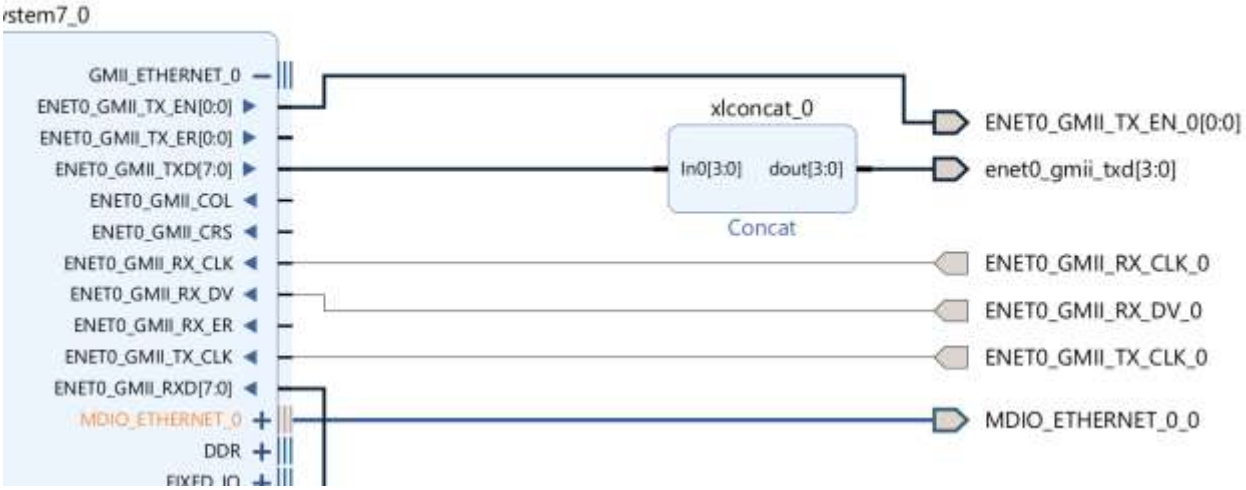
29. Repeat step 25 with ENET0_GMII_RX_DV



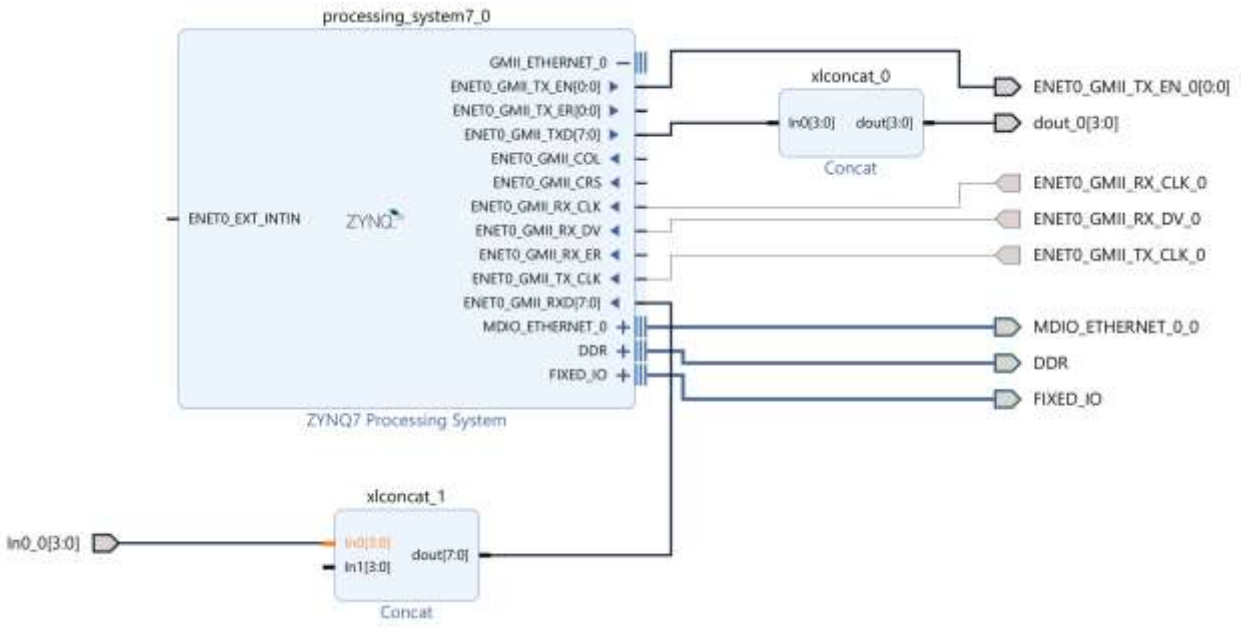
30. Repeat step 25 with ENET0_GMII_TX_CLK



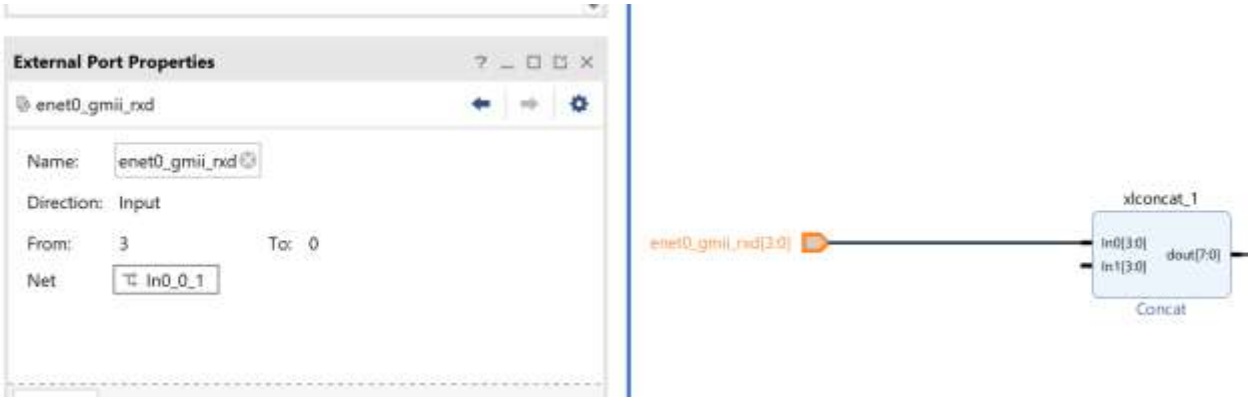
31. Repeat step 25 with MDIO_ETHERNET_0



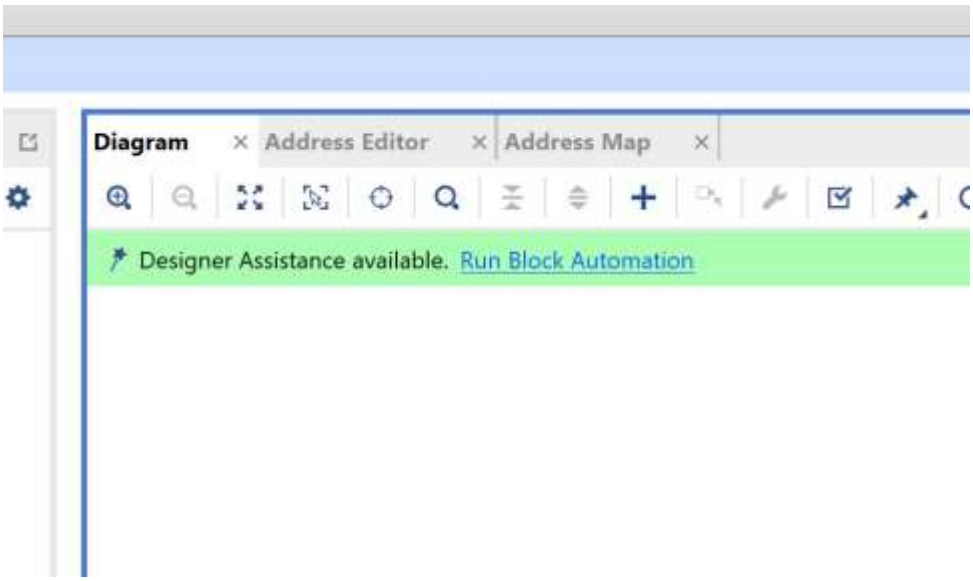
32. Repeat step 25 with in0[3:0] of xlconcat_1 block.



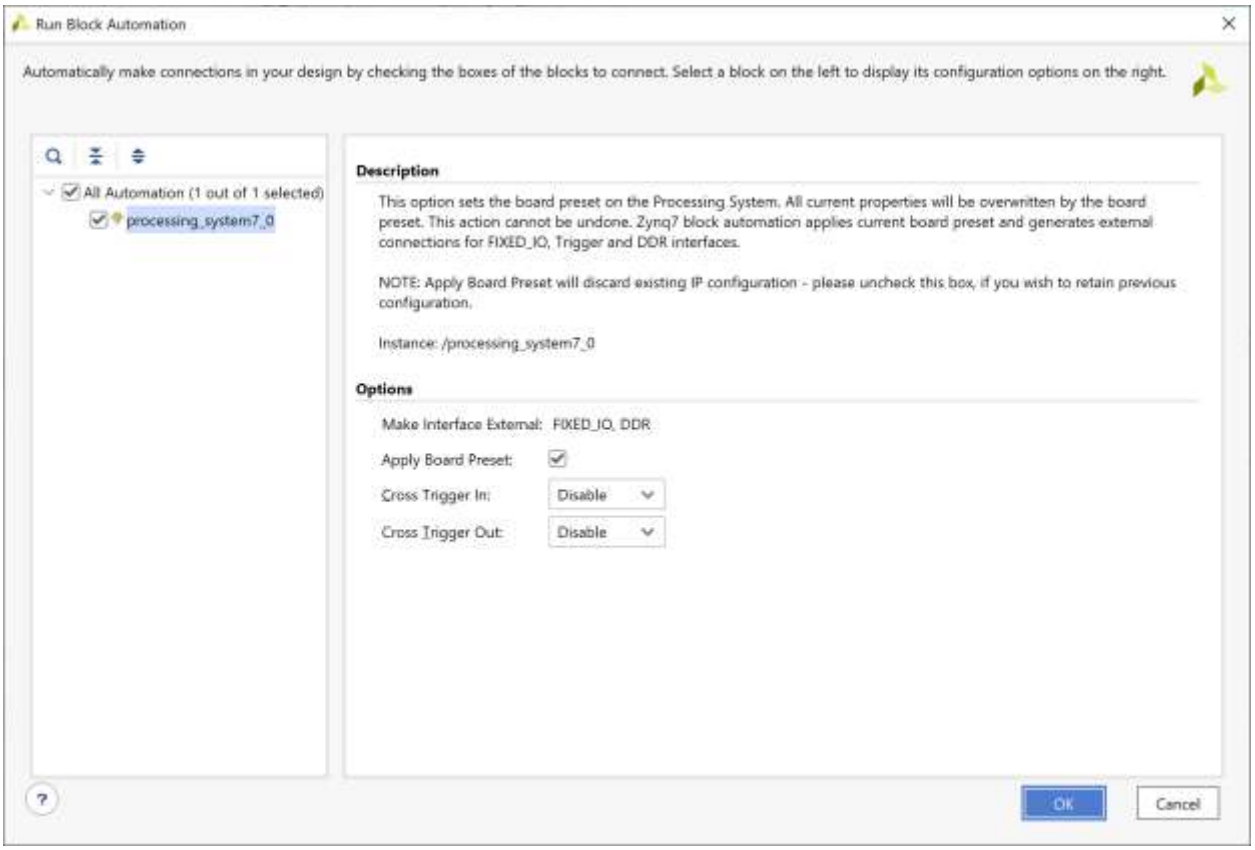
33. Do click on recently created “In0_0[3:0]” in order to change the name to “enet0_gmii_rxd”. The name can be changed under the “External port properties” box.



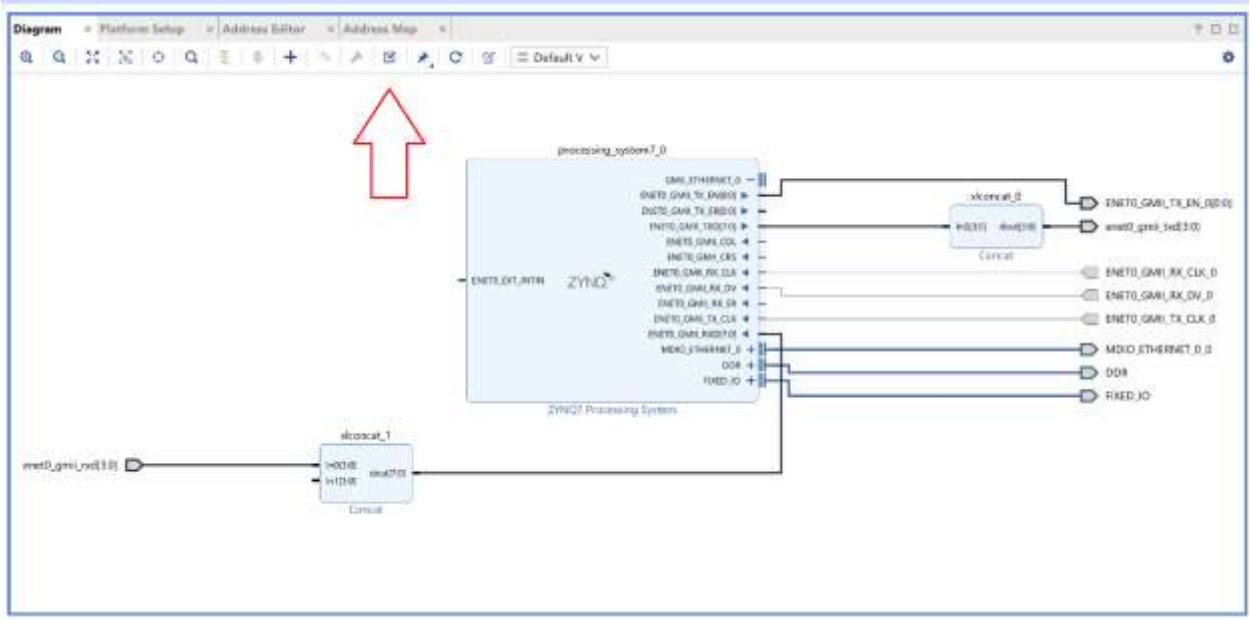
34. We should see this and run it



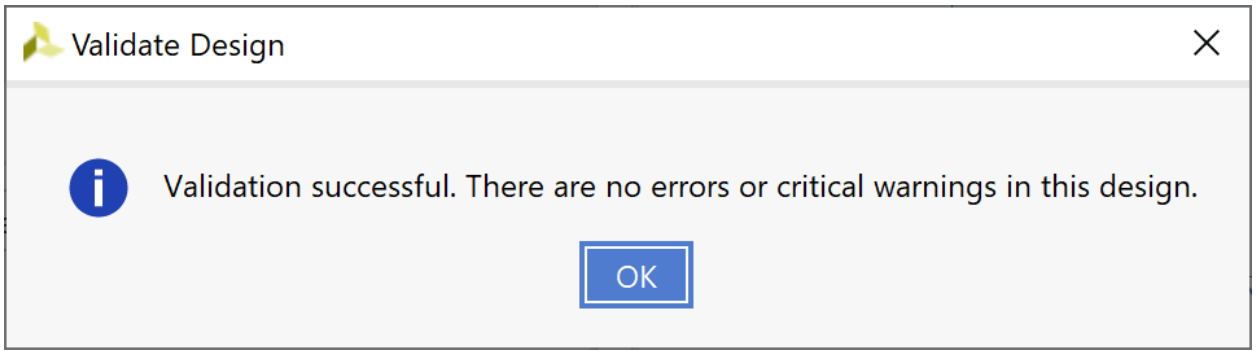
Then you will see this window, click OK



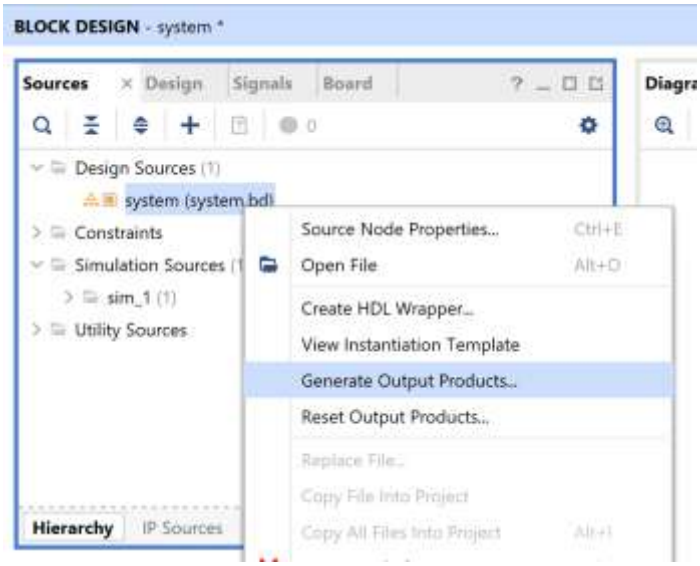
35. Now we need verify the design. To do that, click on this icon



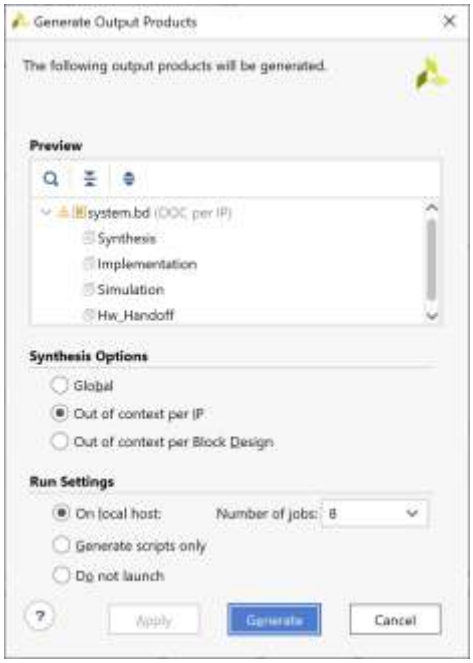
And then you should see this



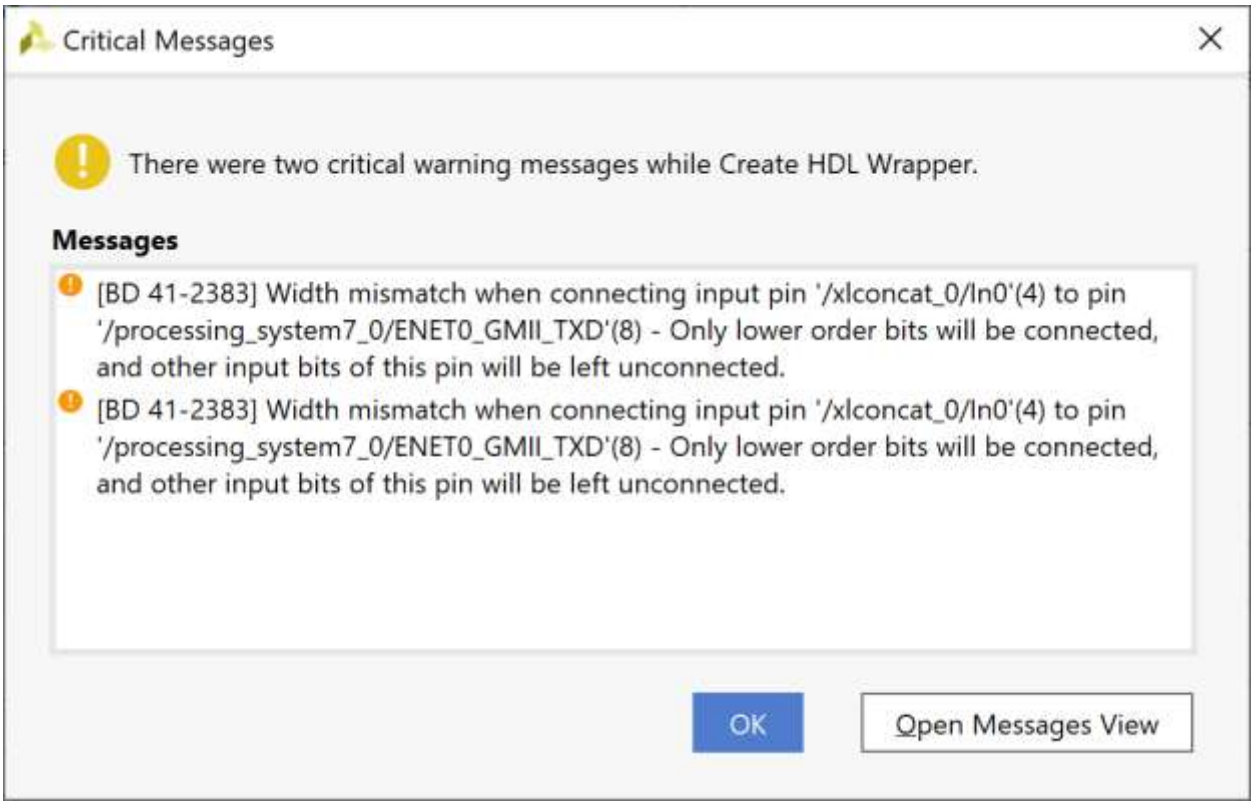
36. Now we need to go to “Sources” tab and do right click on “system.bd” and select generate output products.



