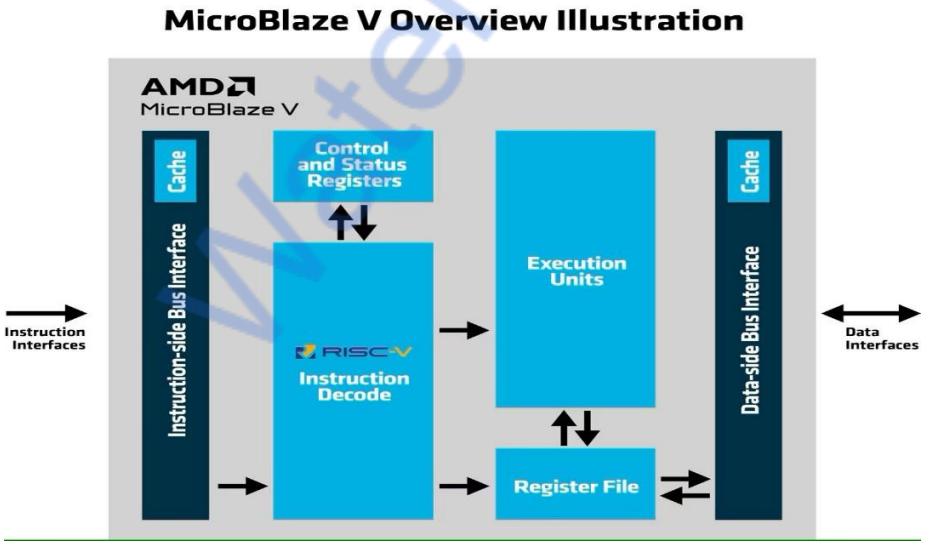
Introduction to MicroBlaze V – Xilinx Soft Processor

1. What is MicroBlaze V?

MicroBlaze V is a softcore RISC-V processor developed by Xilinx (now part of AMD), designed to run inside an FPGA. It is fully customizable and allows you to tailor features based on your project requirements.



Key Features of MicroBlaze V:

- Based on the RISC-V Instruction Set Architecture (ISA) (open and industry-standard)
- Designed for embedded control applications inside Xilinx FPGAs
- Integrated with the Vivado Design Suite and Vitis for software development
- Scalable performance: You can configure pipelines, cache, MMU, etc.

2. MicroBlaze V vs Nios II – Quick Comparison

Feature	MicroBlaze V (Xilinx)	Nios II (Intel/Altera)
Architecture	RISC-V ISA	Proprietary Nios II ISA
Development tools	Vivado + Vitis	Quartus + Nios II SBT
Bus Interface	AXI (Advanced eXtensible Interface)	Avalon
Ecosystem	Open-source RISC-V tools available	Proprietary
Performance	Highly configurable	Also configurable, but limited to Nios II core options
Debugging	Xilinx SDK / Vitis Debugger	Nios II Eclipse-based debugger

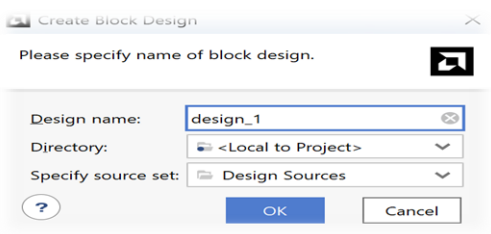
3. Typical Use Cases for MicroBlaze V

- Embedded control inside an FPGA design
- Communication protocol handling
- Simple real-time processing
- Peripheral management (GPIO, UART, SPI, etc.)
- Running lightweight operating systems (optional)

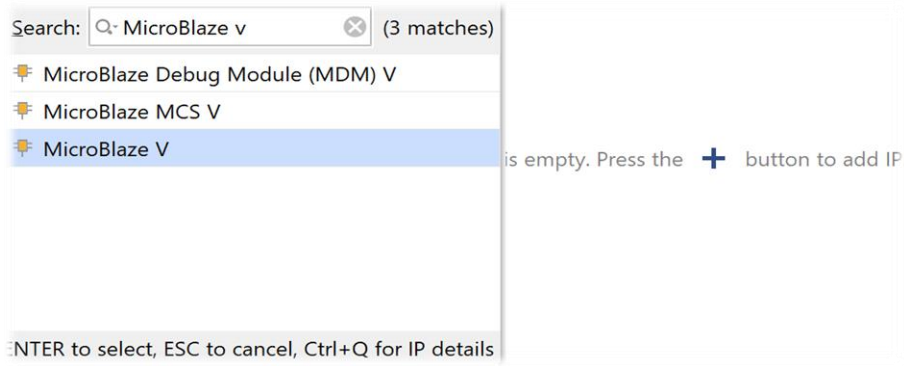
4. Workflow Overview (High Level)