37. In this popup window, select number of jobs: 8 and then click on “Apply” and then in “Generate”.



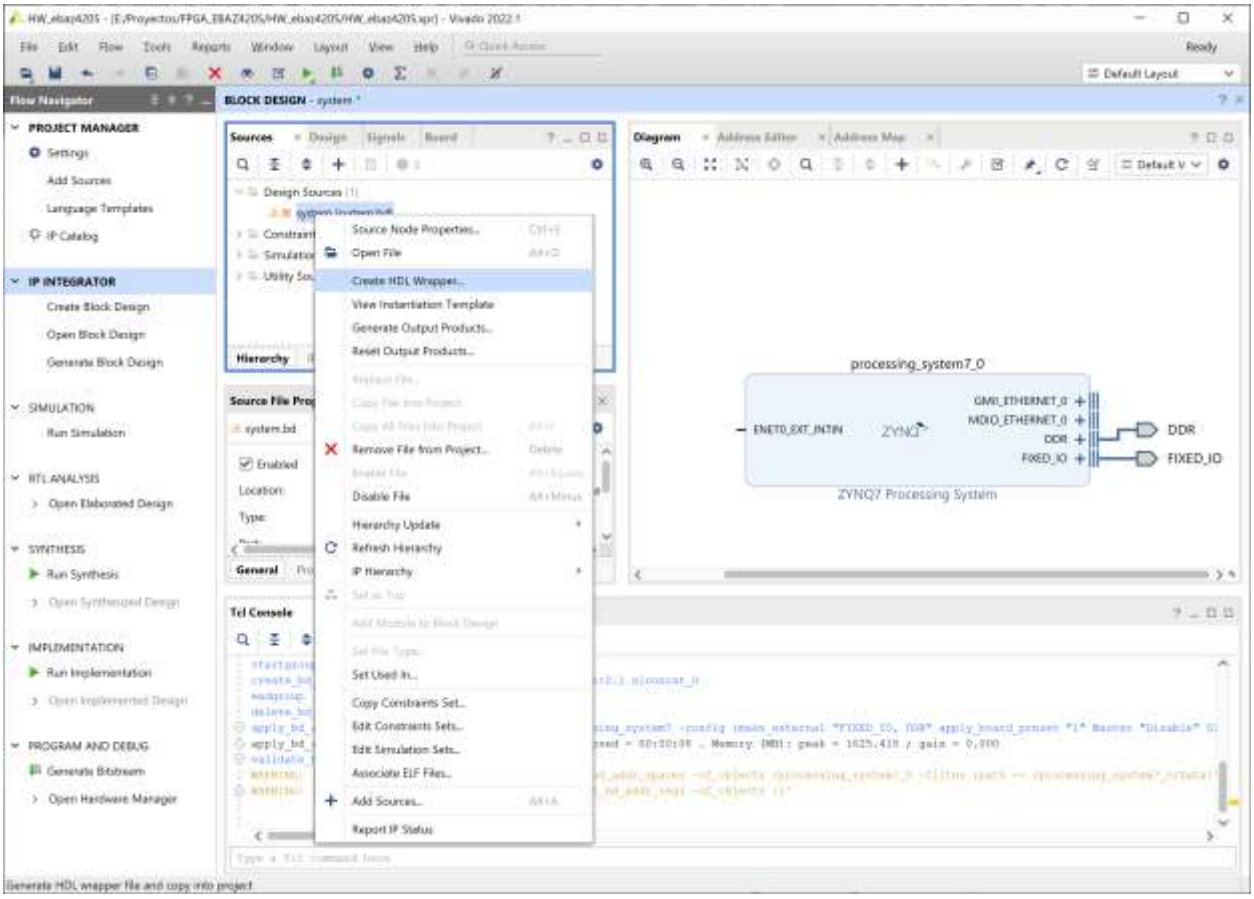
If you see a warning like this:



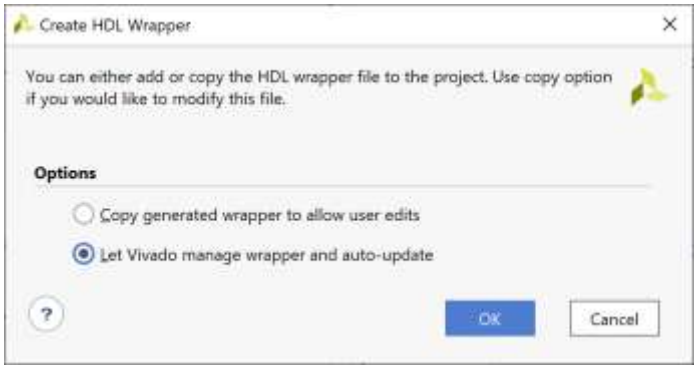
Essentially is telling us that the upper bits of ENET0_GMII_TXD are unconnected (that's what we want anyways). Click OK to authorize.

Once it finishes, you should see this

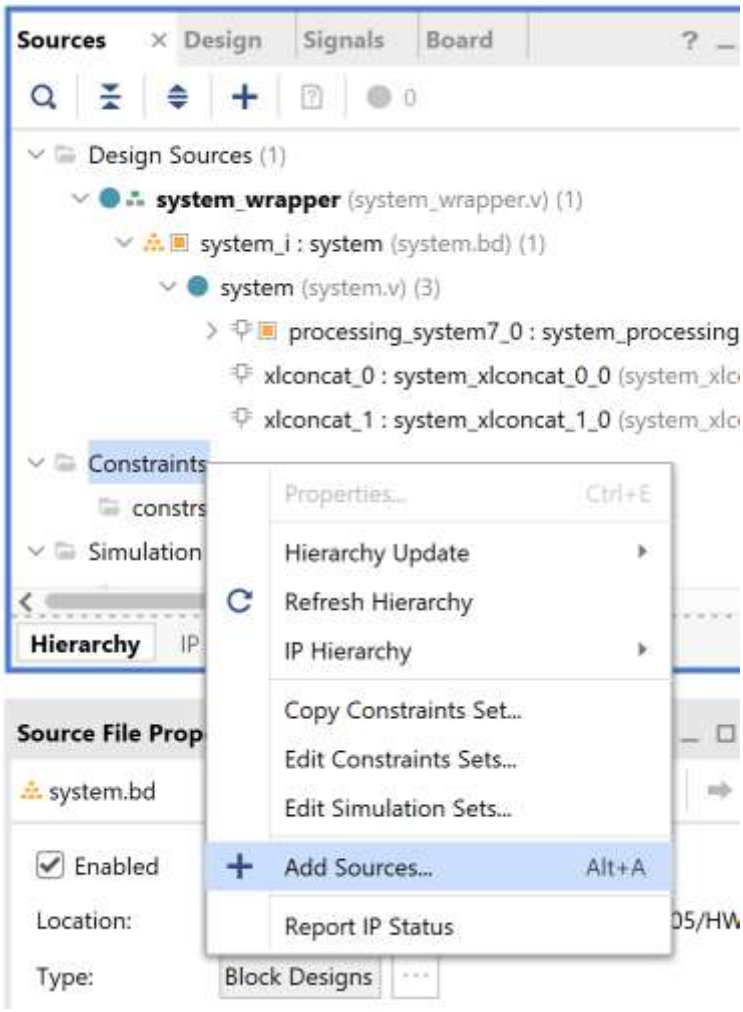
38. Now we need to go to "Sources" tab and do right click on "system.bd" and select "create HDL wrapper"



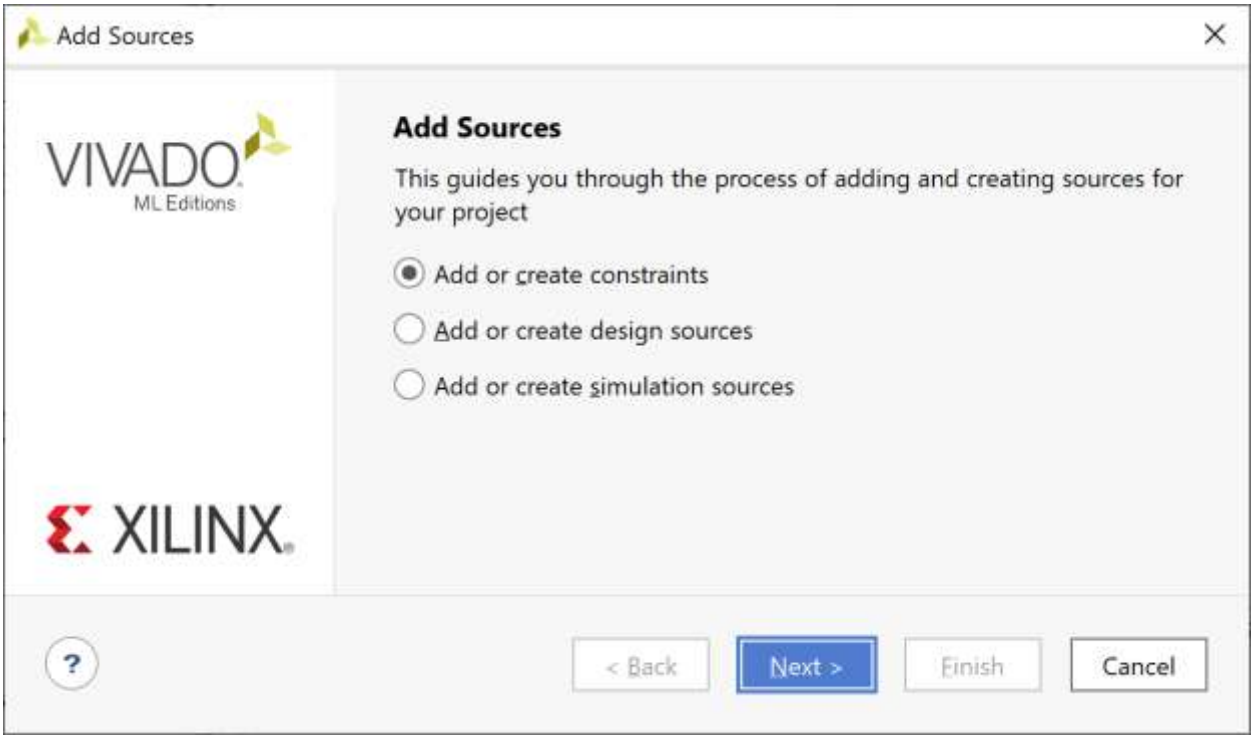
39. Let's Vivado do the work in the next window and click OK.



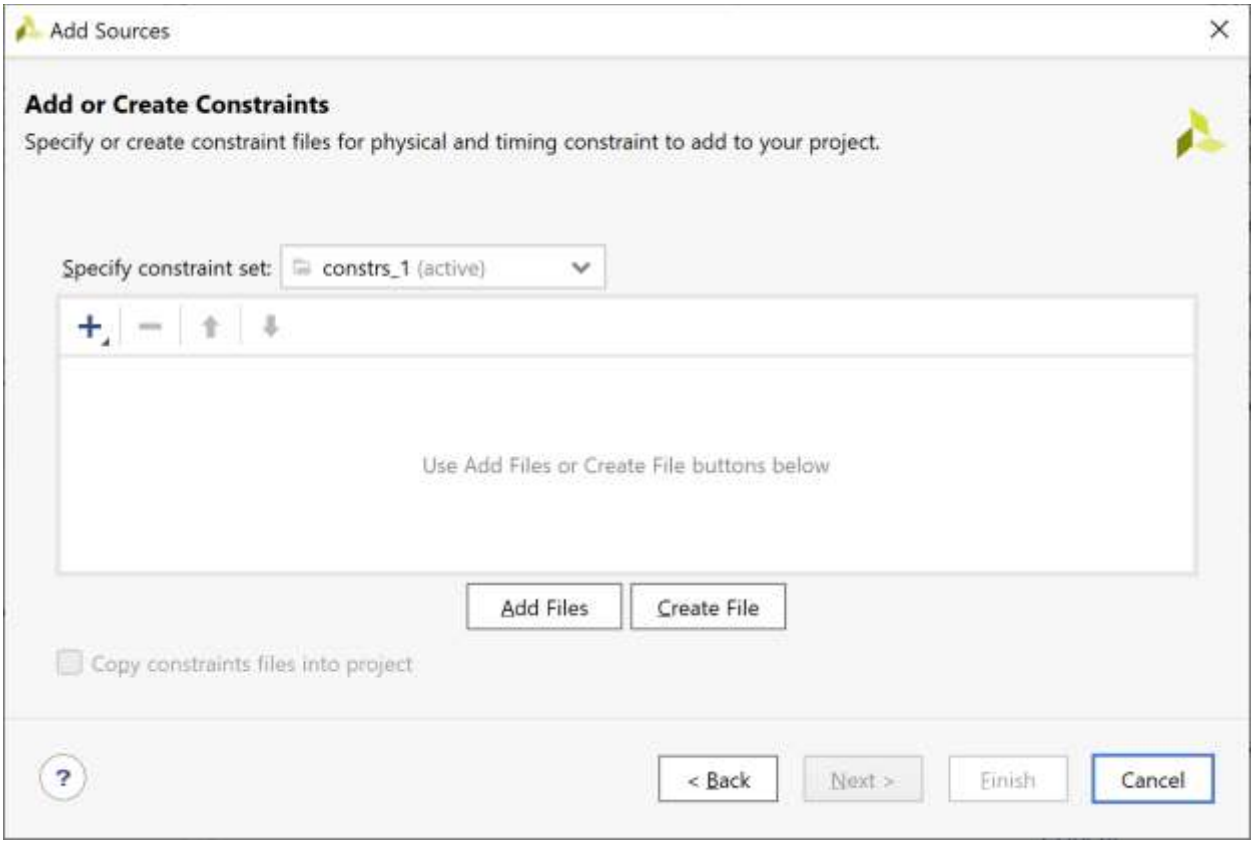
40. Do right click on "constraints" and then select "add sources…"



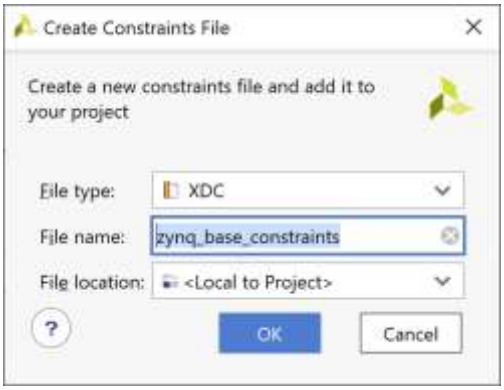
In this window, select Next.



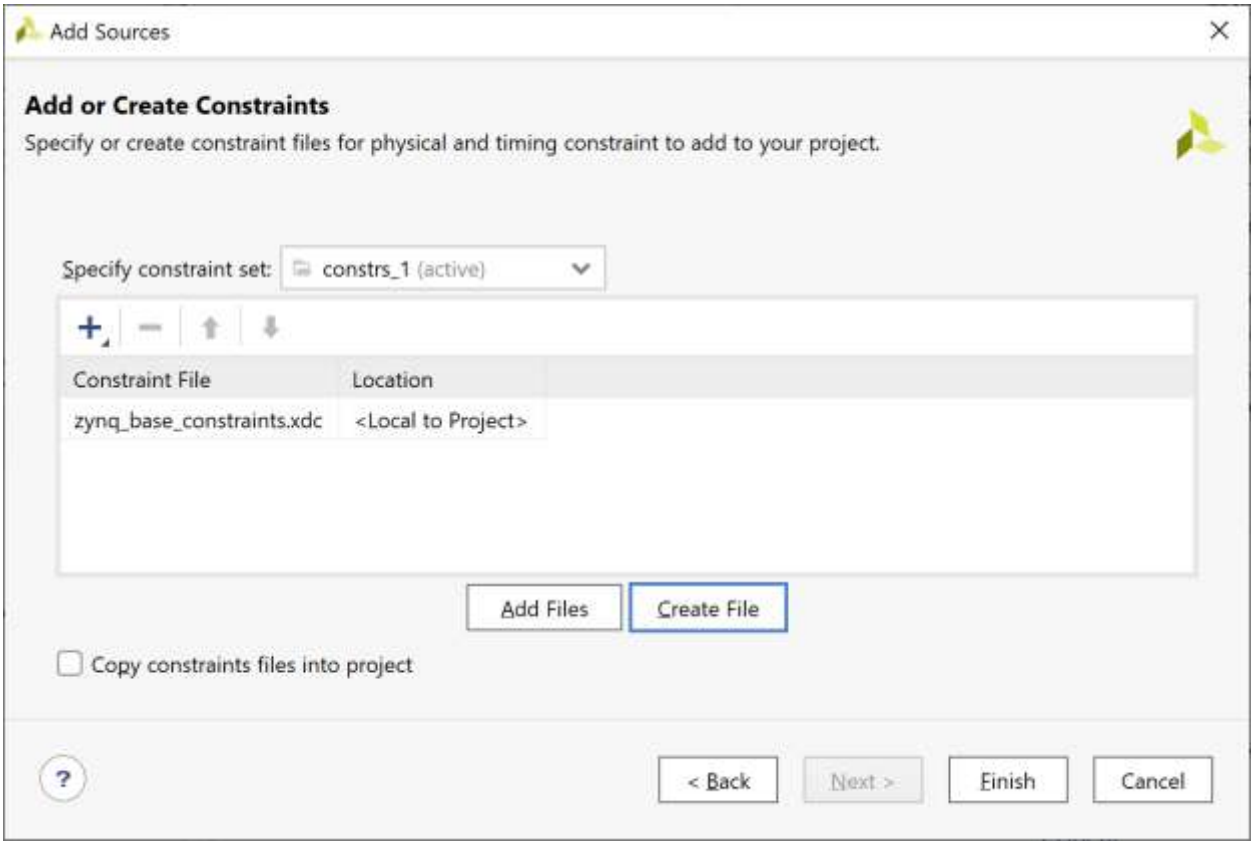
The click on “create file”



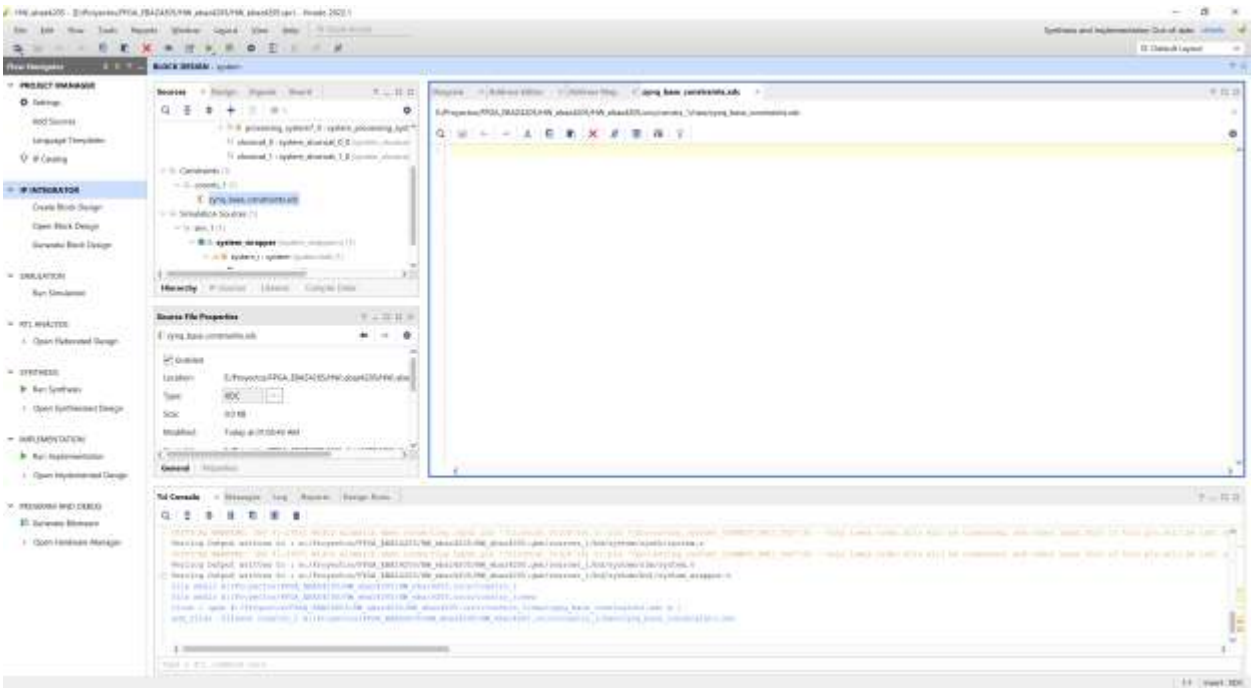
As file name we use “zynq_base_constraints” and press OK.



Then we will see the file listed, and click on finish.