1. **Configure MicroBlaze V in Vivado IP Integrator**
   - Open Vivado Design Suite and **Create** a new project.
   - Choose the correct **FPGA part** (Arty S7-50)
   - Click on **Create Block Design** under **IP Integrator**

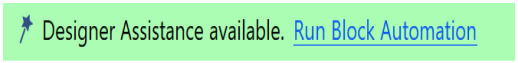
   - Change the design name if you want and click on OK



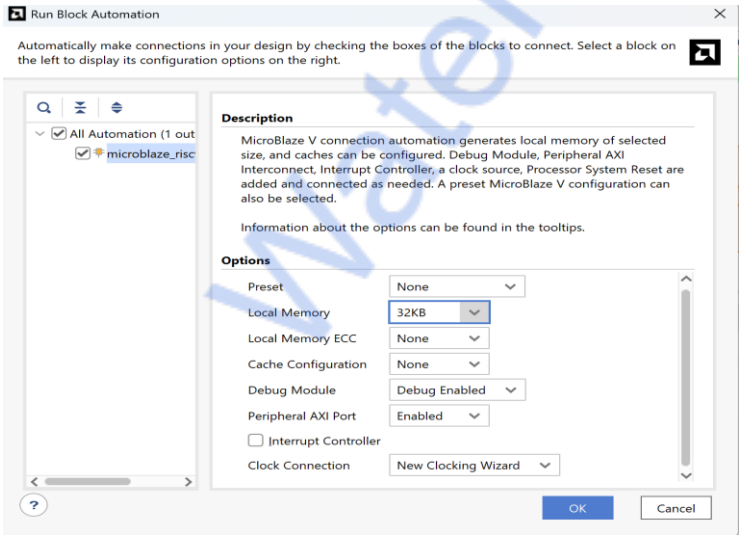
- Add **MicroBlaze V processor** Click **"Add IP" → Search: MicroBlaze V"**



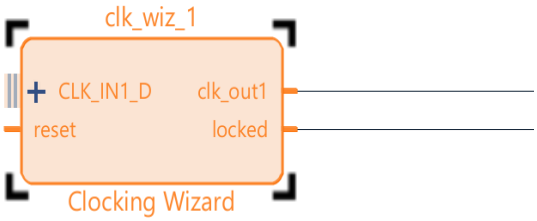
- Run **"Block Automation"**



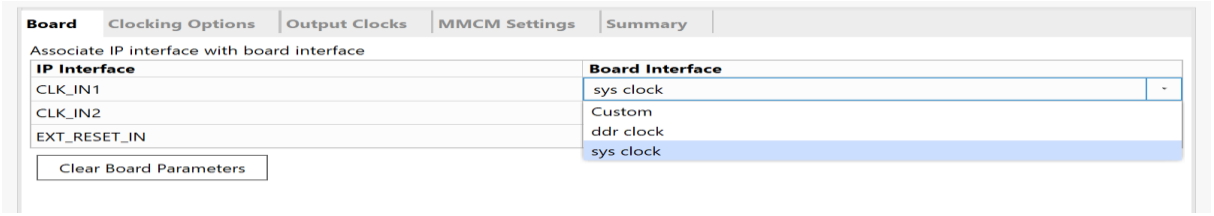
- Customize MicroBlaze V (optional): Configure cache, debug options, memory, and performance settings. For now I will only change the **Local Memory** from **16** to **32 KB**.



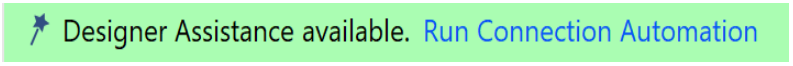
- Now dubbel click on **clk_wiz_1** to customize the system clock.



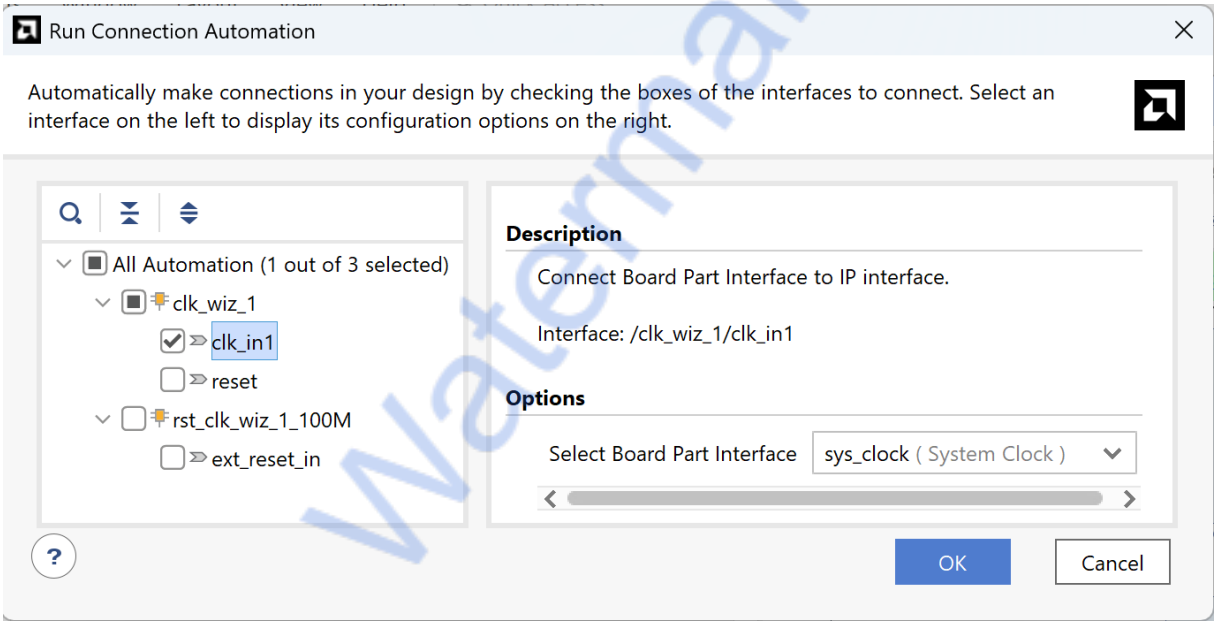
- Click on Board Interface and choose **sys clock** to **CLK_IN1**.



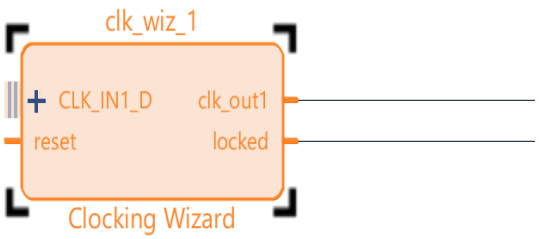
- Run **"Connection Automation".**



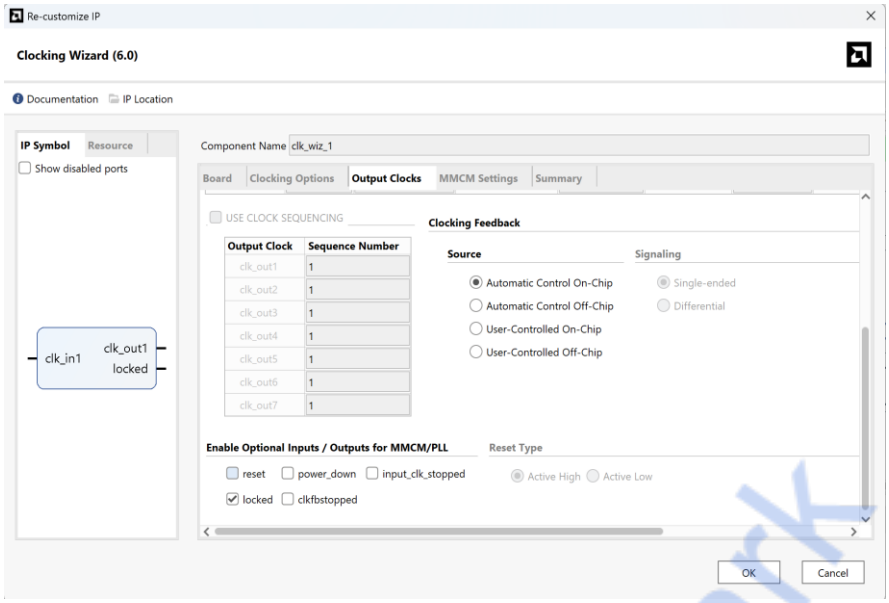
- Check the box for **clk_wiz_1** and click **OK**