41. Double click the constraints file recently generated

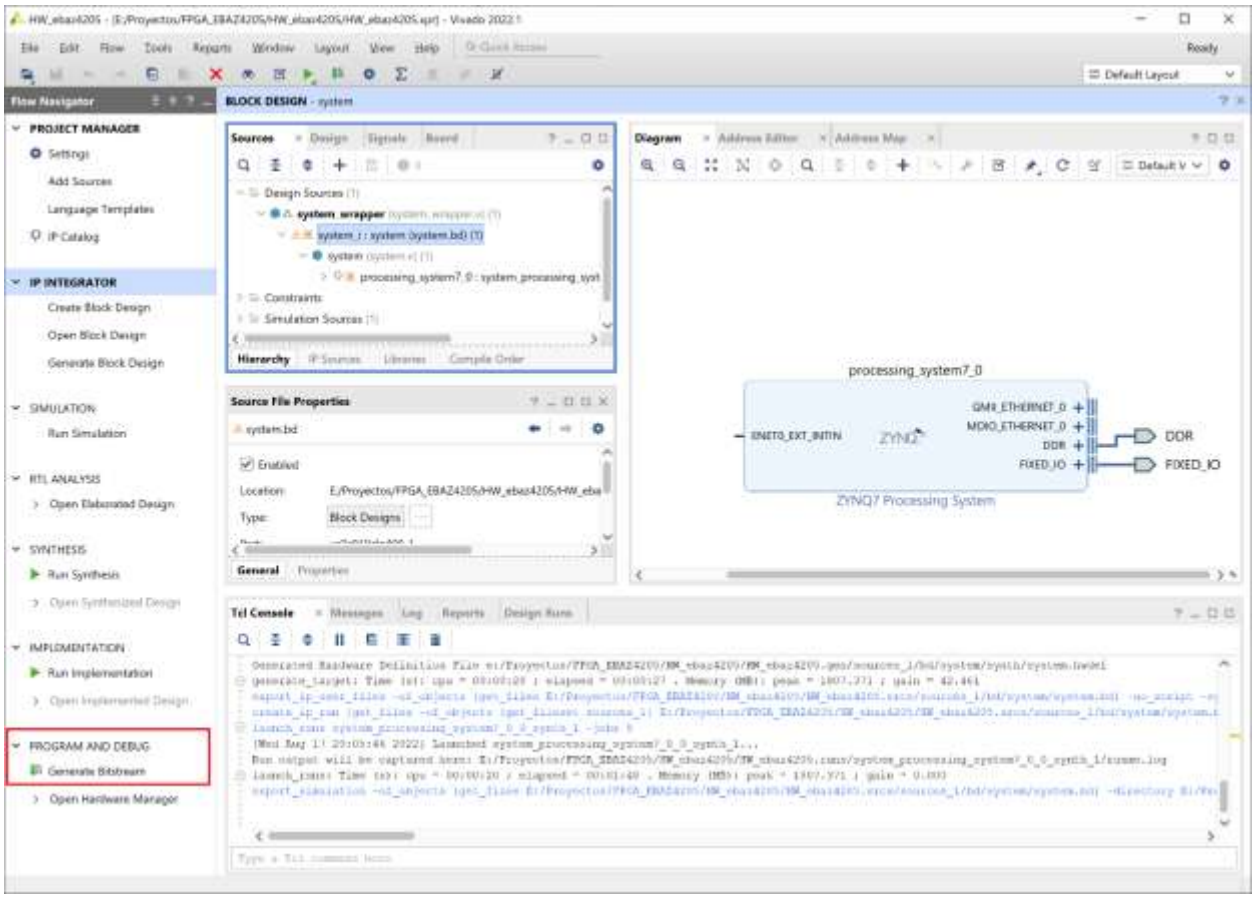


And paste this content and then press save:

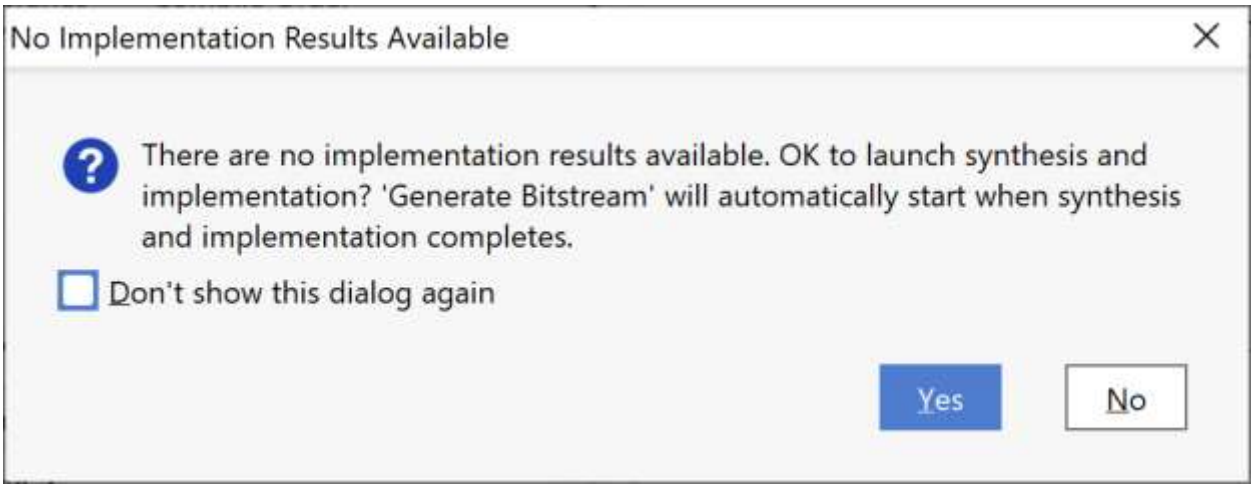
```
set_property IOSTANDARD LVCMOS33 [get_ports ENET0_GMII_RX_CLK_0]
set_property IOSTANDARD LVCMOS33 [get_ports ENET0_GMII_TX_CLK_0]
set_property IOSTANDARD LVCMOS33 [get_ports {enet0_gmii_rxd[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {enet0_gmii_rxd[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {enet0_gmii_rxd[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {enet0_gmii_rxd[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ENET0_GMII_TX_EN_0[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {enet0_gmii_txd[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {enet0_gmii_txd[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {enet0_gmii_txd[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {enet0_gmii_txd[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports ENET0_GMII_RX_DV_0]
set_property IOSTANDARD LVCMOS33 [get_ports MDIO_ETHERNET_0_0_mdc]
set_property IOSTANDARD LVCMOS33 [get_ports MDIO_ETHERNET_0_0_mdio_io]
set_property PACKAGE_PIN U14 [get_ports ENET0_GMII_RX_CLK_0]
set_property PACKAGE_PIN U15 [get_ports ENET0_GMII_TX_CLK_0]
set_property PACKAGE_PIN Y17 [get_ports {enet0_gmii_rxd[3]}]
set_property PACKAGE_PIN V17 [get_ports {enet0_gmii_rxd[2]}]
set_property PACKAGE_PIN V16 [get_ports {enet0_gmii_rxd[1]}]
set_property PACKAGE_PIN Y16 [get_ports {enet0_gmii_rxd[0]}]
set_property PACKAGE_PIN W19 [get_ports {ENET0_GMII_TX_EN_0[0]}]
set_property PACKAGE_PIN Y19 [get_ports {enet0_gmii_txd[3]}]
set_property PACKAGE_PIN V18 [get_ports {enet0_gmii_txd[2]}]
```

set_property PACKAGE_PIN Y18 [get_ports {enet0_gmii_txd[1]}]
set_property PACKAGE_PIN W18 [get_ports {enet0_gmii_txd[0]}]
set_property PACKAGE_PIN W15 [get_ports MDIO_ETHERNET_0_0_mdc]
set_property PACKAGE_PIN Y14 [get_ports MDIO_ETHERNET_0_0_mdio_io]
set_property PACKAGE_PIN W16 [get_ports ENET0_GMII_RX_DV_0]

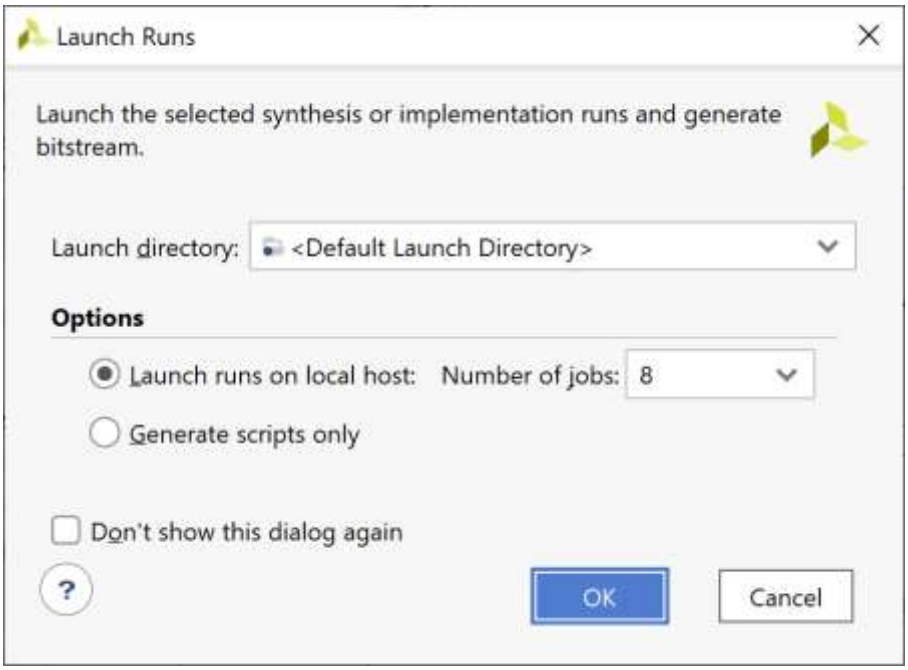
42. Finally, we need to generate bitstream.



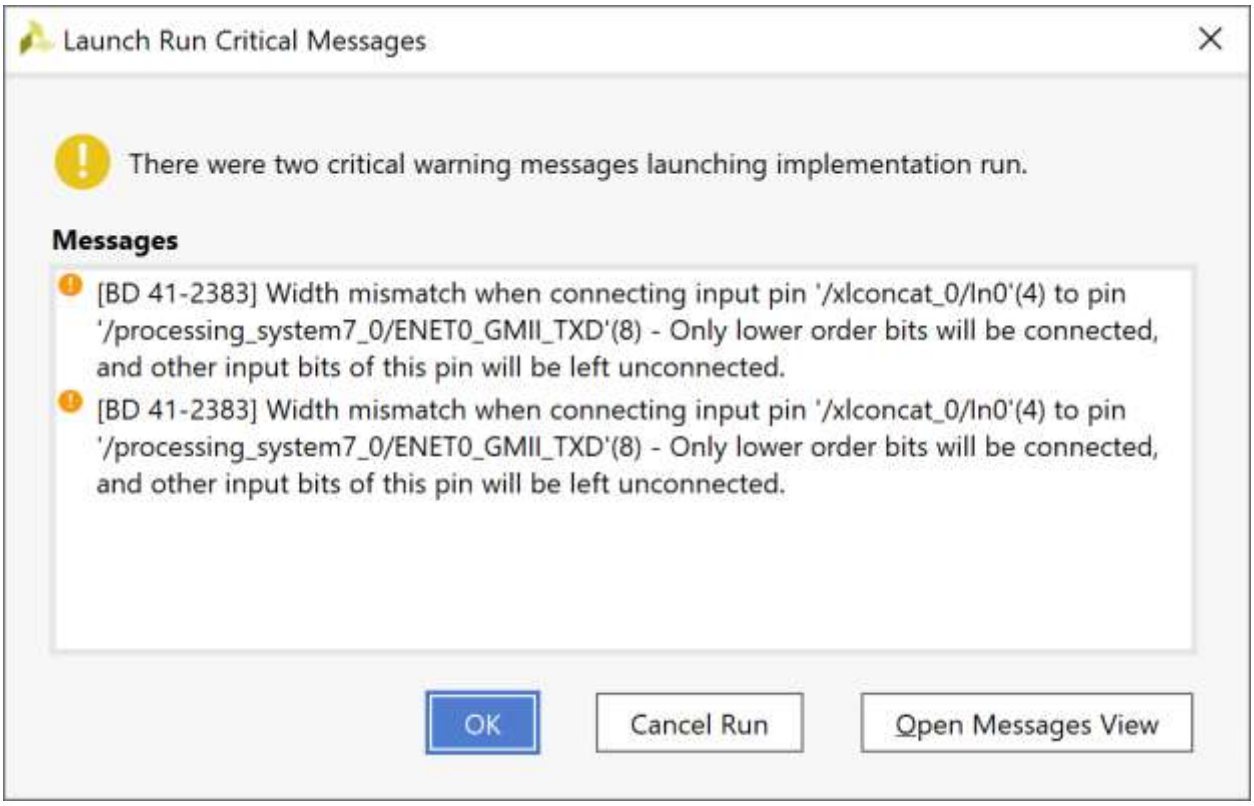
A warning shows up but click “Yes”.



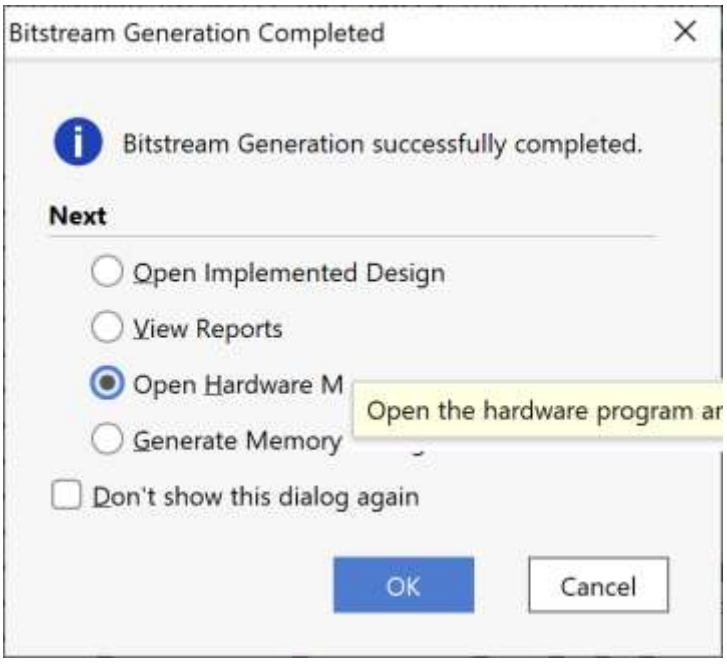
And then another window shows up, just click “OK”.



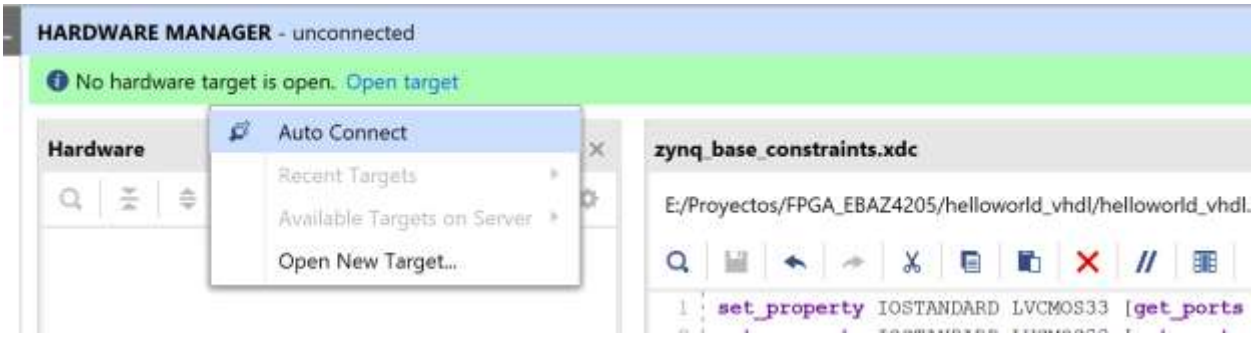
Will show up this warning, it’s safe to proceed. Press OK.



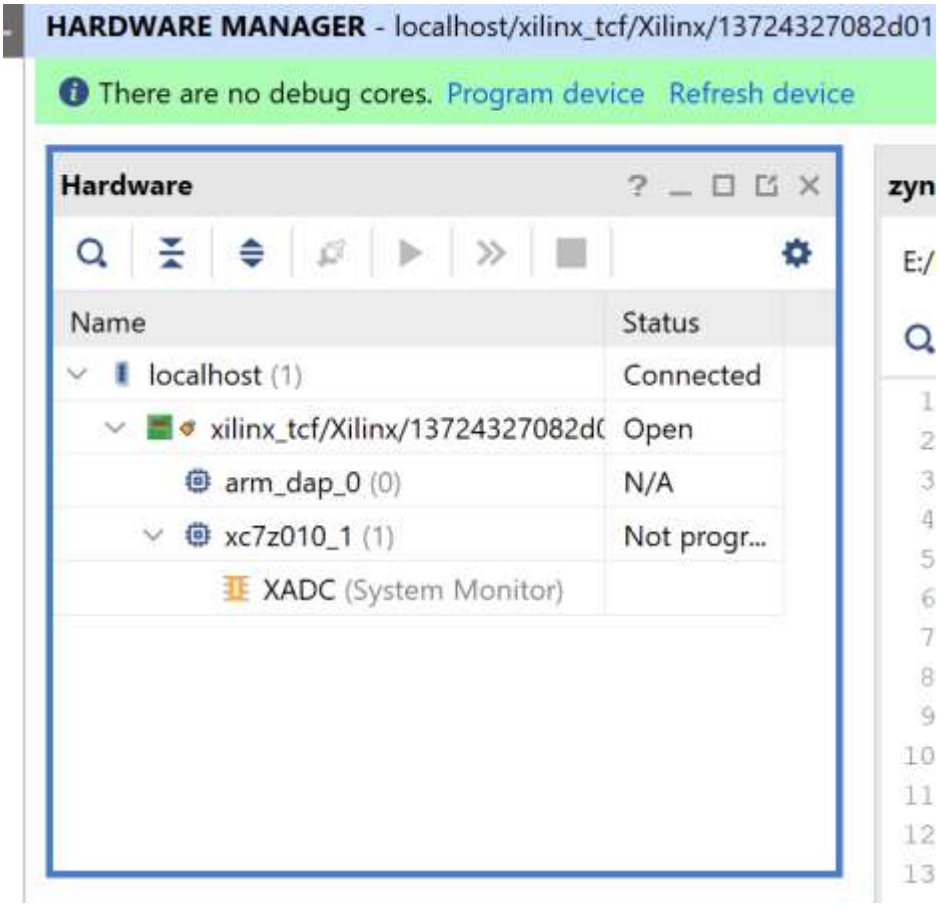
Once the process finish, you should see this (click on cancel)



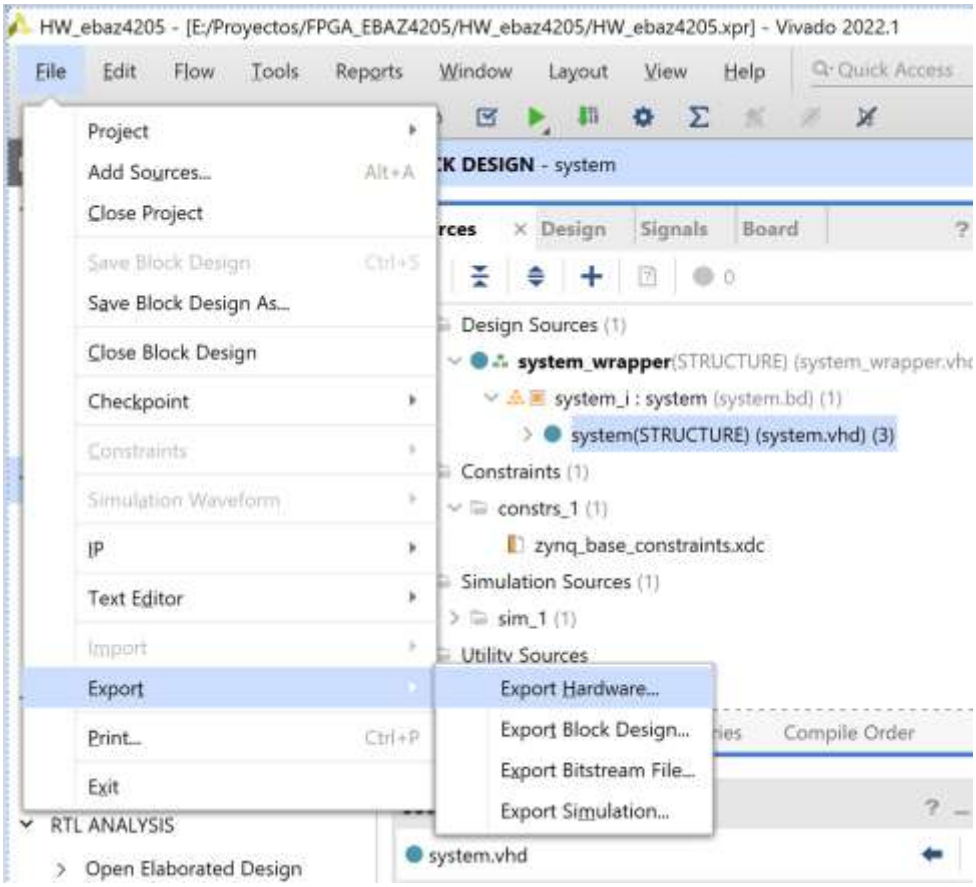
If do you have the JTAG programmer available, you can open de hardware manager. If not, you can jump to the next step. Power on the board, and connect the JTAG. If the status led is green, press on “open target”.



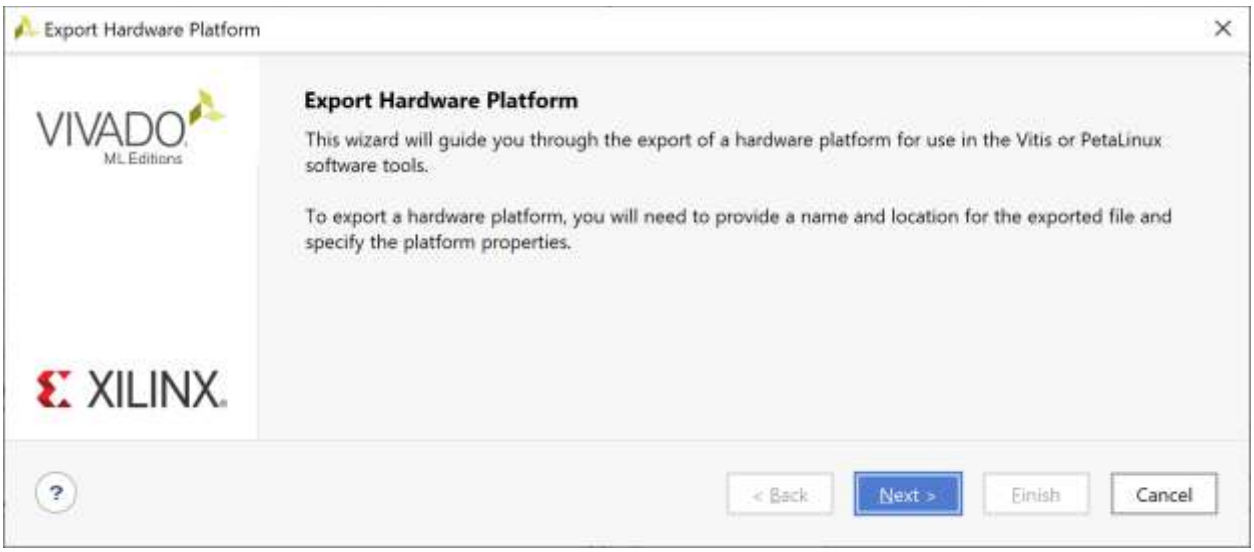
You should see the Zynq connected.