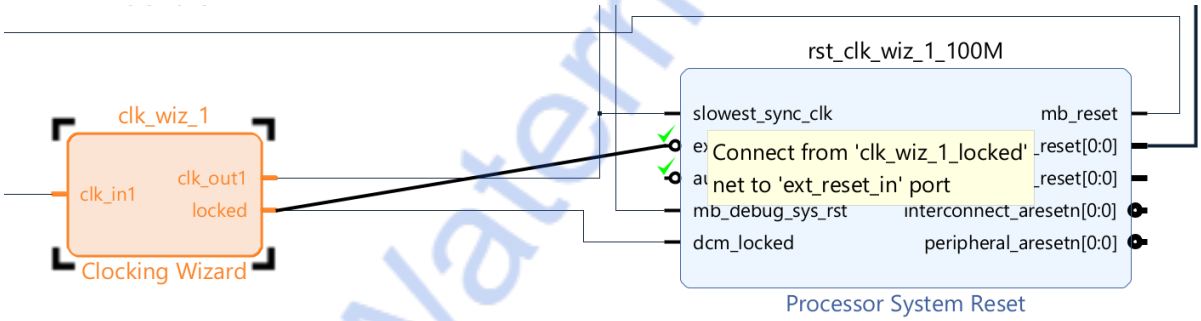
- Agine dubbel click on **clk_wiz_1**



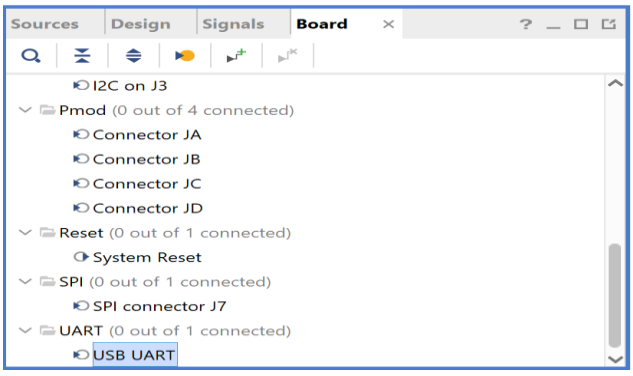
- Under **Outputs Clocks** check off the **reset** box and click on **OK**



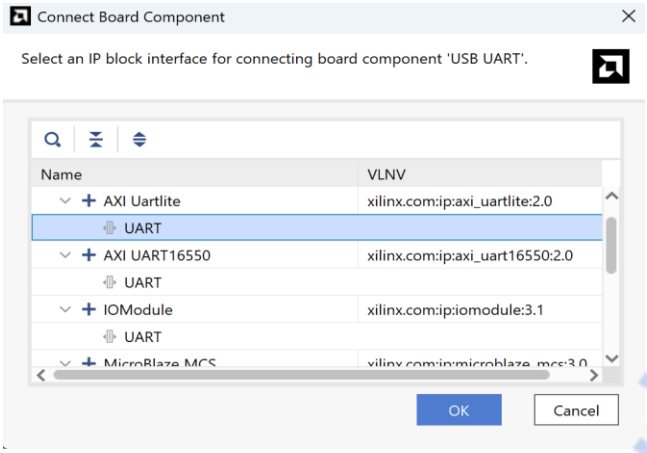
- Press the **LOCKED** pin and drag to **ext_reset_in** to connect them.



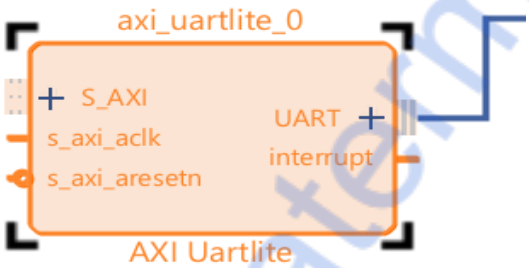
- Now we will add the **UART.** Click on **Board** to se all peripherals on your board.



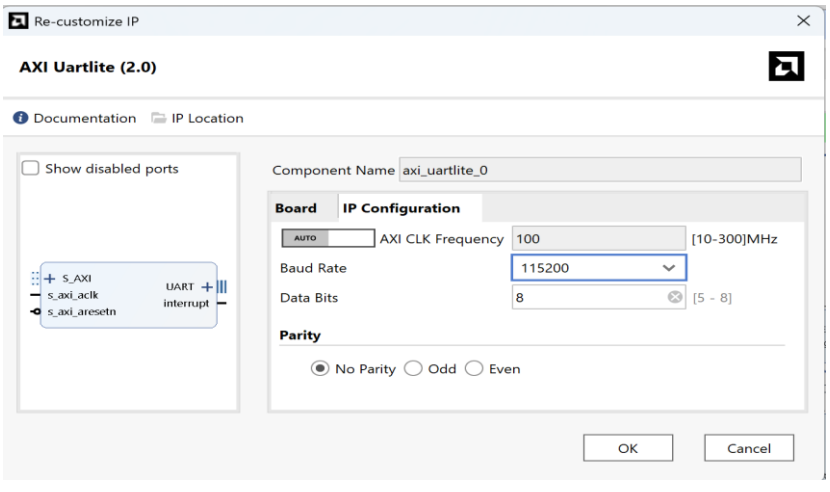
- Scroll down and select **USB UART** → Dubbel Click on it. Choose **UART** and click on **OK**



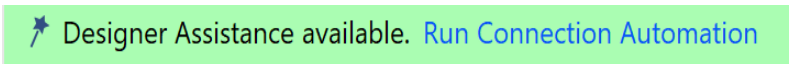
- Now dubbel click on **axi_uartlite_0** to change the **AXI CLK frequency.**



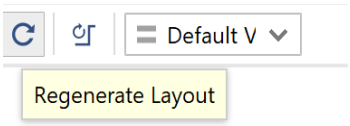
- Click on **IP Configuration** and change the **Baud Rate** to (**115200**) if you want to use external terminal programs such as **Tera Term** and click on **OK**.



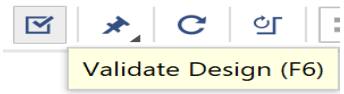
- Run **"Connection Automation".**



- Regenerate the layout

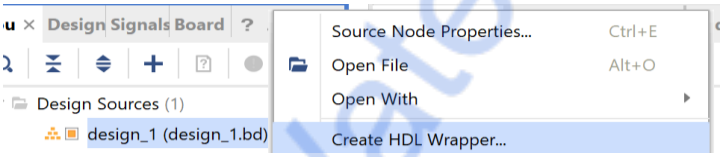


- Validate the design to make sure that all connections are ok

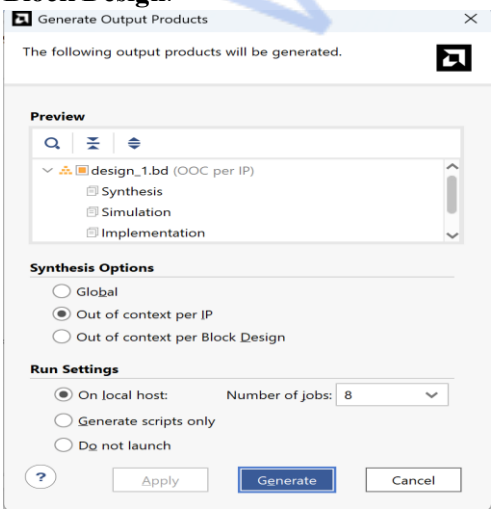


2**. Generate HDL Warpper and Block Design**

- Now we are going to create a **HDL Wrapper** and save it. Right click on the design block and select Create **HDL Wrapper** and let Vivado manage wrapper.



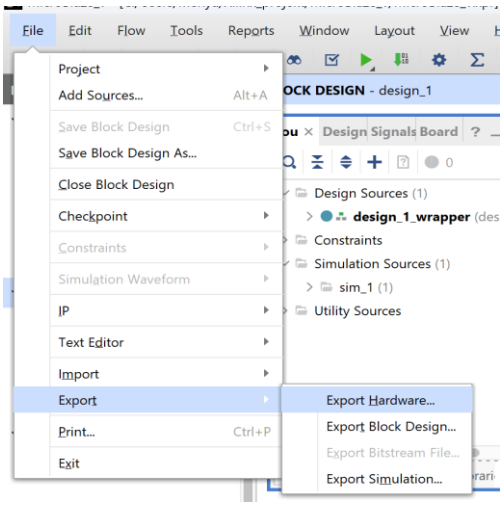
Next step will be generating a block design. under **IP INTEGRATOR** Click on **Generate Block Design**.



Click on **Generate**.

3. **Export the hardware file (xsa)**

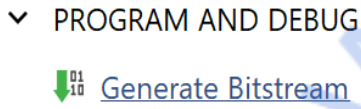
- Go to **"File → Export → Export Hardware"**