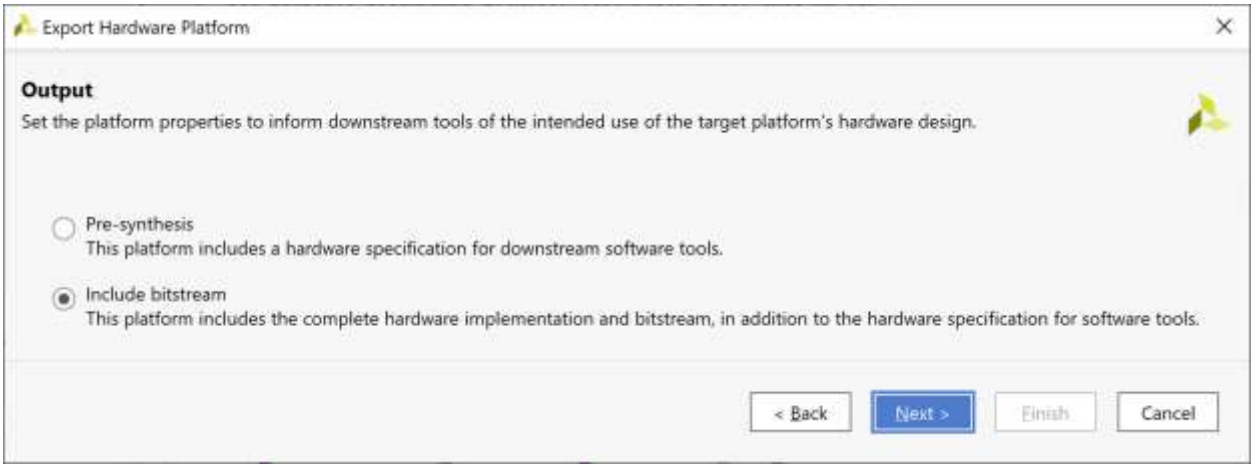
43. Now we need to export the hardware to the SDK. To do that, we need to go to File -> export… -> export hardware…



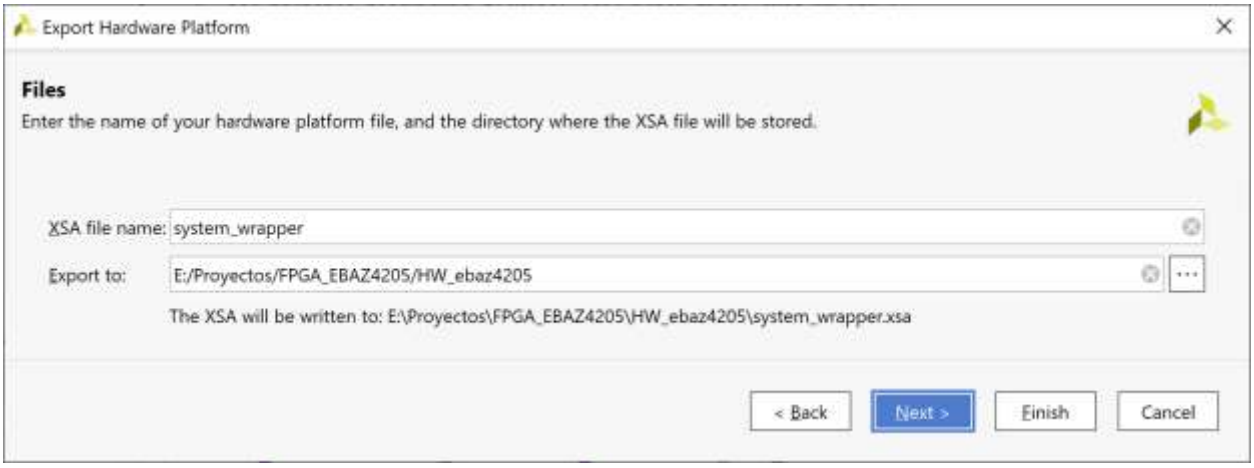
A new popup window shows up, click “Next”



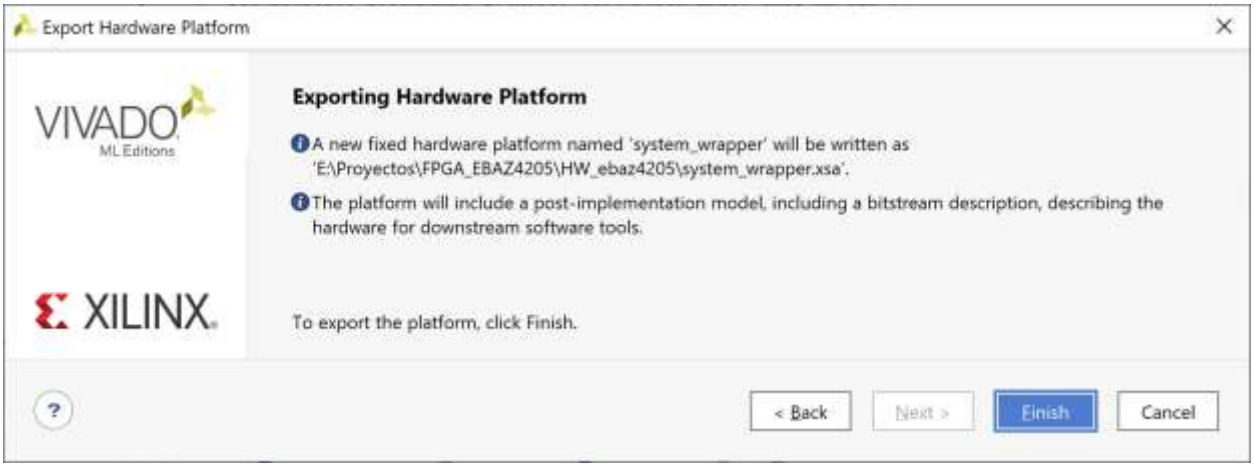
In the next window, we need to select “include bitstream” and click on “next”



In the next window, click on “next”



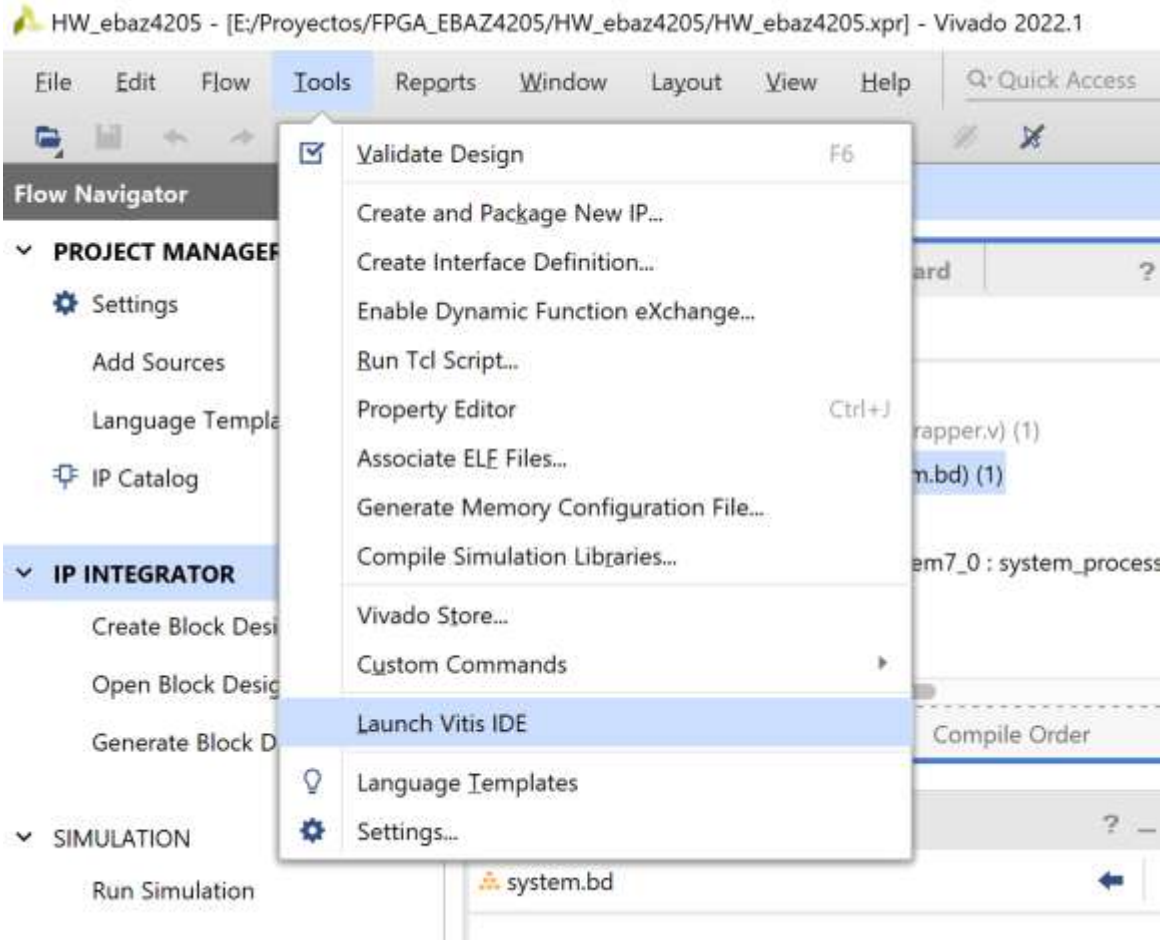
In this window, click “finish”



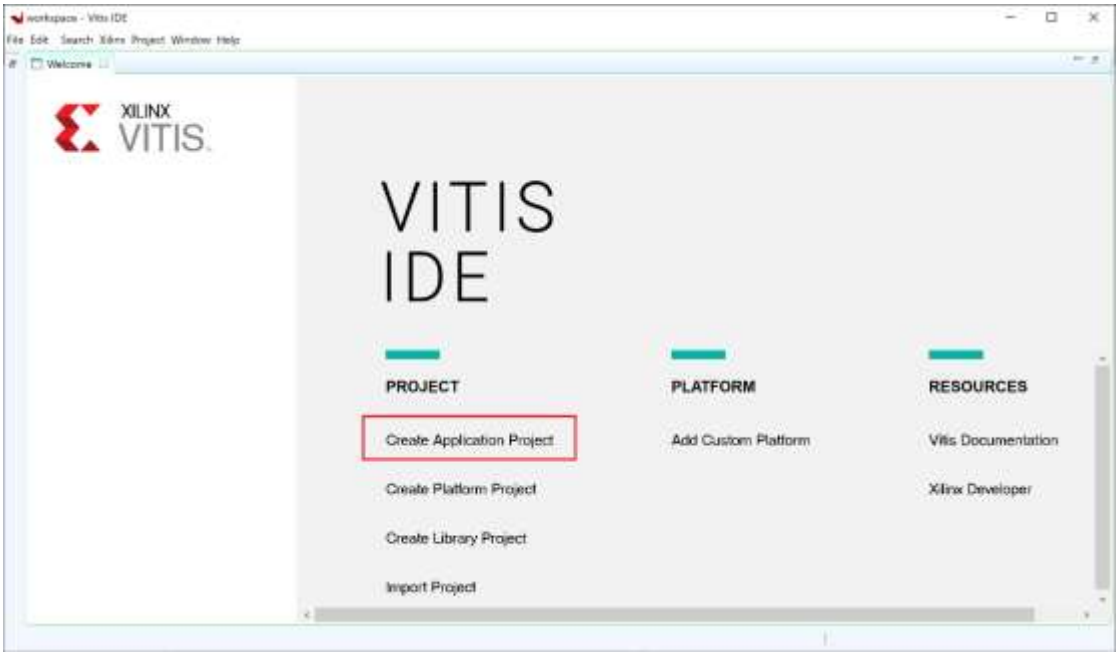
OK, this is the end of this section. Now we will continue with the SDK part.

Instructions to create the software:

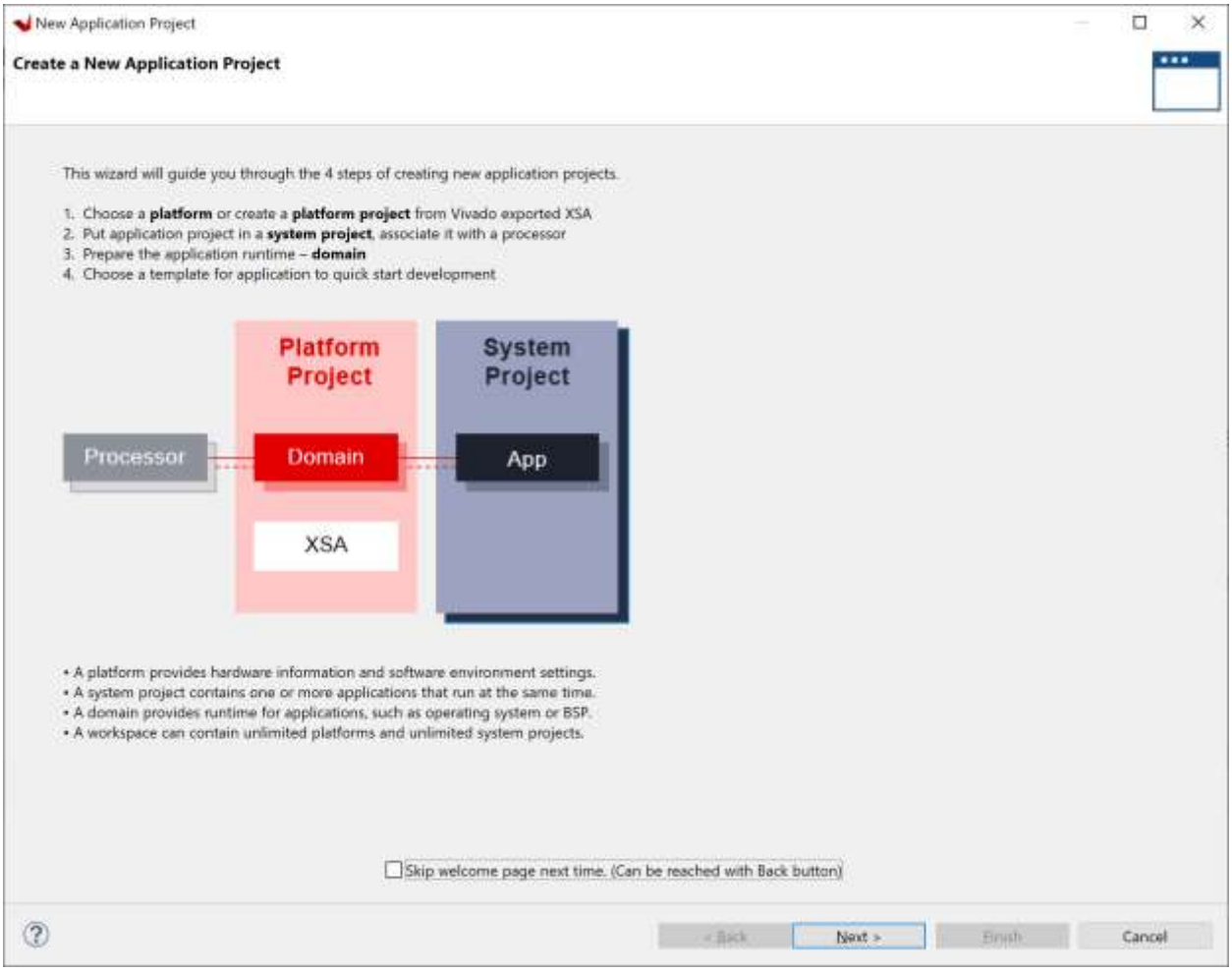
1. After the hardware export is complete, select Tools -> Launch Vitis IDE from the menu bar to launch the SDK development environment. As shown below:



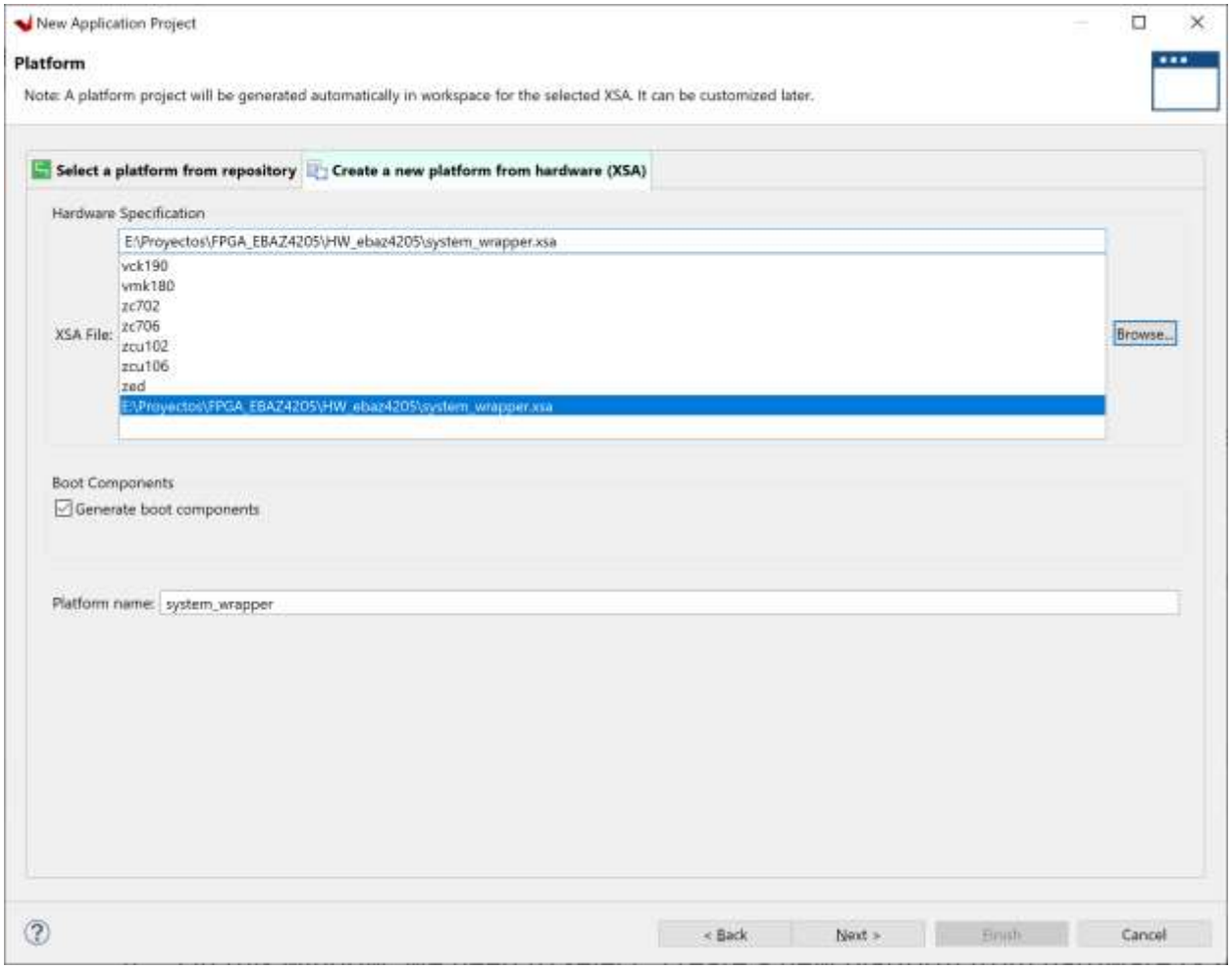
2. If the first time you launch it, a popup window will ask you for default location of your workspace. Click OK and continue. Then you should see this window, and click on “Create application project”



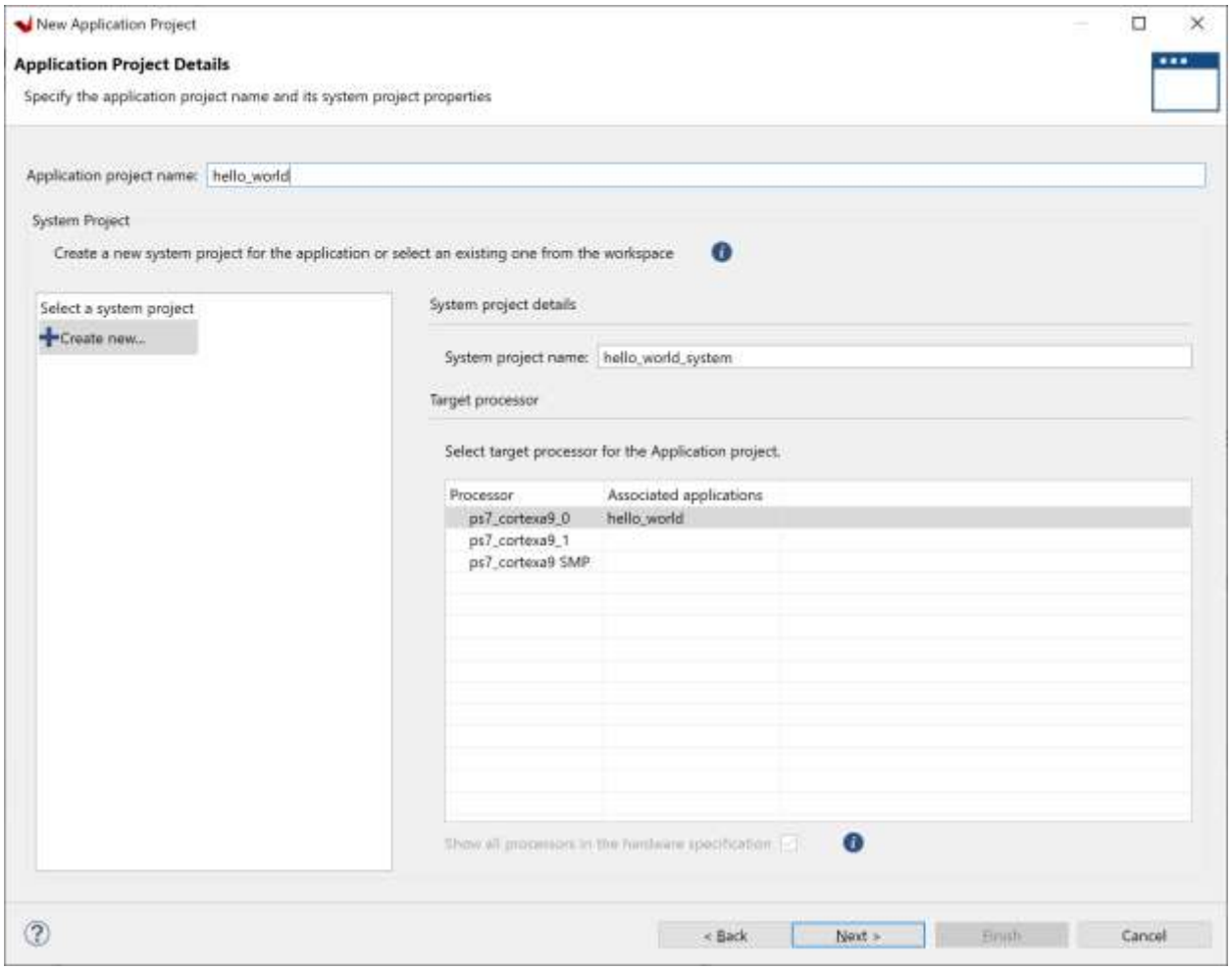
3. In this window click on “next”