- Check **"Pre-synthesis" Next** and **Finish.**

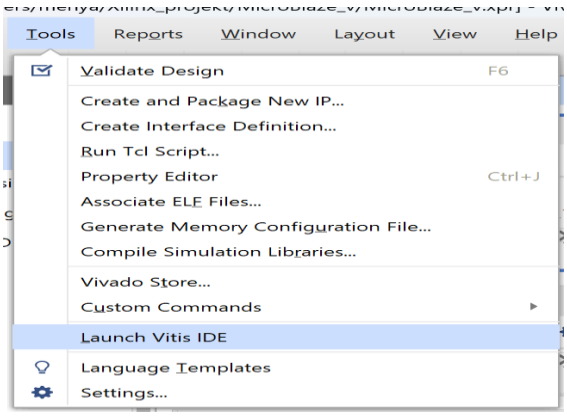
4. **Run the Vivado flow:**

- **Synthesis**
- **Implementation**
- **Generate Bitstream**

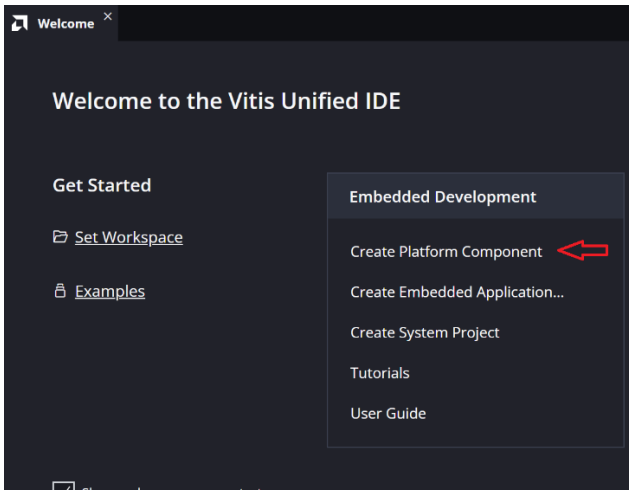


5. **Develop software in Vitis IDE**

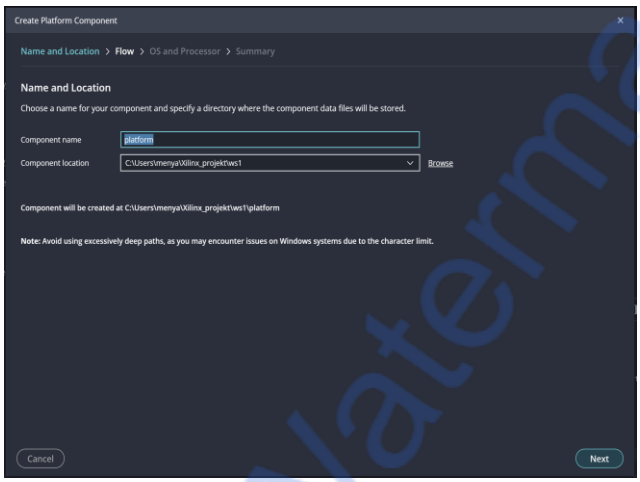
- Launch Vitis directly from Vivado**:** Click **"Tools → Launch Vitis"**



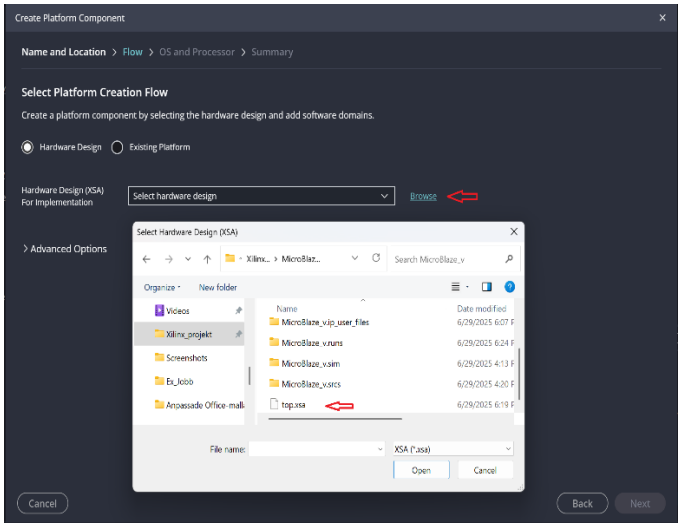
- Next you need to create a **platform component**



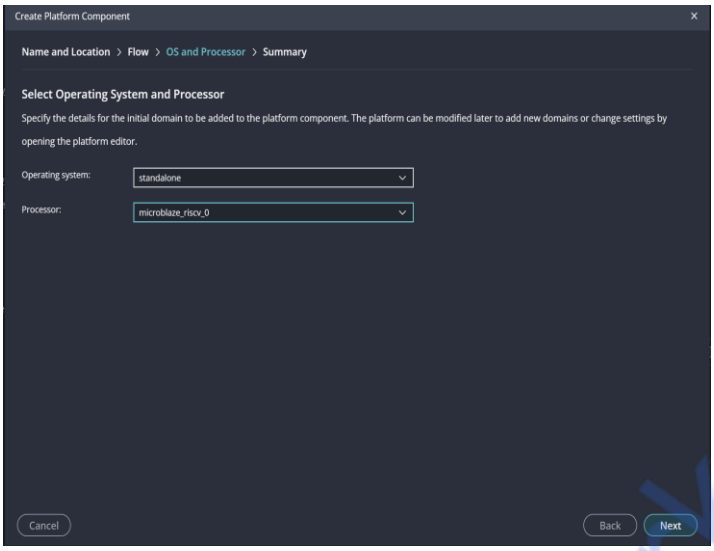
- Change the name of the component if you want and click on **Next**