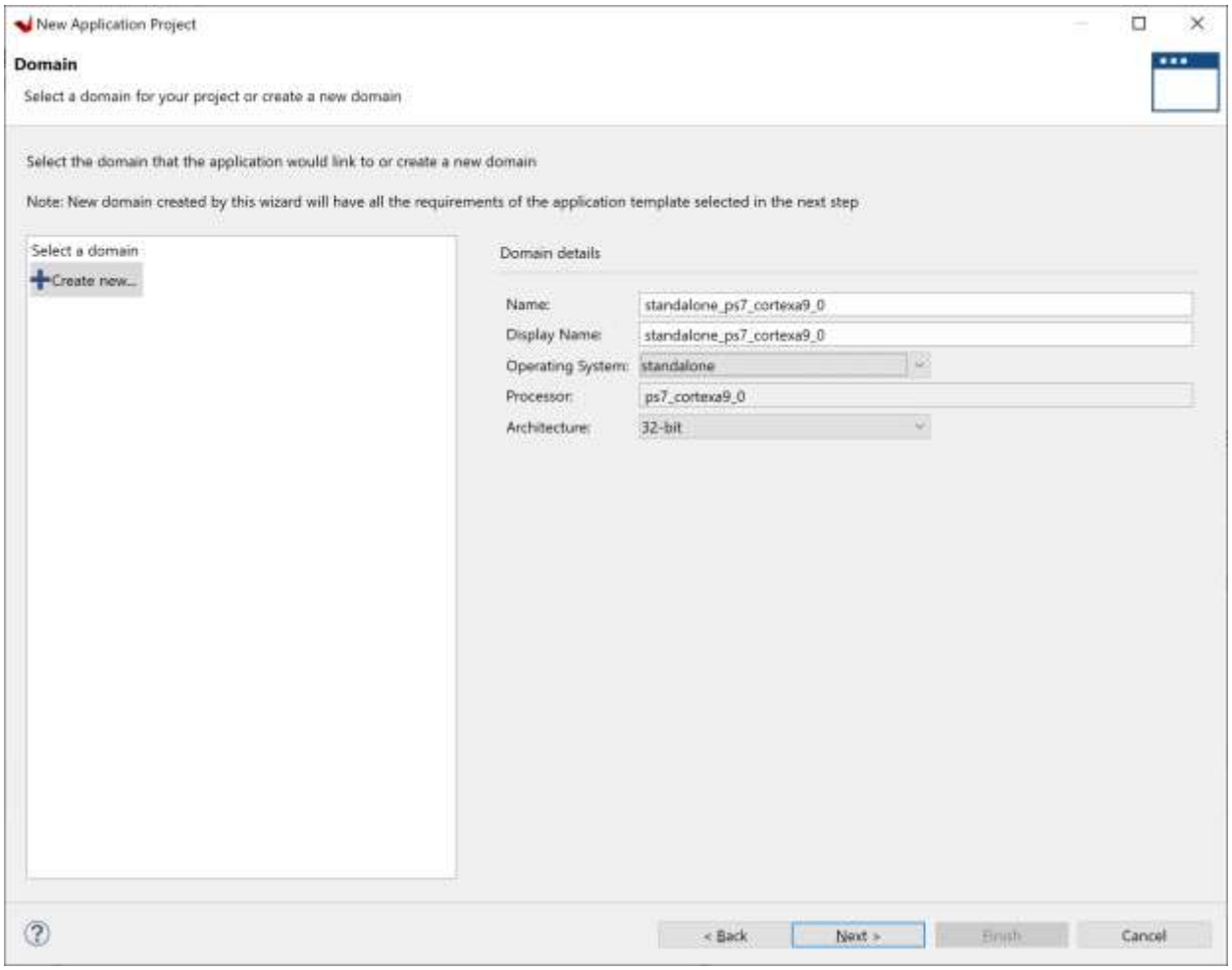
4. On this window, we need to select “create a new platform from hardware (XSA)” tab and then click on “browse”. Search for the project directory created previously and you will find the XSA file on the root. Then we click on next.



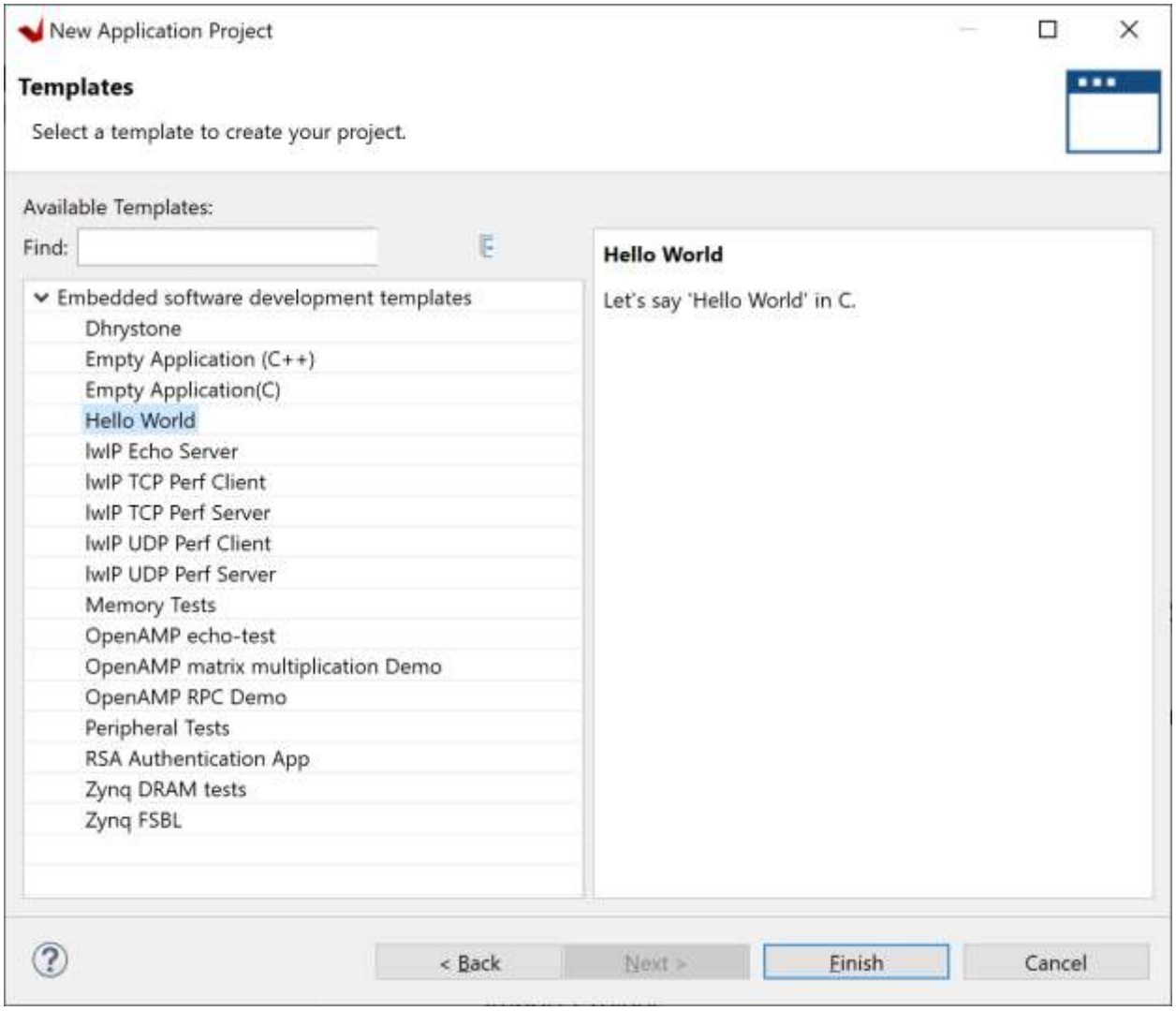
5. On “application project” field we select “hello world” and then select as processor “ps7_cortexa9_0”. Then click next



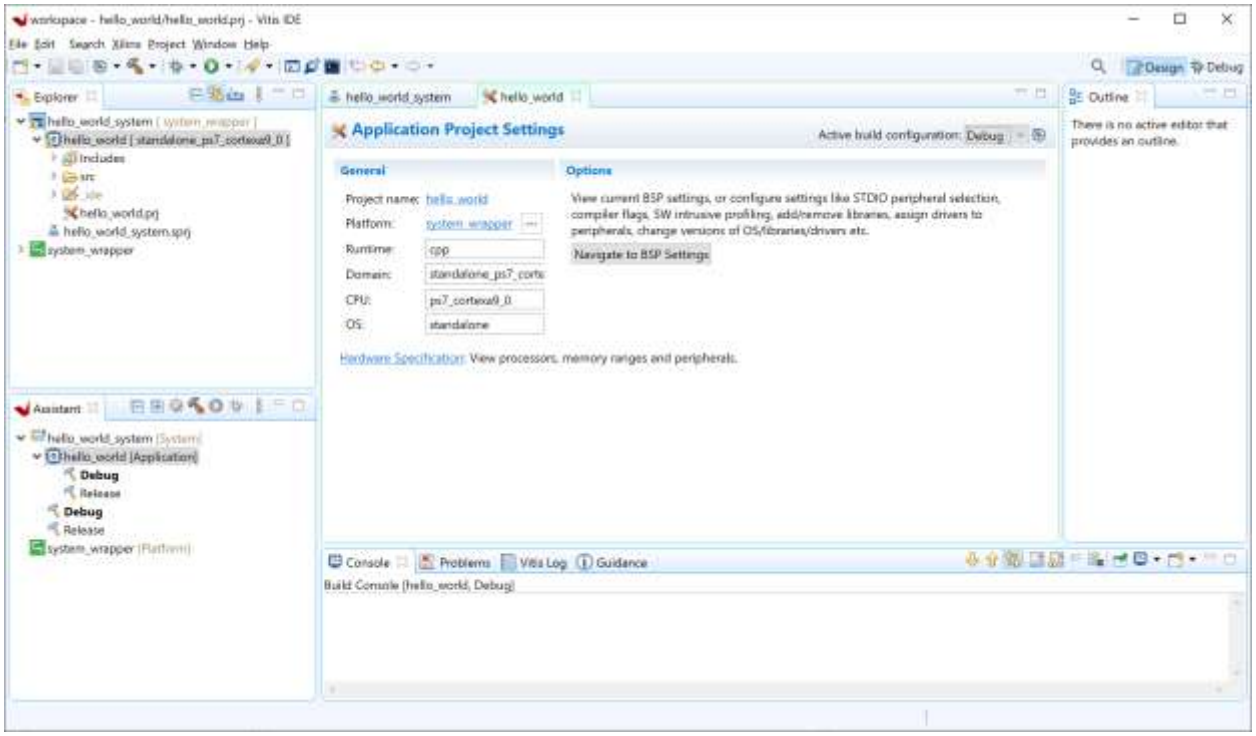
6. On this window just click on next.



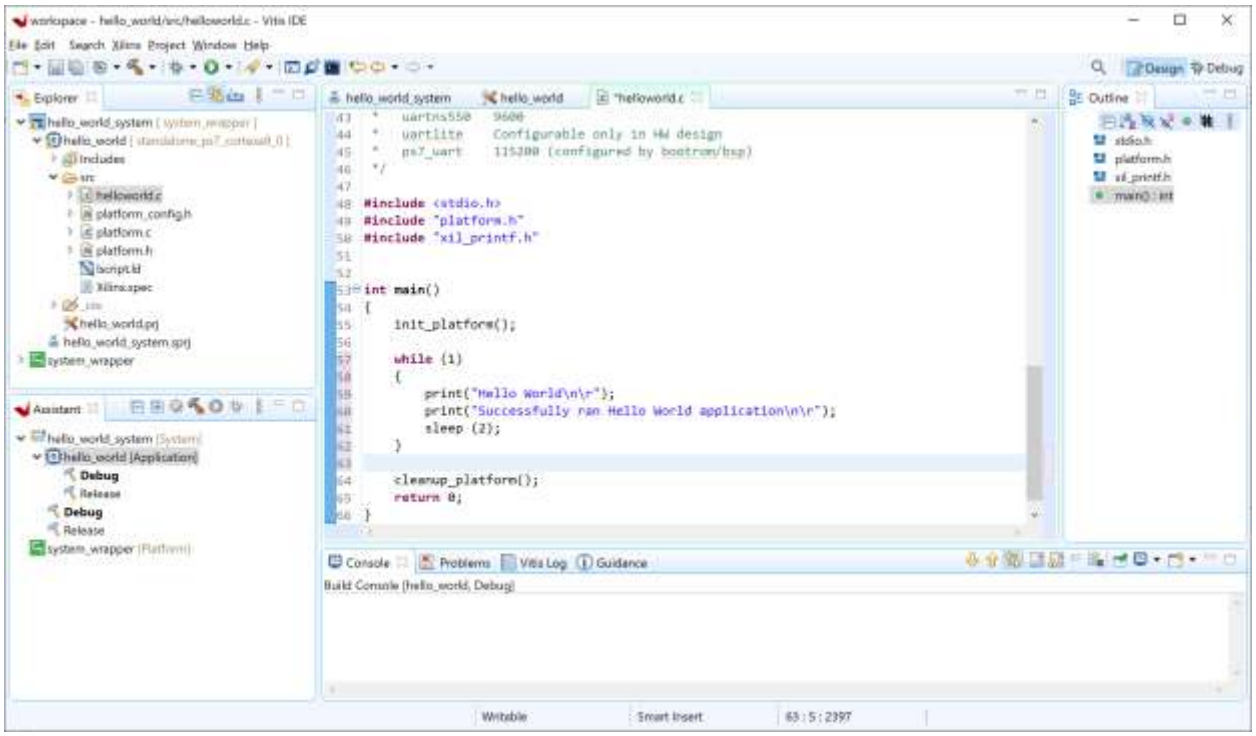
7. Now we need to select the project. Select hello world in C and then click on finish.



8. The result is the following



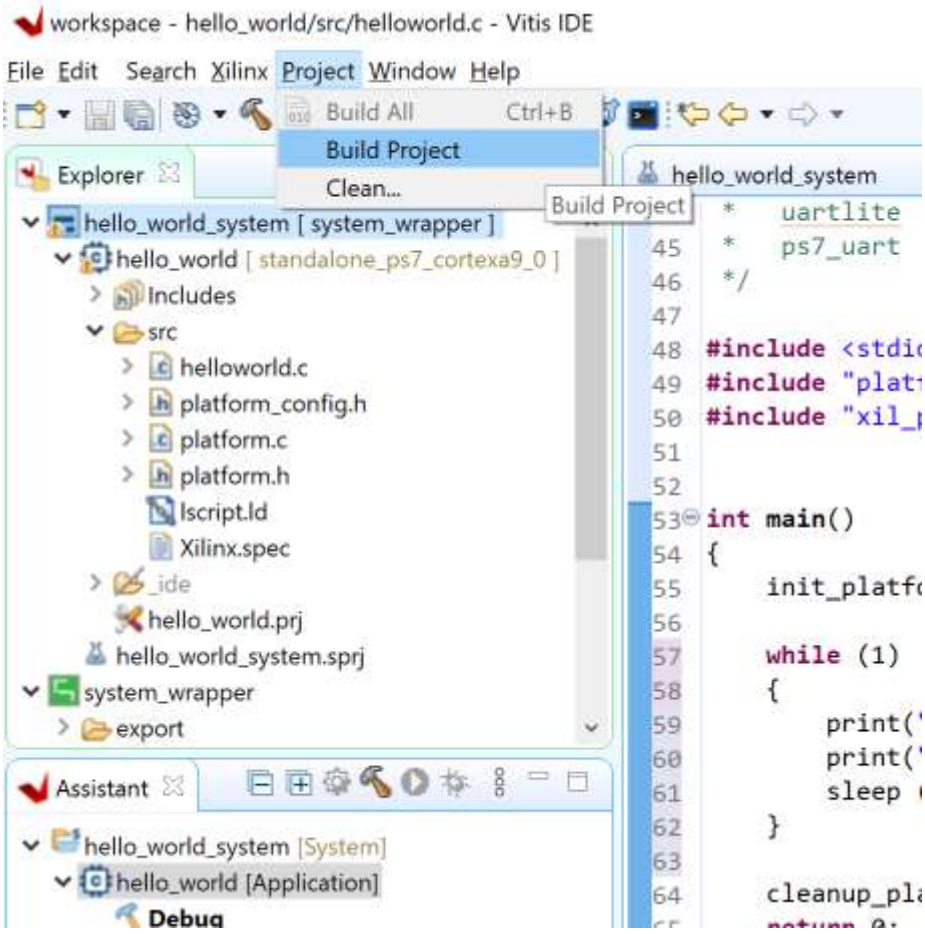
9. On the left panel, go to src folder and then double click on “helloworld.c”, This will open the file content. I would like to add a loop to print hello world on the serial port every two seconds.



Function in plane text to copy/paste:

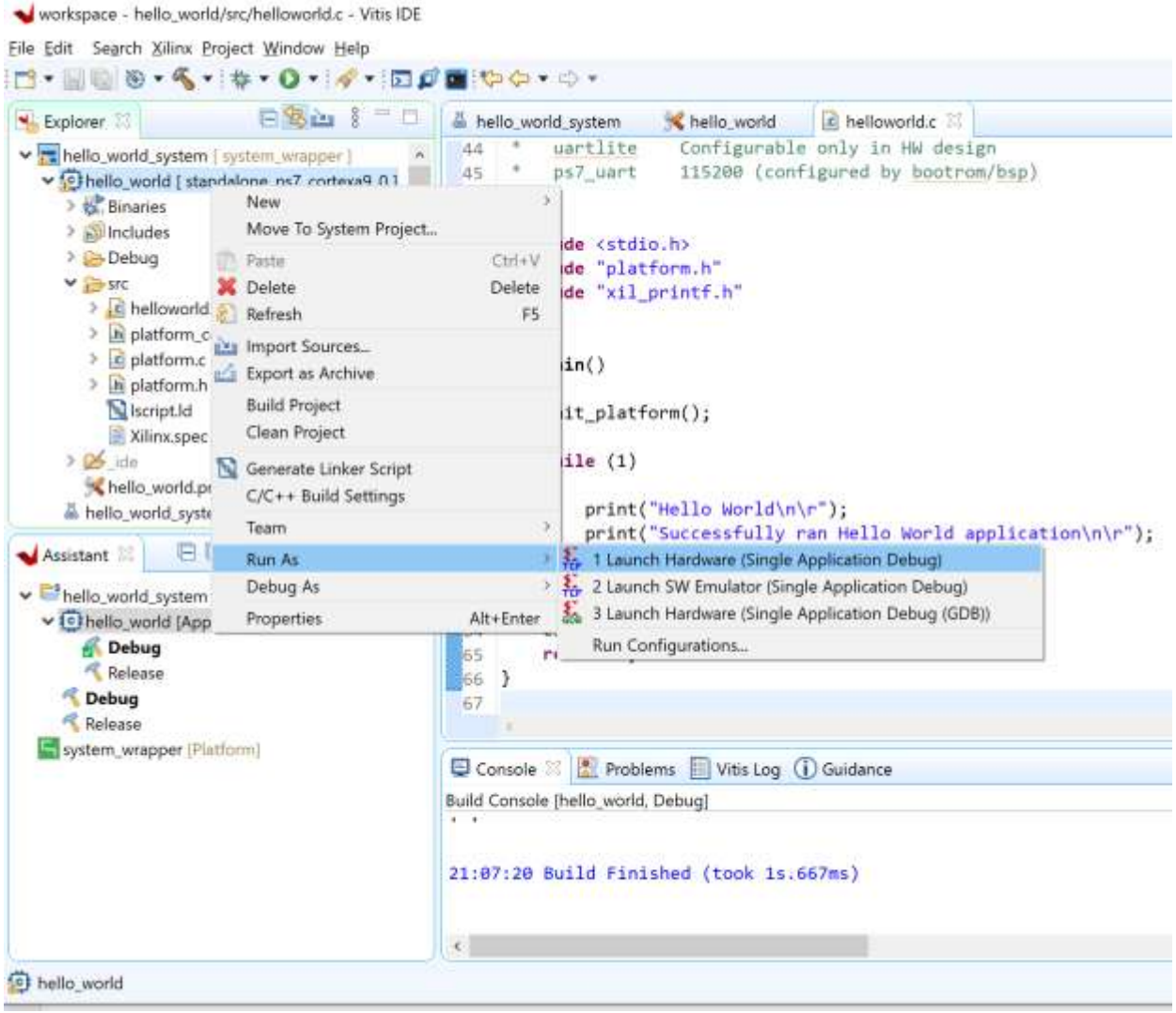
```
int main()
{
    init_platform();

    while (1)
    {
            print("Hello World\n\r");
            print("Successfully ran Hello World application\n\r");
            sleep (2);
    }

    cleanup_platform();
    return 0;
}
```

10. Now we go to Project -> build project.

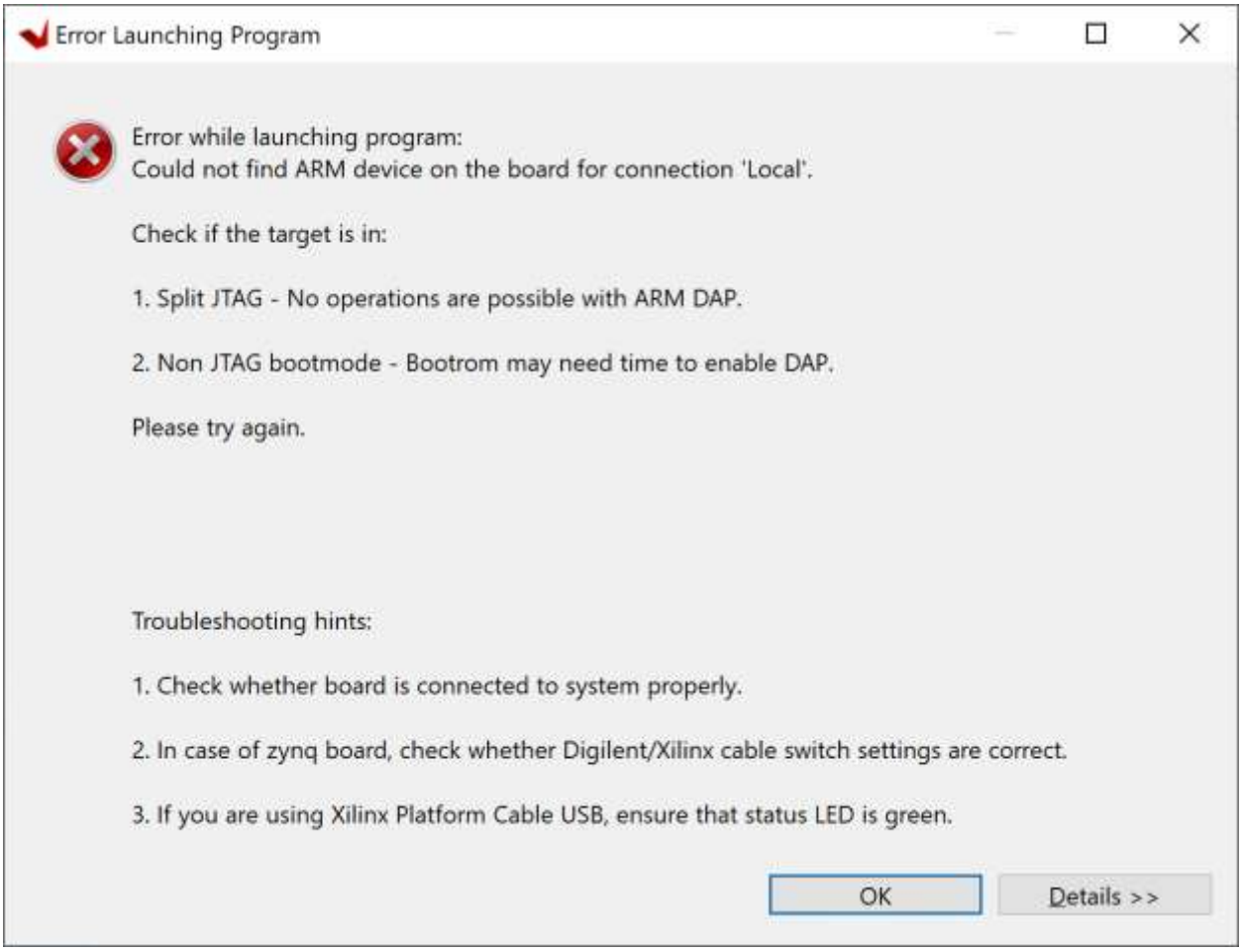


11. Then on the message box you should see it finished with no errors.

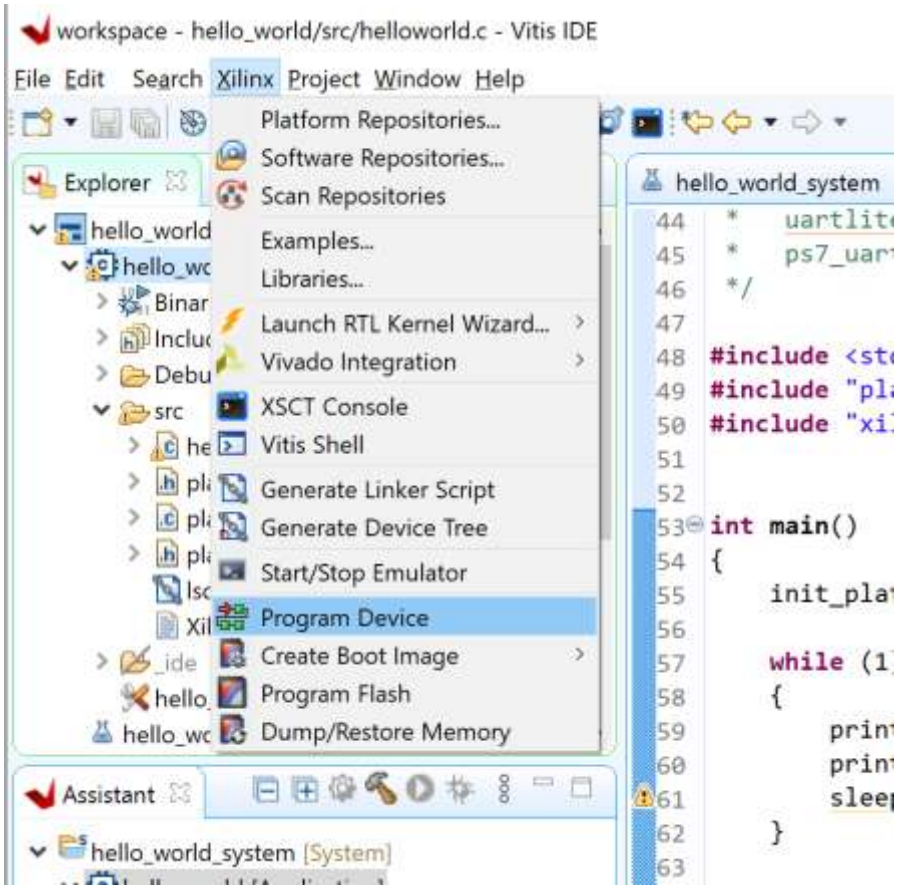
12. Let's try to run the software on the board. To achieve this, we should make sure that the boot mode configuration of the board is JTAG mode. We will need to power the board, connect the USB serial board, and the JTAG. Then, go to hello_world, do right click, select run as, the select "launch hardware".



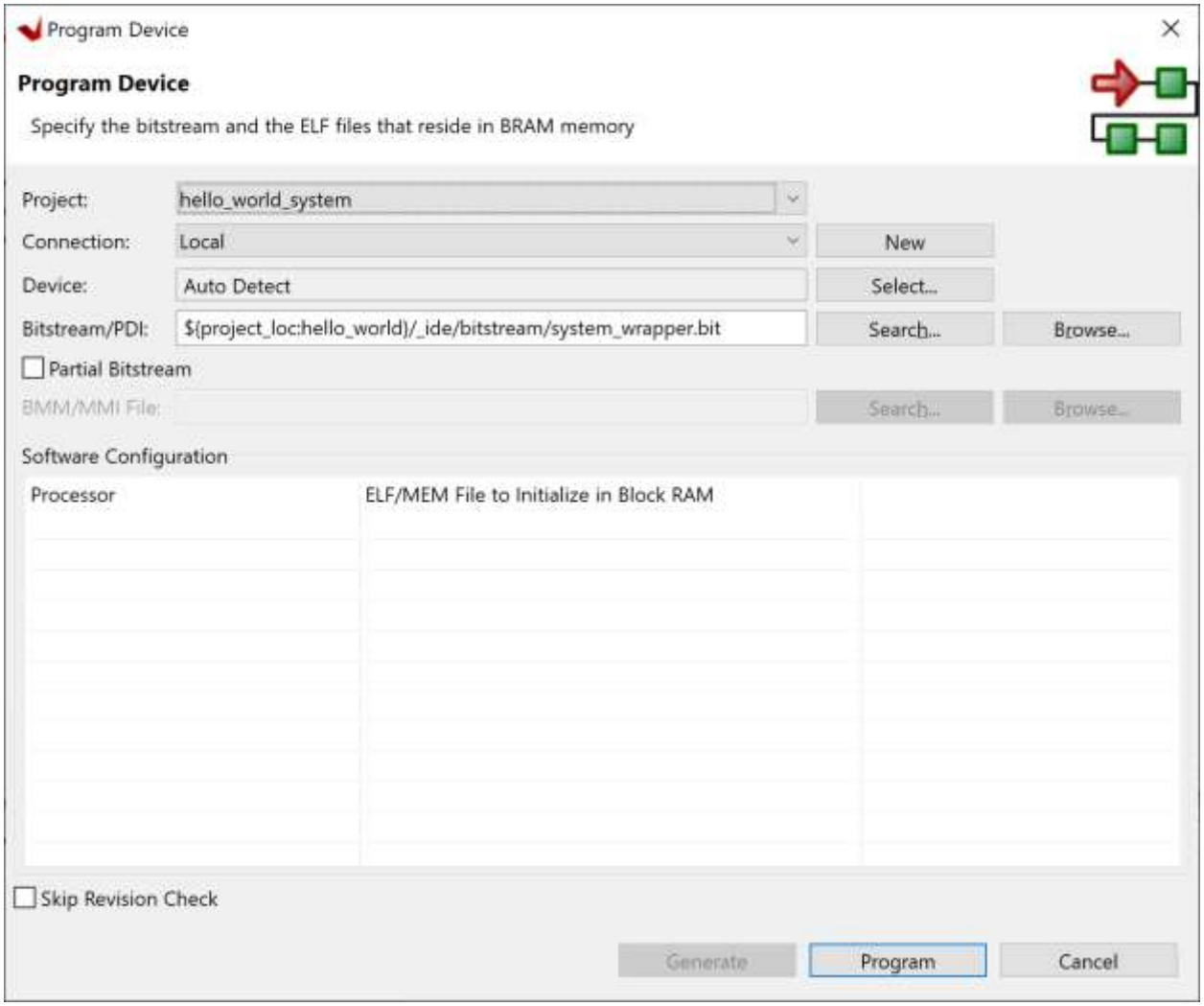
Using this, I got an error:



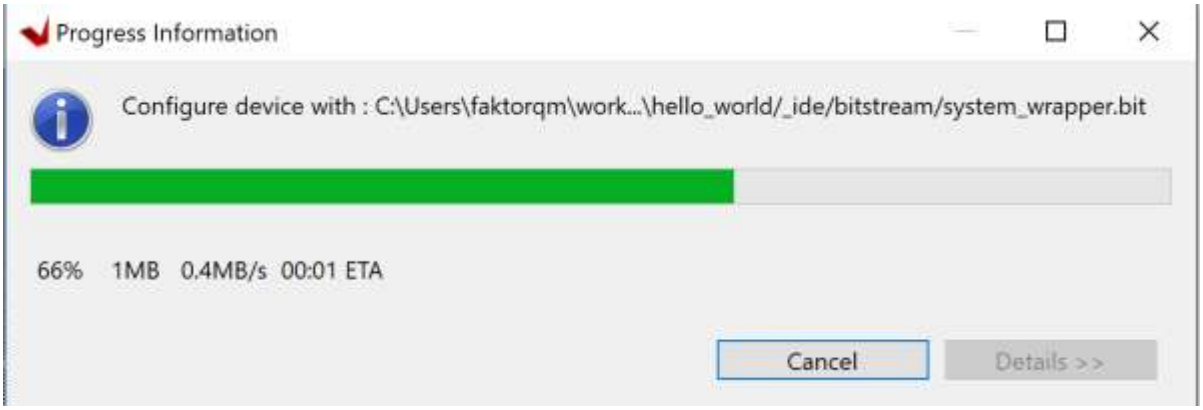
13. Let's do this:



Then click on program

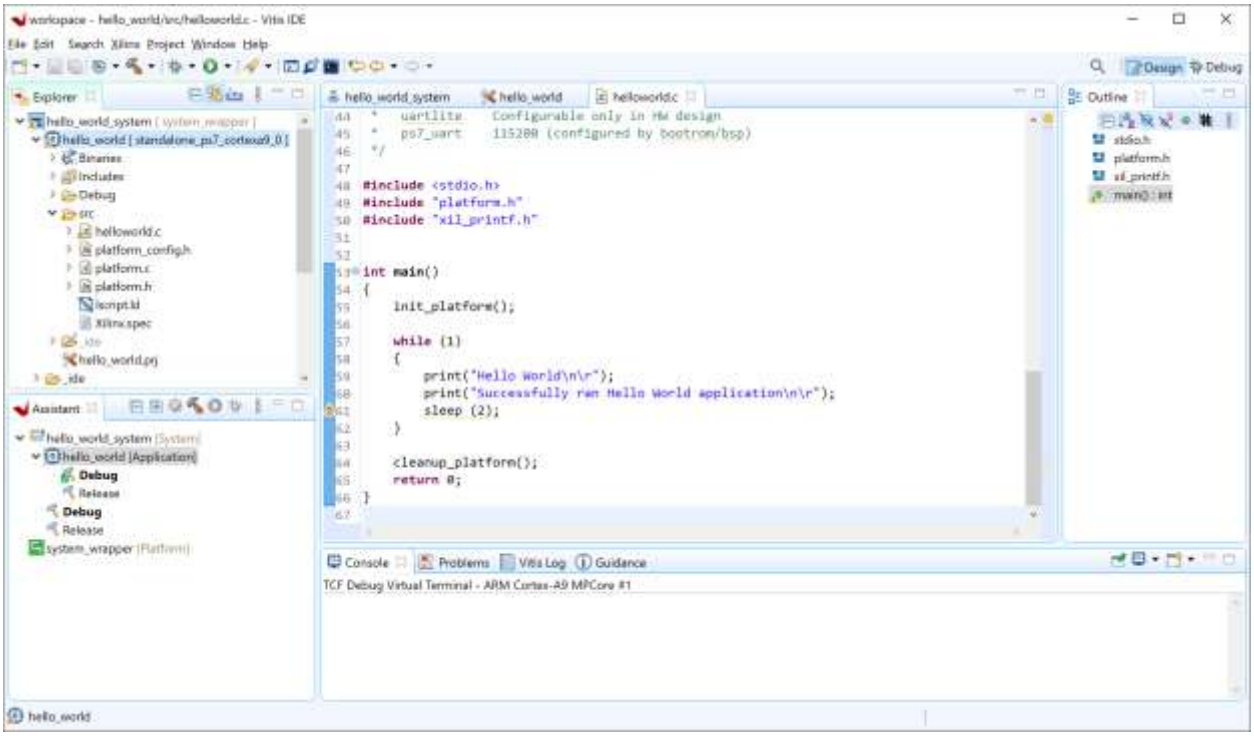


And you will see a window with the progress if you have everything connected.

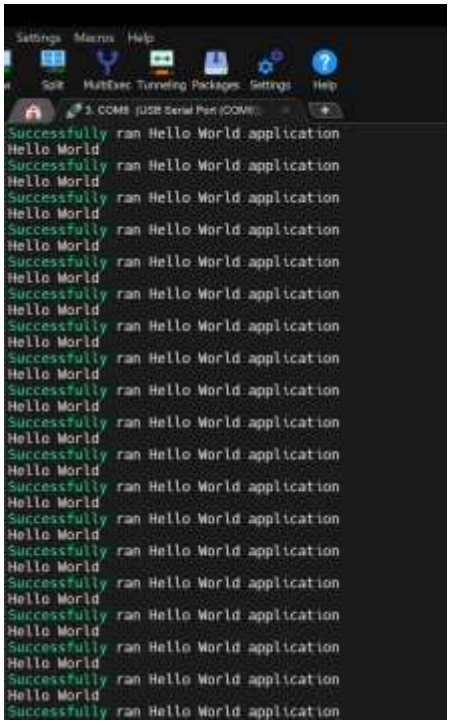


IMPORTANT: Do not reset your board, you will lose the changes made.

Repeat step 12 to actually run the program desired. The result should be this, with the debug console opened



14. I opened the UART port using MobaXterm. This is the result:

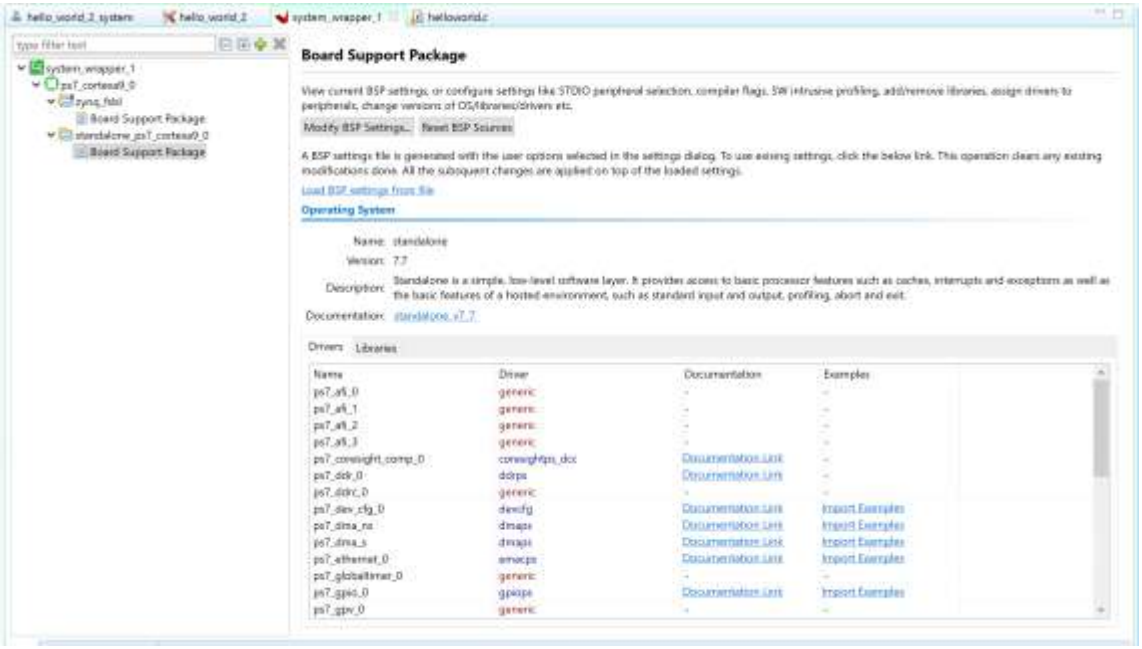


Congratulations! The program is working fine!

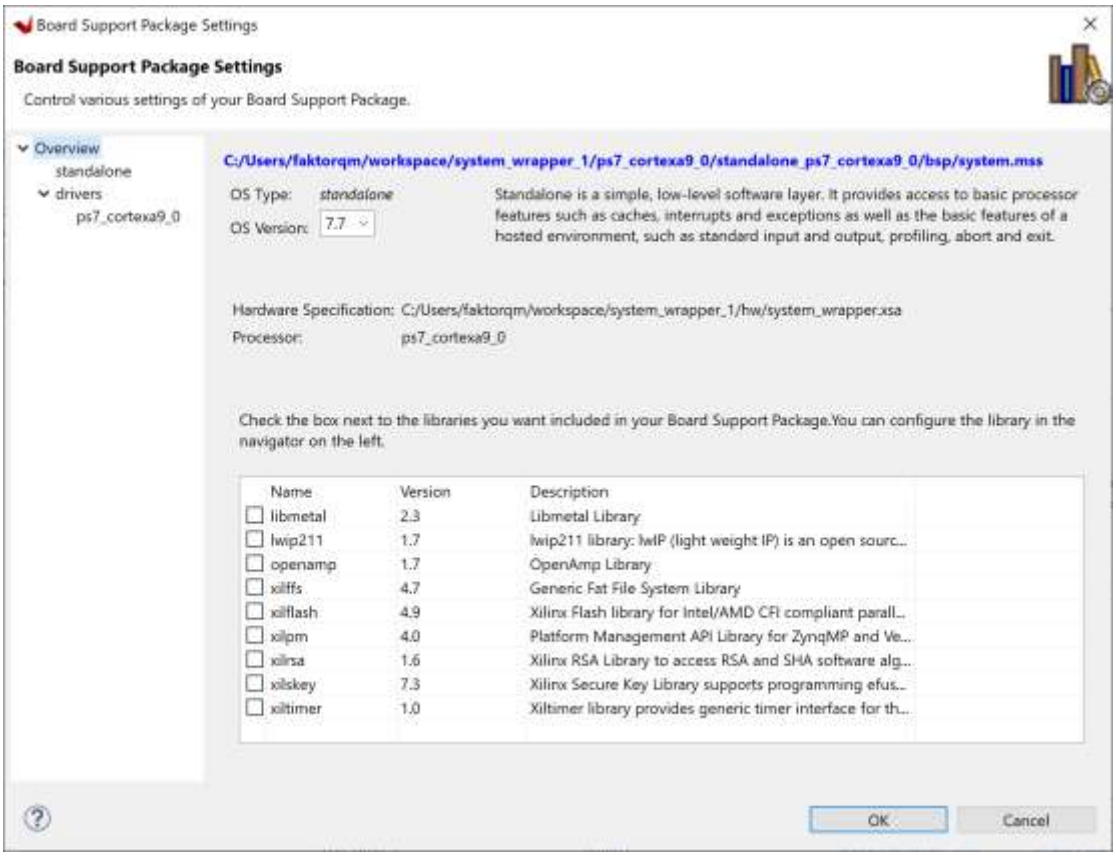
Steps to put this software on the NAND:

1. Go to “Hello_world.prj” file and select the button “Navigate to BSP Settings”.

2. Select “Board Support package” under “standalone_ps7_cortexa9_0” in the left panel. Then click on “modify BSP Settings…”