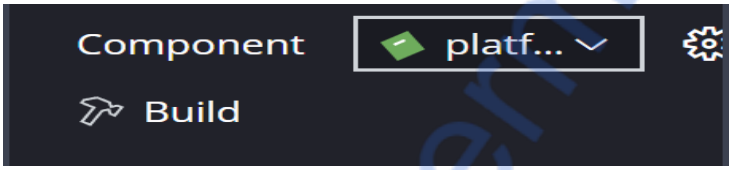
- In the next step we need to attach the **xsa** file located in the project folder and **Next**



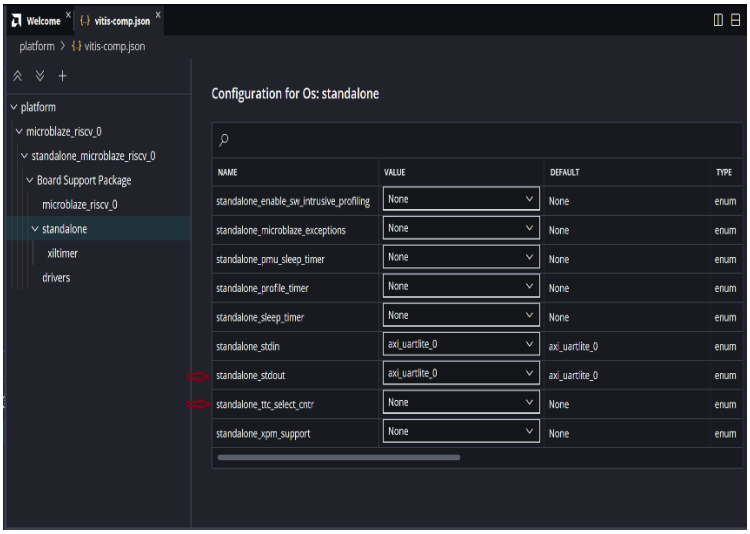
- No need to change anything on the next step, **Next** and **Finish**



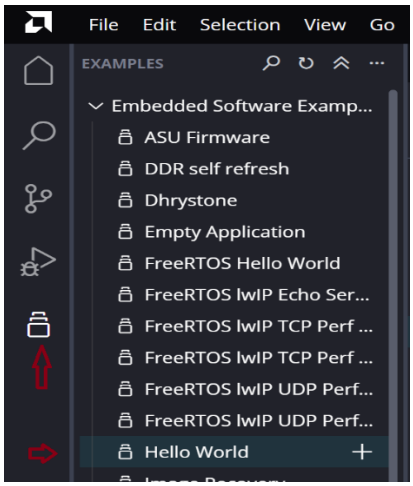
- Now the platform needs to be built, go ahead and click on **Build**



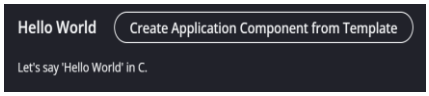
- Now after we build it, we can go to the support package and here we can see that we are using the **AXI UART LITE**



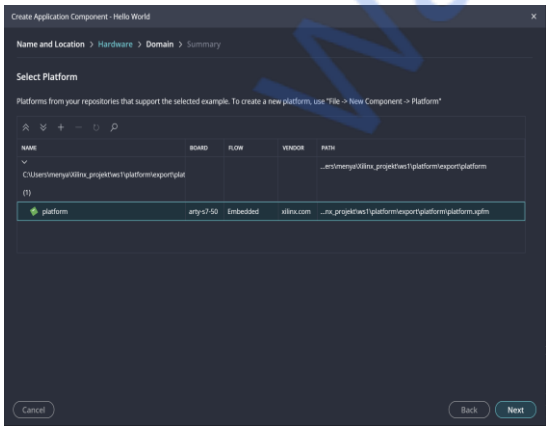
- In the vertical list on the left, you will find examples. I'm choosing **Hello World** template for now.



- Select the template you want and Click on **Create Application from Template**, and **Next**.



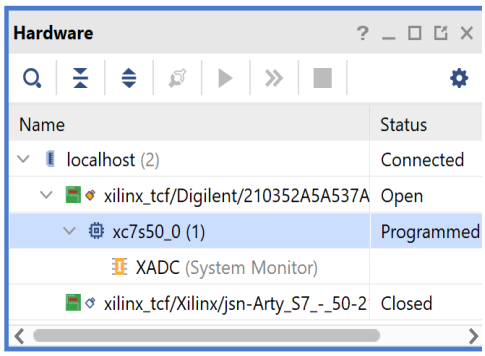
- Click on the **platform** that you create it for this project and **Next → Next → Finish**



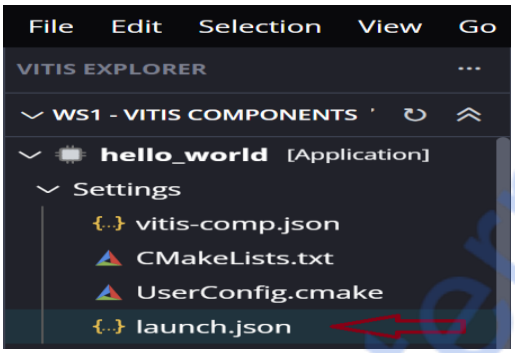
Now you can go ahead and build again by clicking on **Build**

5. Program FPGA and run your application

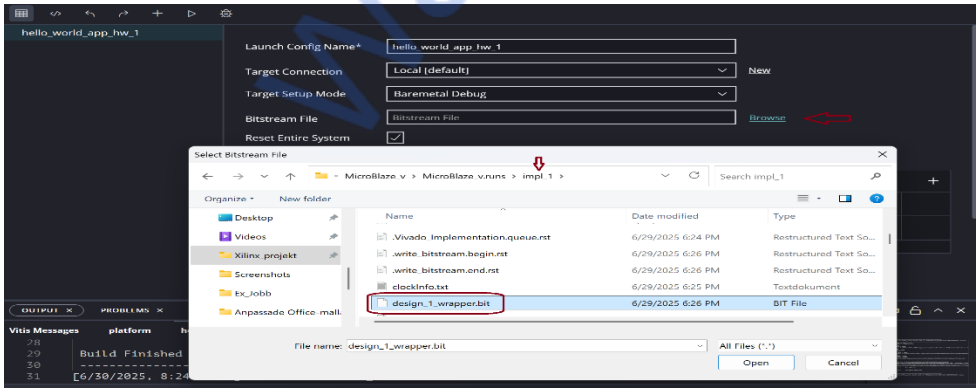
- Go back to Vivado and click on Hardware Manger and connect your FPGA-board



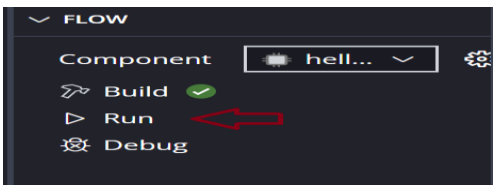
- Now we need to attach the bitstream file and to do this click on **launch.json** under **settings** in the flow navigator of the hello_world



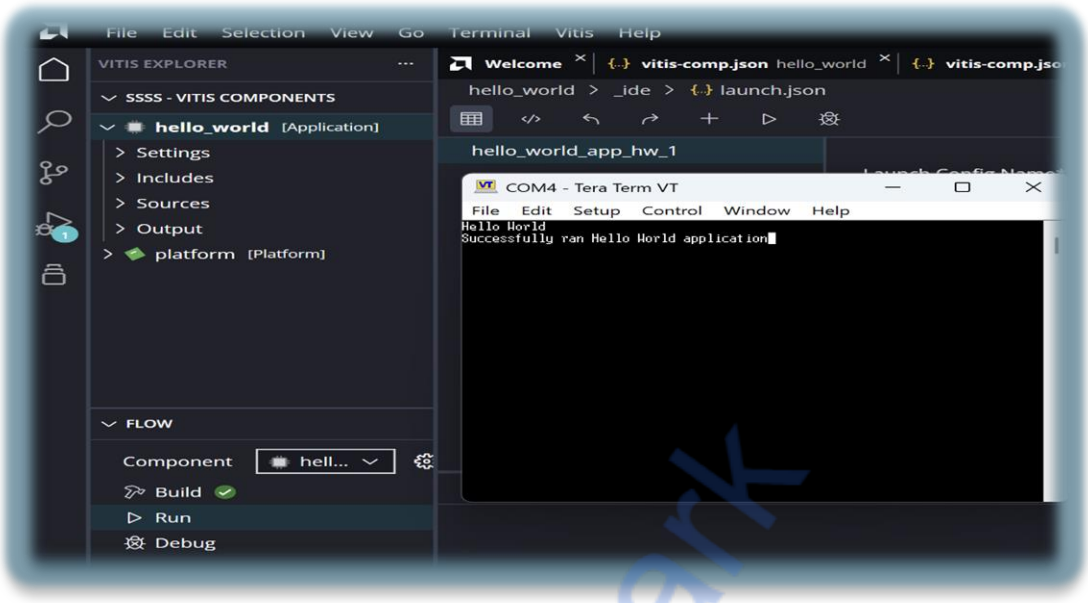
- Then click on **Browse** to select **bit** file, you will find it in the **impl_1** folder.



- Now you are ready to run it, so go ahead and click on **Run**



Congratulations! you have built your first program with MicroBlzae



Demo

https://www.youtube.com/watch?v=RFfxE_HxrOs

Useful Resources

- MicroBlaze V Product Page (Xilinx)
  https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/microblaze.html
- RISC-V Instruction Set Overview
  https://riscv.org/about/
- Vitis Unified Software Platform
  https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/vitis.html