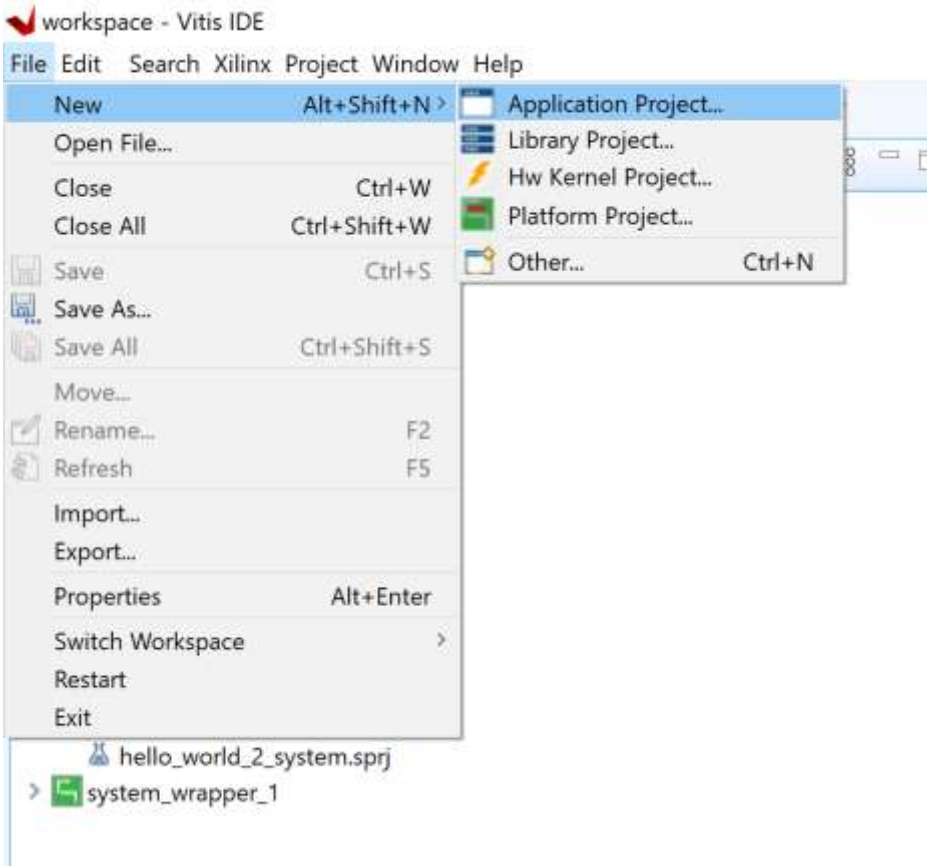


3. In the next window, we need to select “xilffs” and press OK.

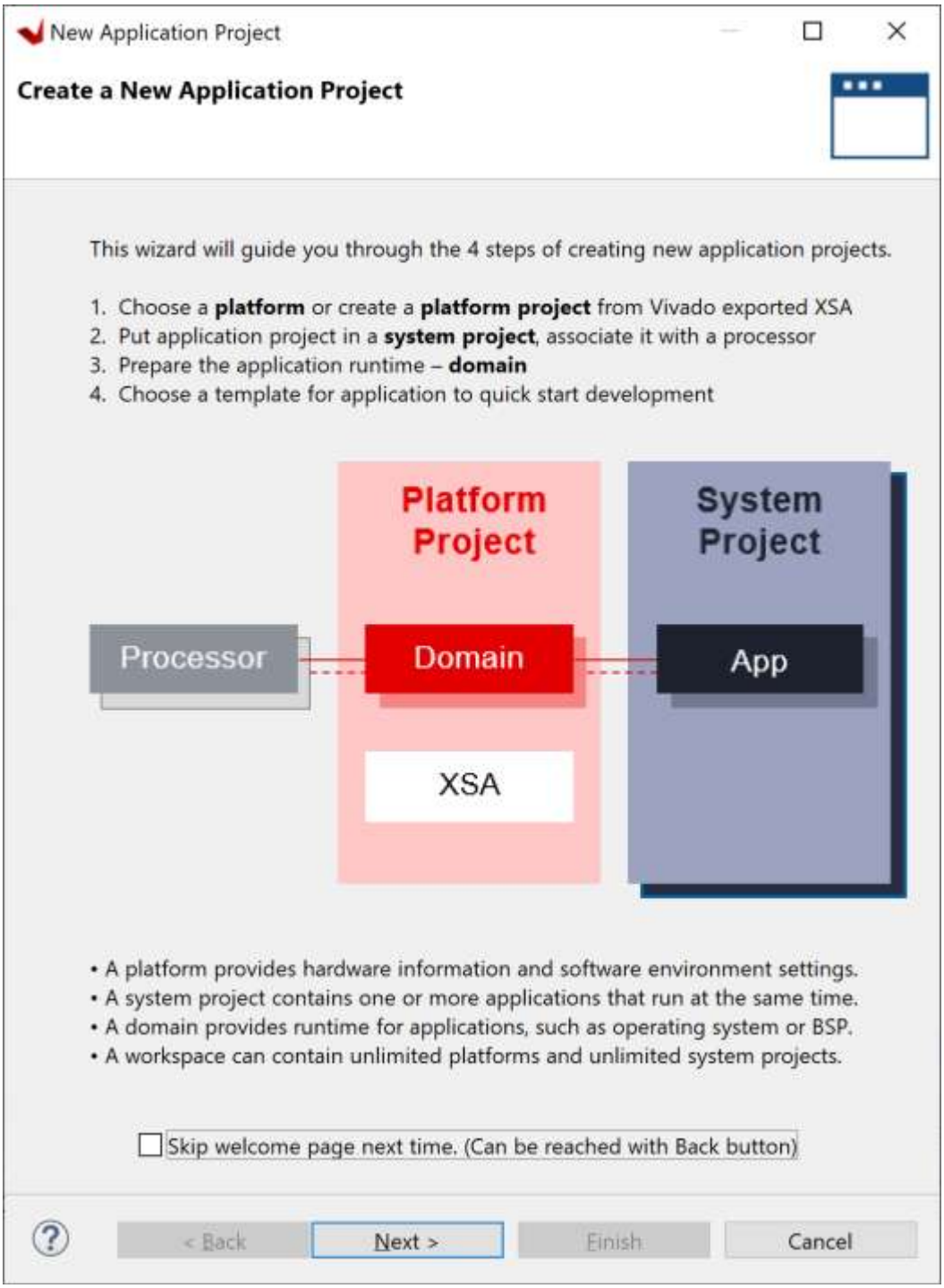


4. Then you will see a window like this and then it disappears.

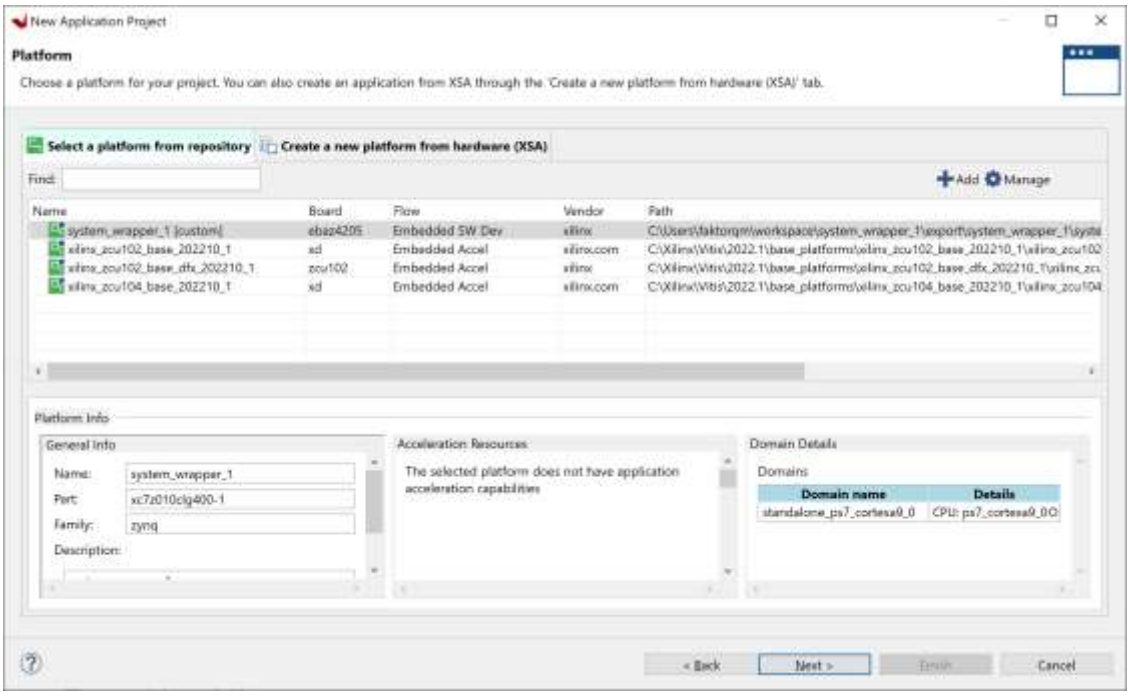
5. Go to Project -> Build all. Once it finishes,
6. Go to Select File -> New -> Application Project…



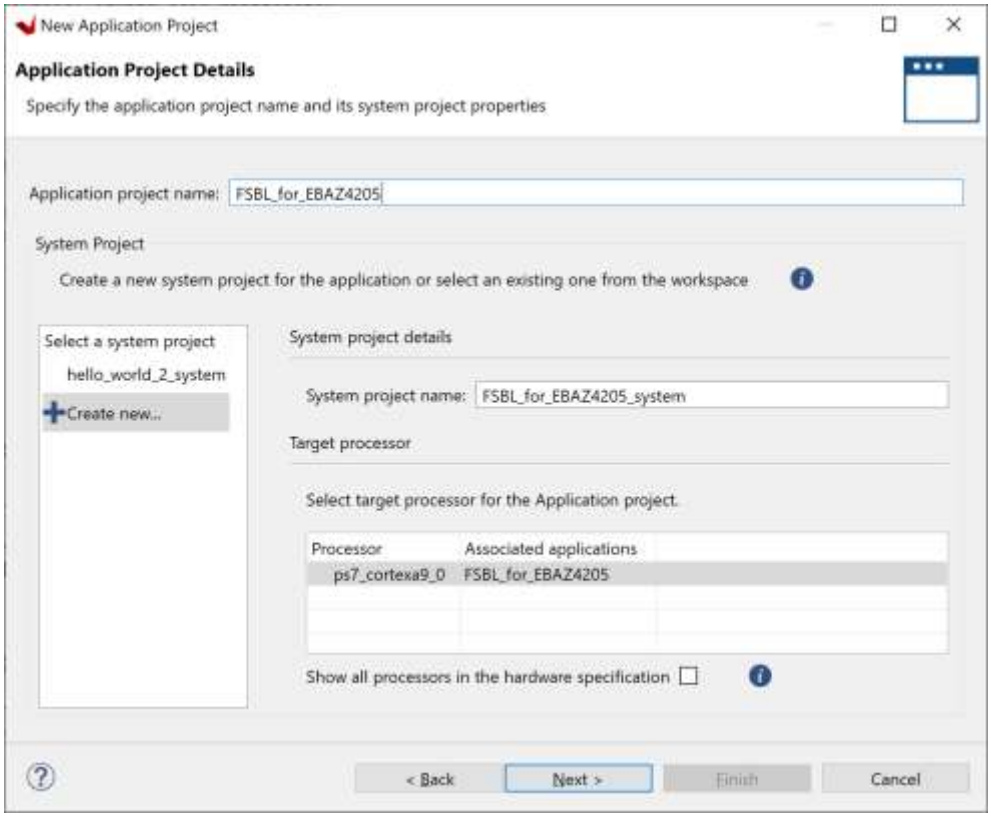
7. Then we will see this screen, press on next



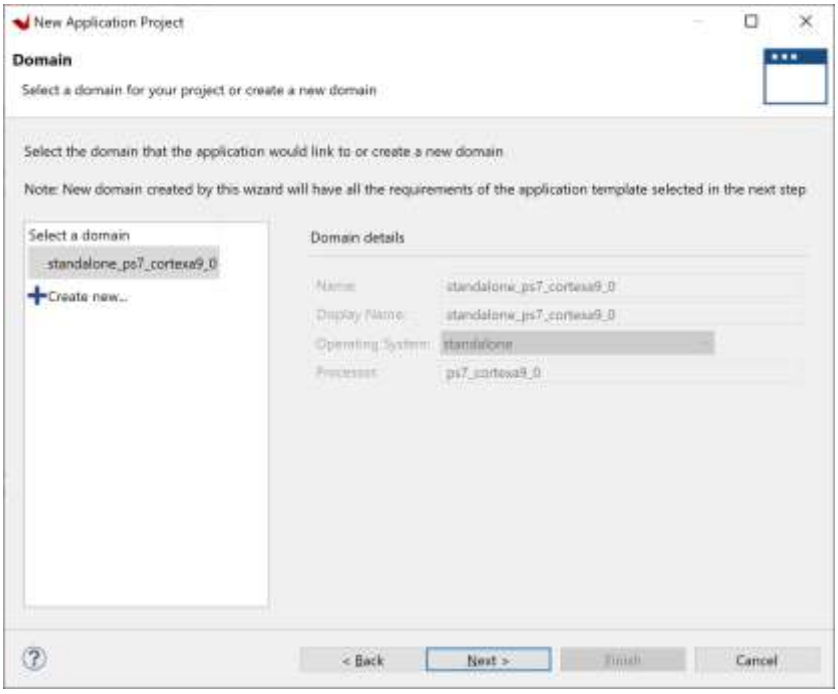
8. We need to select the same platform we have been working on and click Next.



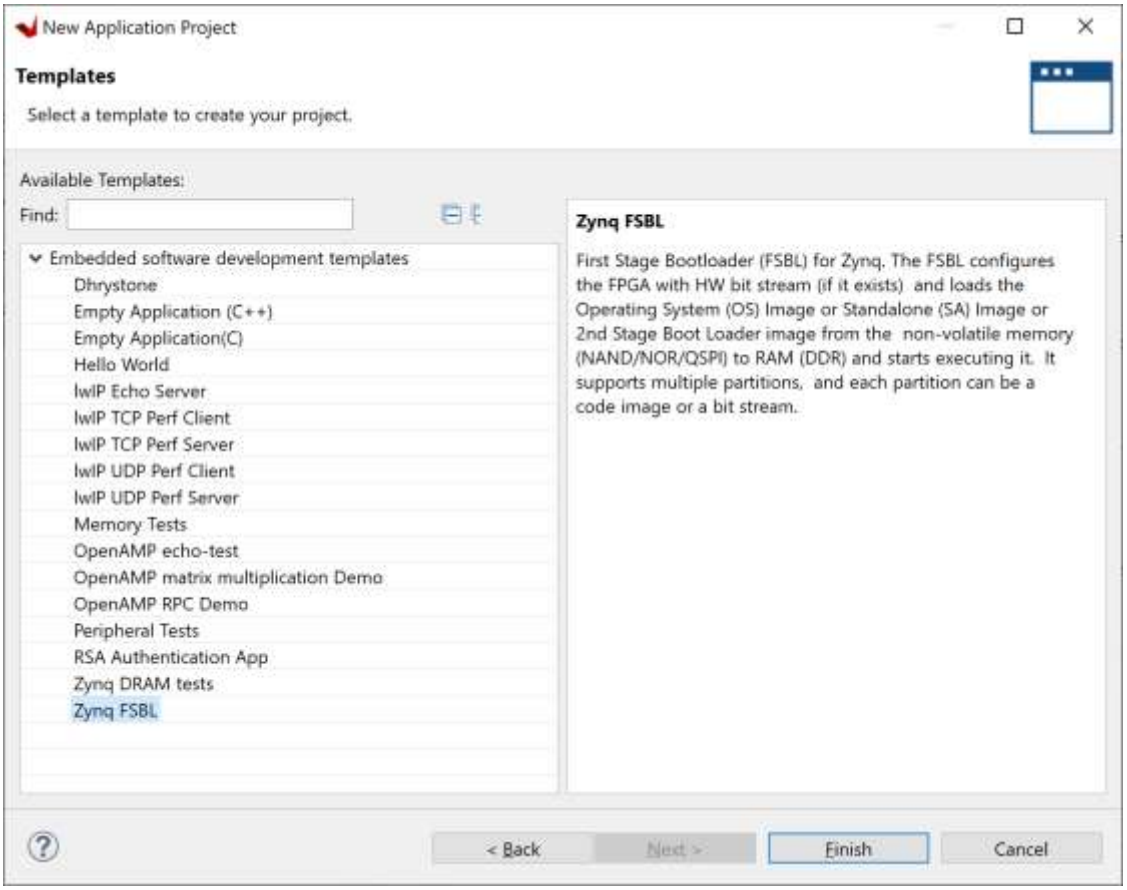
9. We need to assign a project name and click on next.



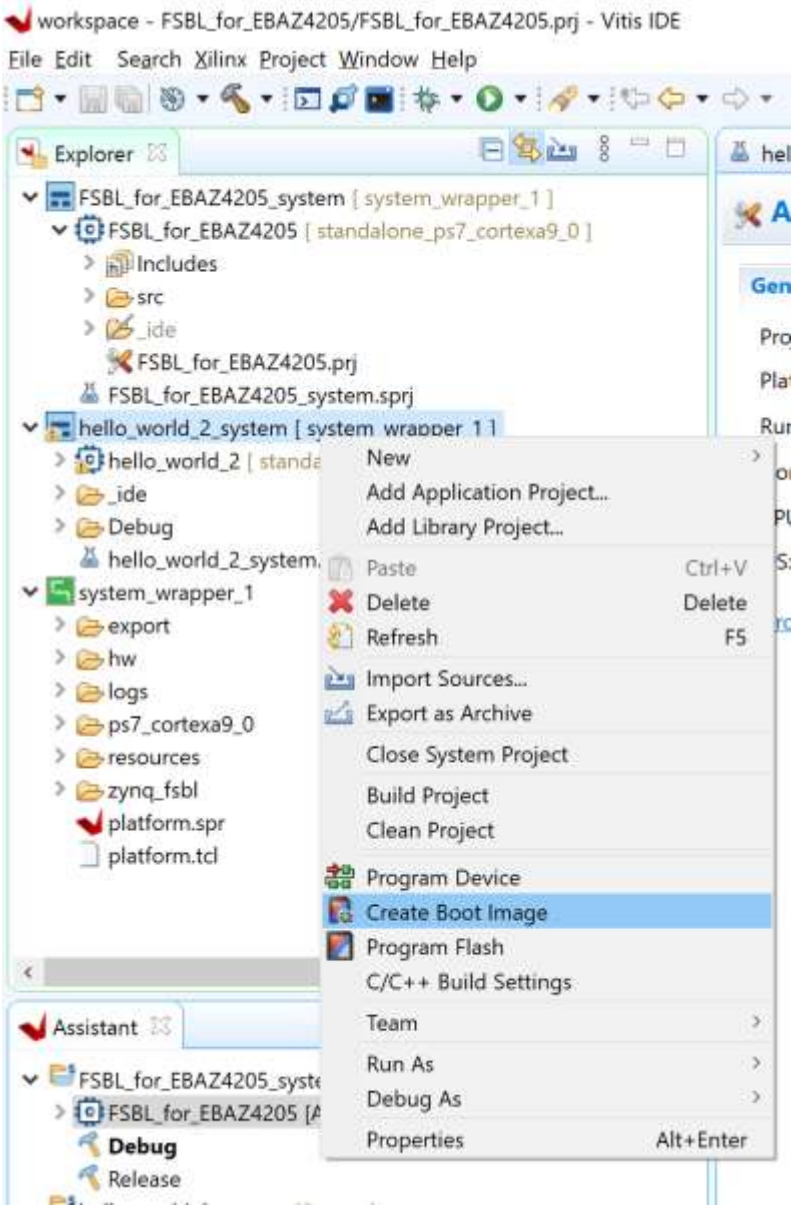
10. Click on next



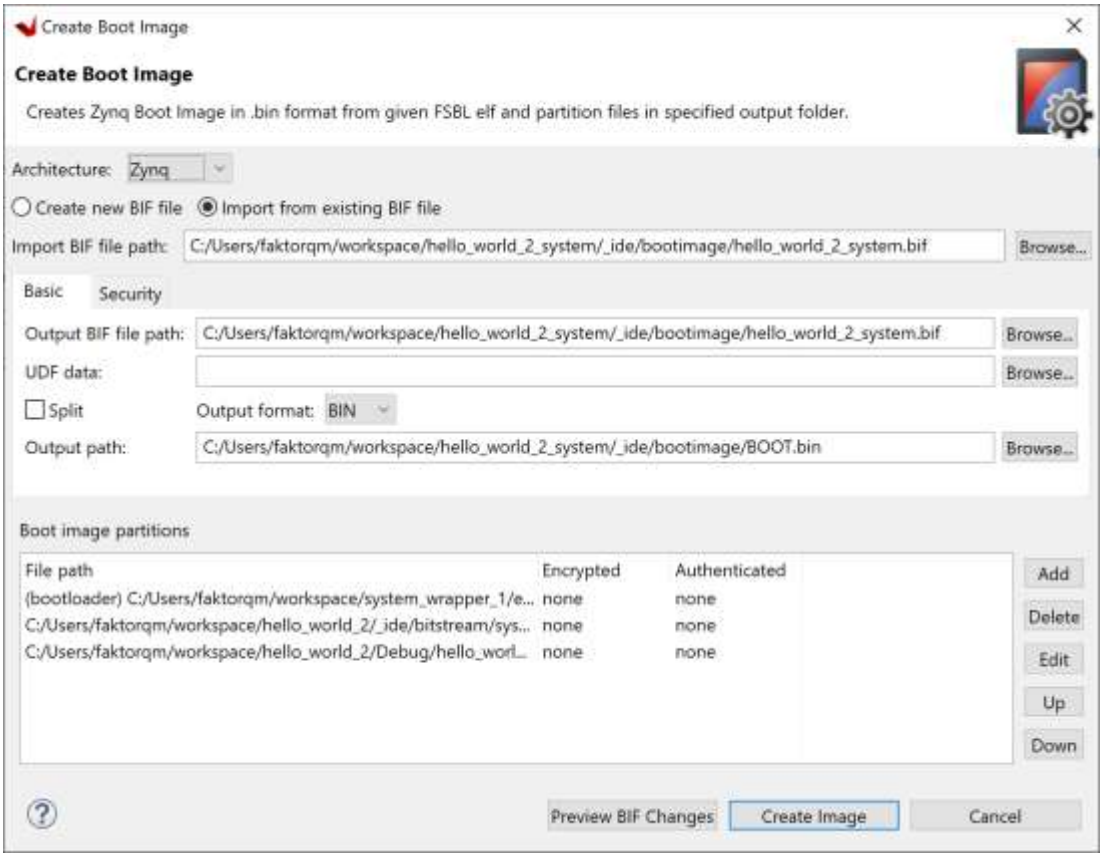
11. Select Zynq FSBL and click on Finish.



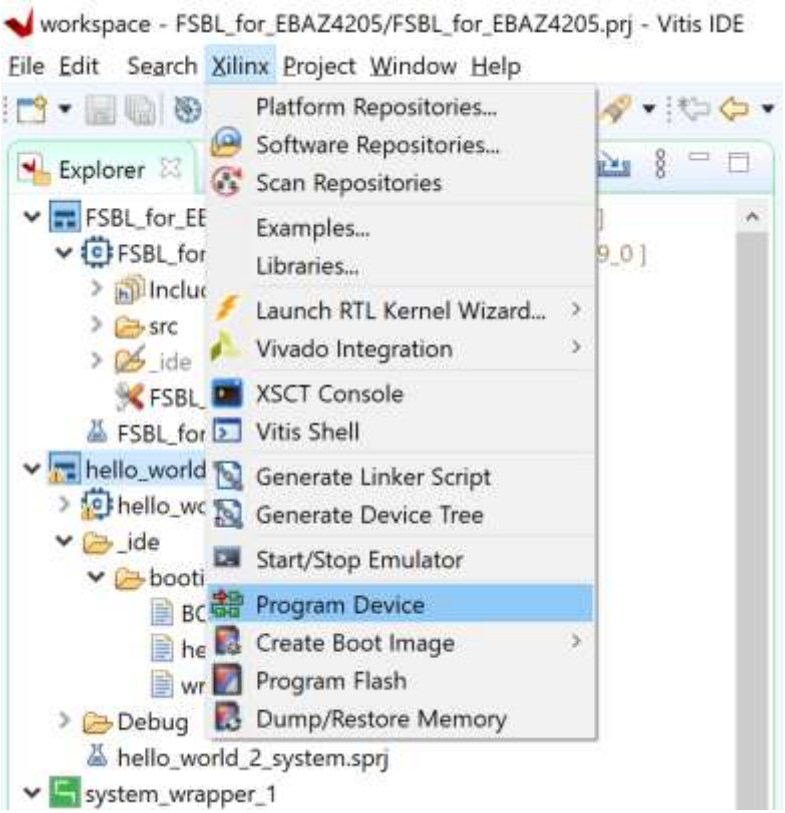
12. Do right click on the application project (not FSBL) and select “create boot image”.



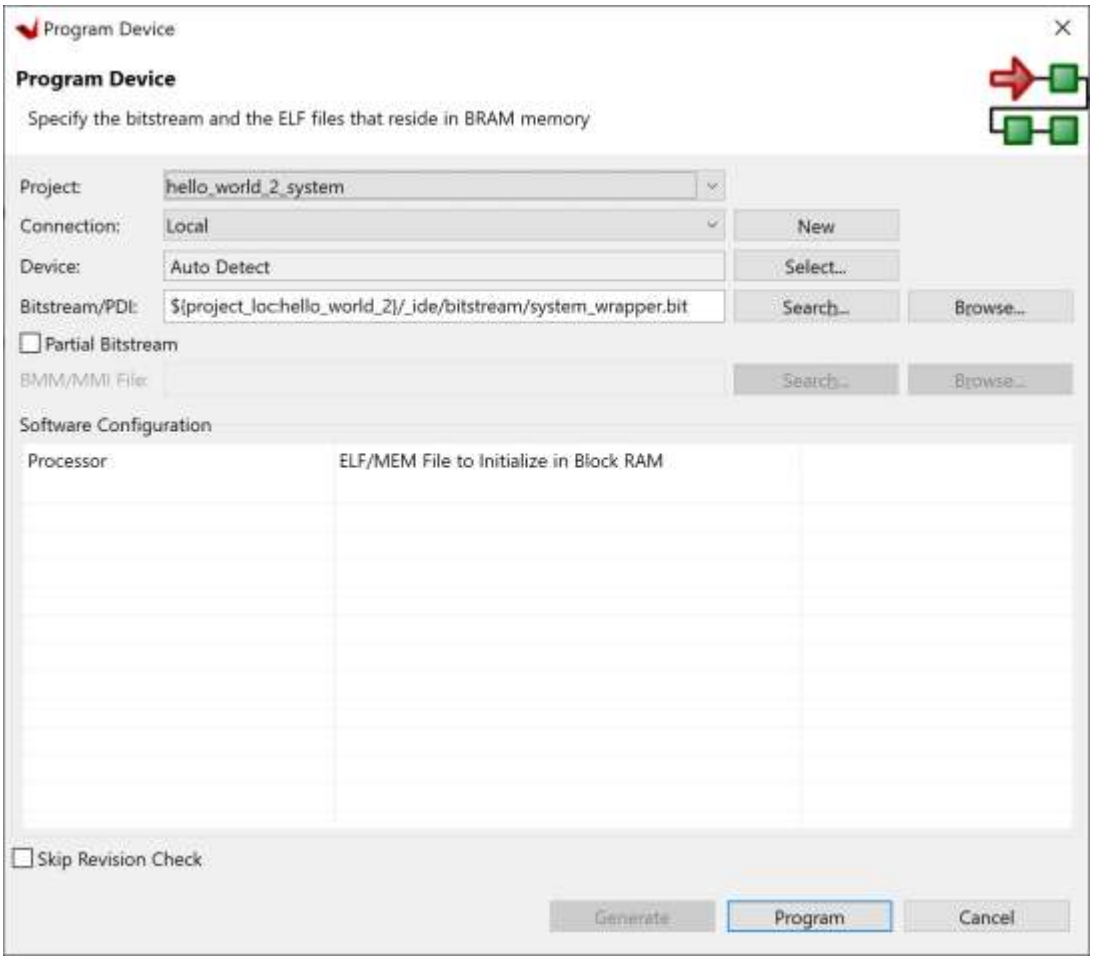
13. In the next window check the order of items in the boot image partition below must be FSBL -> Bitstream File -> App. I don’t remember if I already generated a bootimage. But if you are generating a new one you need to select the files as in the picture. Then press “create image”.



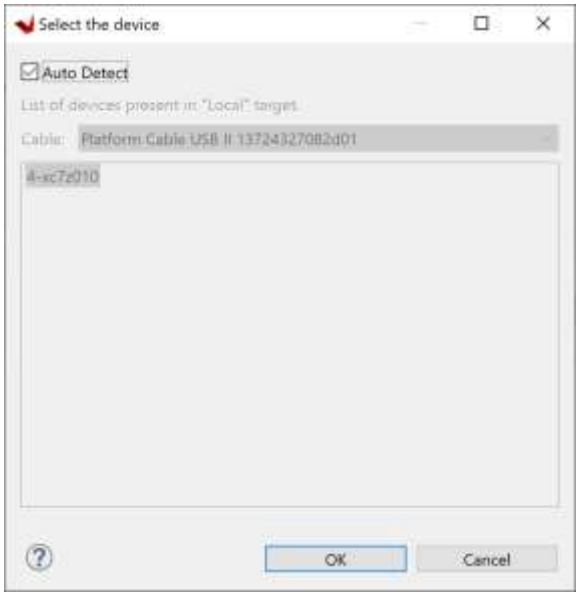
14. Now we need to program the image into the device. Go to Xilinx -> Program device



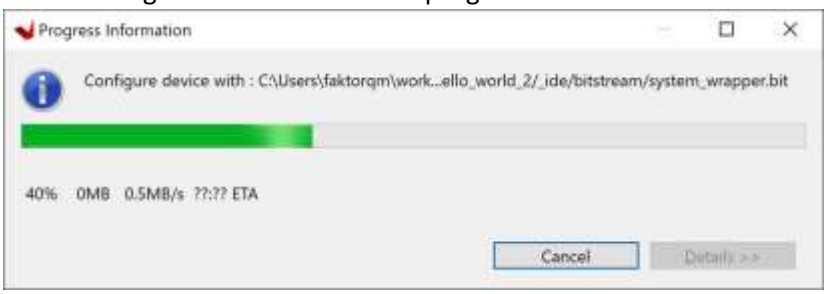
15. We assume that you already have the EBAZ powered up and with the JTAG connected.



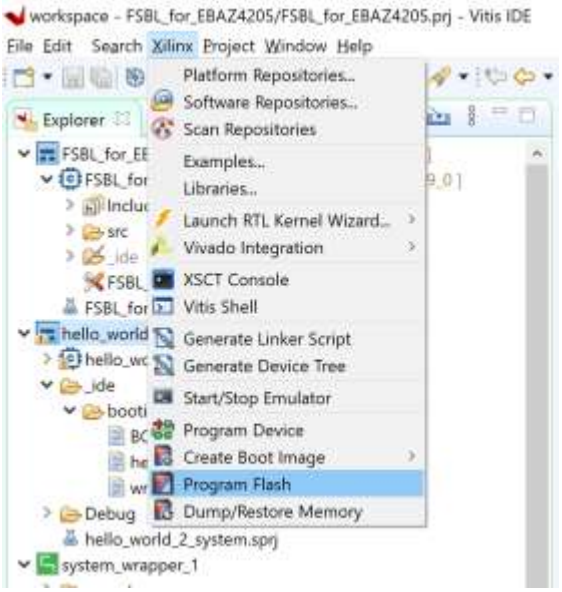
16. On Device, if it is not automatically select, click on “select…” and this window show up. Make sure your device is listed and then click OK.



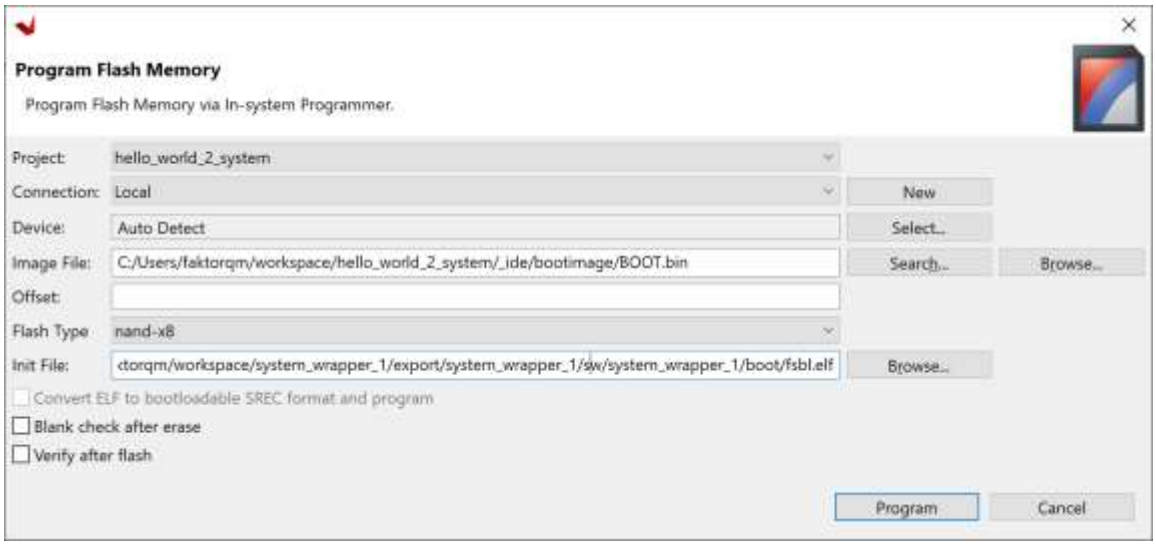
17. Click on Program. You should see a progress bar like this:



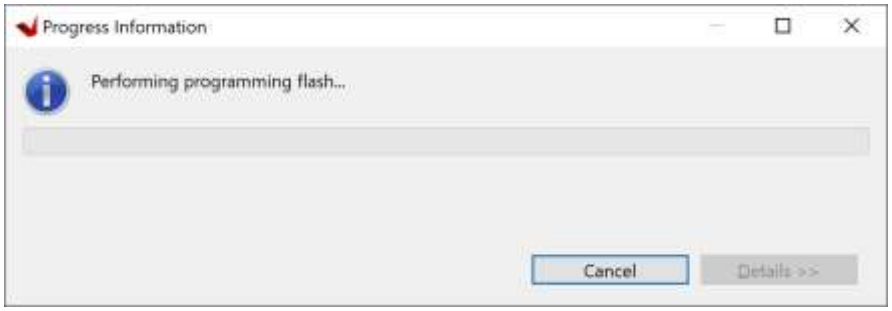
18. Now go to Xilinx -> Program Flash



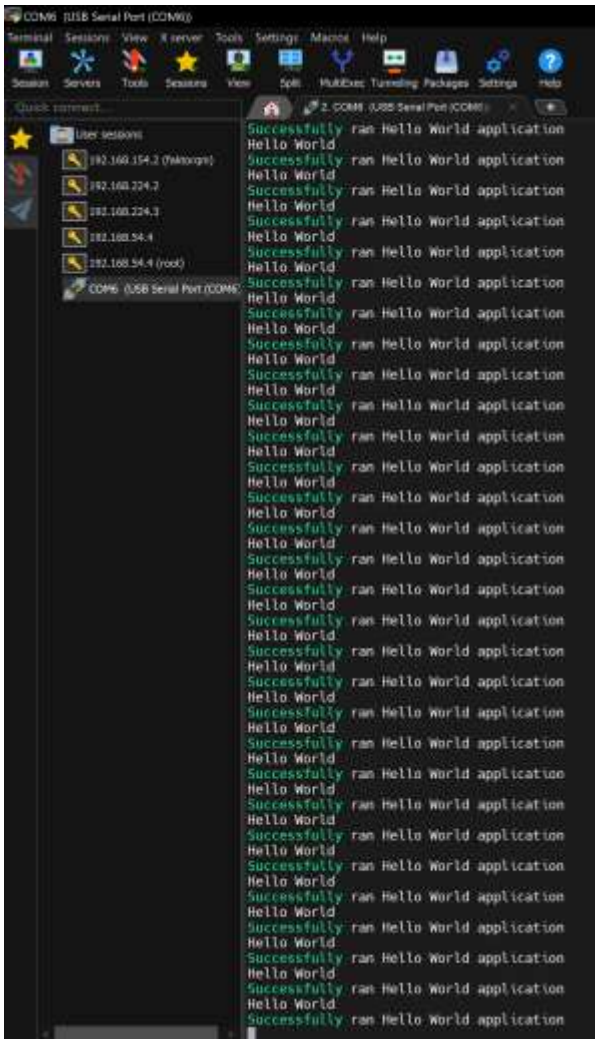
19. In the next window, select nand X8 under the flash type. Then press on program.



20. You should see a window like this with the process and then closes automatically.



21. After the download is complete, power off the board. Change the boot mode from JTAG to NAND. Then power up and see the output from the serial port:



Steps to put this software on the uSD:

1. Format the uSD card in “FAT”.
2. Locate the generated boot.bin and hello_world_system.bif files. (You can find them at picture 19 of the previous steps)
3. Copy both files to the uSD
4. Change the boot mode of the board to uSD.
5. Connect the uSD on the board
6. Power up and enjoy! 😊

Testing peripherals

Text text etx

Ethernet

Text text etx

I/O

Text text etx

I2C

Text text etx

Push buttons

Text text etx

Links & resources

https://theokelo.co.ke/getting-starting-with-ebaz4205-zynq-7000/

https://www.reddit.com/r/FPGA/comments/jvjskn/anyone_have_experience_with_chinese_zynq_boards/

https://www.reddit.com/r/FPGA/comments/kmk9f9/ebaz4205_recycle_a_cheap_cryptominer_part_1/

https://github.com/XyleMora/EBAZ4205

https://github.com/xjtuecho/EBAZ4205/

https://github.com/KeitetsuWorks/EBAZ4205

https://github.com/blkf2016/ebaz4205

https://github.com/Leungfung/ebaz4205_hw

https://github.com/Elrori/EBAZ4205

https://github.com/xjtuecho/EBAZ4205

https://github.com/adafruit/Adafruit-PCF8523-RTC-Breakout-PCB

https://github.com/Xilinx/Vitis_Embedded_Platform_Source

https://github.com/SuperThunder/EBAZ4205_Resources

https://github.com/nightseas/ebit_z7010

https://github.com/ATmega8/EBAZ4205/

https://github.com/guido57/EBAZ4205

https://www.hackster.io/mindaugas2/creating-xilinx-vivado-board-files-for-ebaz4205-a7b120

https://t.me/ebaz4205 [Telegram group]

https://embed-me.com/ebaz4205-recycle-cheap-crypto-miner-part-1/

https://embed-me.com/ebaz4205-recycle-cheap-crypto-miner-part-2/

https://embed-me.com/ebaz4205-recycle-cheap-crypto-miner-part-3/

https://embed-me.com/ebaz4205-recycle-cheap-crypto-miner-part-4/

https://embed-me.com/qemu-how-to-emulate-your-zynq-7000/

https://embed-me.com/qemu-how-to-emulate-your-zynq-7000-part-2/

https://hhuysqt.github.io/zynq1/

https://hhuysqt.github.io/zynq2/

https://hhuysqt.github.io/zynq3/

http://www.hellofpga.com/index.php/2021/07/16/ebaz4205_source/

http://www.hellofpga.com/index.php/category/ebaz4205/

https://cxybb.com/article/Turix/109738107

https://www.eevblog.com/forum/fpga/ebaz4205-(zynq-7000-based-development-board)/

https://www.eevblog.com/forum/fpga/anyone-played-with-these-cheap-($50)-zynq-boards/

https://hackaday.com/2020/12/10/a-xilinx-zynq-linux-fpga-board-for-under-20-the-windfall-of-decommissioned-crypto-mining/

https://hackaday.com/2020/11/18/hacking-the-fpga-control-board-from-a-bitcoin-miner/

https://hackaday.io/project/187351-ebaz4205-fpga-development-environment

https://blog.csdn.net/oHuYaWen/article/details/104886201

https://blog.csdn.net/savantor/article/details/120177180

https://blog.csdn.net/leo_xu_/article/details/105589314

https://blog.csdn.net/Zhu_Zhu_2009/article/details/103890321

https://blog.csdn.net/Markus_xu/article/details/107752300

https://blog.csdn.net/weixin_42741023/article/details/103336872

https://blog.csdn.net/z951573431/article/details/89739165

https://blog.csdn.net/u011413001/article/details/105348256

https://blog.csdn.net/oHuYaWen/article/details/104886201

https://blog.csdn.net/weixin_42157664/article/details/109366044

https://blog.csdn.net/weixin_42741023/article/details/103335948

https://blog.csdn.net/weixin_42741023/article/details/103336872

https://blog.csdn.net/Turix/article/details/109738107

https://blog.csdn.net/Turix/article/details/109748220

https://blog.csdn.net/Turix/article/details/109773258

https://blog.csdn.net/yihuajack/article/details/120714268

https://www.programmersought.com/article/29027857811/

https://www.programmersought.com/article/63646026059/

https://www.programmersought.com/article/78977621396/

https://www.programmersought.com/article/63123379082/

https://www.programmersought.com/article/26537901479/

https://www.programmersought.com/article/78977621396/

https://www.programmersought.com/article/20087623050/

https://www.programmersought.com/article/73739556763/

https://www.programmersought.com/article/16037626664/

https://www.programmersought.com/article/68374821034/

https://www.programmersought.com/article/39214086399/

https://chowdera.com/2022/03/202203040647107860.html

https://webuiltawallwebuiltthepyramids.blogspot.com/2021/01/yet-another-ebaz4205-writeup-1-crystal.html

https://webuiltawallwebuiltthepyramids.blogspot.com/2021/01/ebaz4205-petalinux-installation.html

https://webuiltawallwebuiltthepyramids.blogspot.com/2021/01/yet-another-ebaz4205-writeup-3-pynq.html

https://webuiltawallwebuiltthepyramids.blogspot.com/2021/03/yet-another-ebaz4205-writeup-4-pynq.html

https://blog.katastros.com/a?ID=01750-da5cc7e2-0c69-4186-a5c8-536d2ad4fefe

https://www.nxp.com/products/peripherals-and-logic/signal-chain/real-time-clocks/rtcs-with-ic-bus/100-na-real-time-clock-calendar-with-battery-backup:PCF8523

https://learn.adafruit.com/adafruit-pcf8523-real-time-clock/downloads

http://www.360doc.com/content/13/1113/20/8744436_328989607.shtml

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841996/Linux

https://whycan.com/t_2297.html

https://www.jianshu.com/p/b83c663ecaaa

https://www.jianshu.com/p/370f95f0068f

https://www.cnblogs.com/ifpga/p/10778040.html

https://www.cnblogs.com/beihaixingchen/p/13920868.html

https://its201.com/article/liboxiu/99569799

https://russianblogs.com/article/2163592017/

https://www.cnblogs.com/beihaixingchen/p/13920868.html

https://docs.xilinx.com/v/u/2021.2-English/ug940-vivado-tutorial-embedded-design

https://docs.xilinx.com/r/en-US/ug895-vivado-system-level-design-entry/

https://xilinx.github.io/Embedded-Design-Tutorials/docs/2021.2/build/html/docs/Introduction/Zynq7000-EDT/2-using-zynq.html

https://digilent.com/reference/programmable-logic/guides/getting-started-with-vivado

https://weble.upc.edu/asig/ESDC0/XAPP1026/zc702_GigE/SW/design_1_wrapper_hw_platform_0/ps7_init.html

https://support.xilinx.com/s/question/0D52E00006IJrM7SAK/is-it-possible-to-use-petalinux-for-the-ebaz-4205-board-xc7z7010clg4001?language=en_US

https://programmerclick.com/article/5851348415/

https://gist.github.com/CodeAsm/57d5856370546adbc41941ca716988f8

Projects:

[Getting started with Zynqberry]

https://forum.digikey.com/t/getting-started-with-the-zynqberry/13327

[Debian: Getting Started with the Zynq-7000]

https://forum.digikey.com/t/debian-getting-started-with-the-zynq-7000/14380

http://www.hellofpga.com/index.php/2022/03/08/ps_lcd_io_spi/

http://www.hellofpga.com/index.php/2022/03/13/tf_card_boot/

http://www.hellofpga.com/index.php/2021/07/21/first/

https://www.programmersought.com/article/14214479812/

https://www.programmersought.com/article/62434136328/

[Uboot changes]

https://tuxengineering.com/blog/2020/09/03/U-Boot-changes-to-distro-boot.html

[LVDS driver]

https://blog.csdn.net/memoff/article/details/106331579

[Minimal breakout board]

https://git.sr.ht/~ajk/ebaz4205_minimal_breakout/tree/master

https://github.com/dave18/EBAZ4205-Daughter-Board

[Ubuntu 16.04 for EBAZ4205]

https://renjikai.com/ebaz4205-sd-ubuntu18/ (Explanation how to flash)

https://github.com/bjrjk/EBAZ4205 (Vivado project)

[Getting started with PetaLinux]

https://matthewtran.dev/2021/08/getting-started-with-petalinux/

[PetaLinux]

https://www.reddit.com/r/FPGA/comments/obgh7w/prebuilt_petalinux_for_ebaz4205_zynq_7010_board/

https://www.fpgadeveloper.com/how-to-install-petalinux-2019.1/

[SDR Radio with EBAZ4205]

https://hackaday.io/project/186329-30mhz-spectrum-and-sdr-in-a-fpga/details

[Other boards with same ZYNQ]

https://github.com/KarolNi/S9miner_sample

https://github.com/ChinaQMTECH/ZYNQ_BAJIE_BOARD/tree/master/XC7Z010

https://github.com/wuxx/Colorlight-FPGA-Projects#colorlight-i5-v70

[Video]

https://space.bilibili.com/208826118

[Examples]

https://qiita.com/kan573

https://www.hackster.io/xilinx/products/vivado-design-suite-hlx-editions?ref=project-a7b120

[VHDL Learn by example]

http://esd.cs.ucr.edu/labs/tutorial/

[FPGAs]

https://github.com/Hermann-SW/FPGAs

[Zynq PS register summary]

https://weble.upc.edu/asig/ESDC0/XAPP1026/zc702_GigE/SW/design_1_wrapper_hw_platform_0/ps7_init.html