Control AXI GPIO IP - I/Os with software on MicroBlaze

In this tutorial we are going to add a AXI GPIO IP to the Blockdesign and connect two external LEDs to the AXI GPIO IP Core. After generating the Bitstream we export the design to Vitis and control the LEDs with software(C/C++) running on the MicroBlaze. In the same way you can connect any other I /Os external or internal the Chip.
This tutorial is based on the the reference design of the TE0717 Board and Vitis 2021.2 was used.

1. First open the prebuilt reference design with the "_create_win_setup.cmd" script and open the block design.
2. Add a second "AXI GPIO IP" from the IP catalog to the block design.

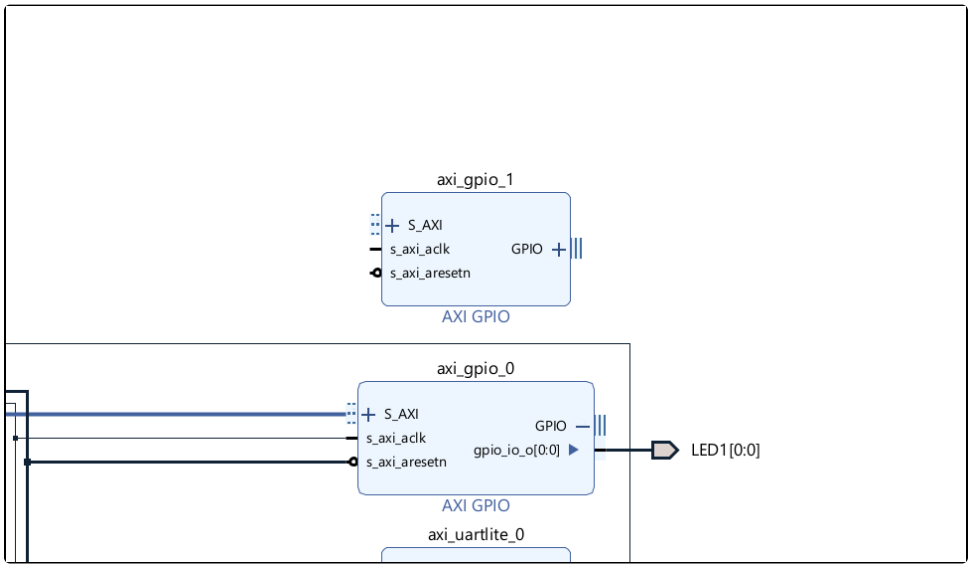

Figure 1: Adding AXI GPIO IP Core

3. Configure the IP by double-clicking on the IP. Make the changes and click "ok".

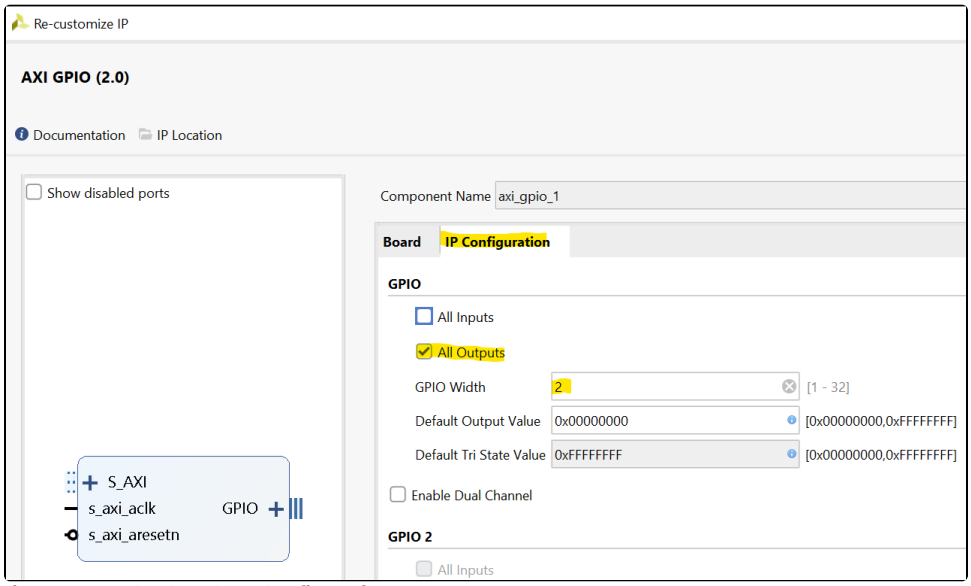

Figure 2: AXI GPIO IP Core Configuration

4. Right click on the port and choose "Make external" in the context menu

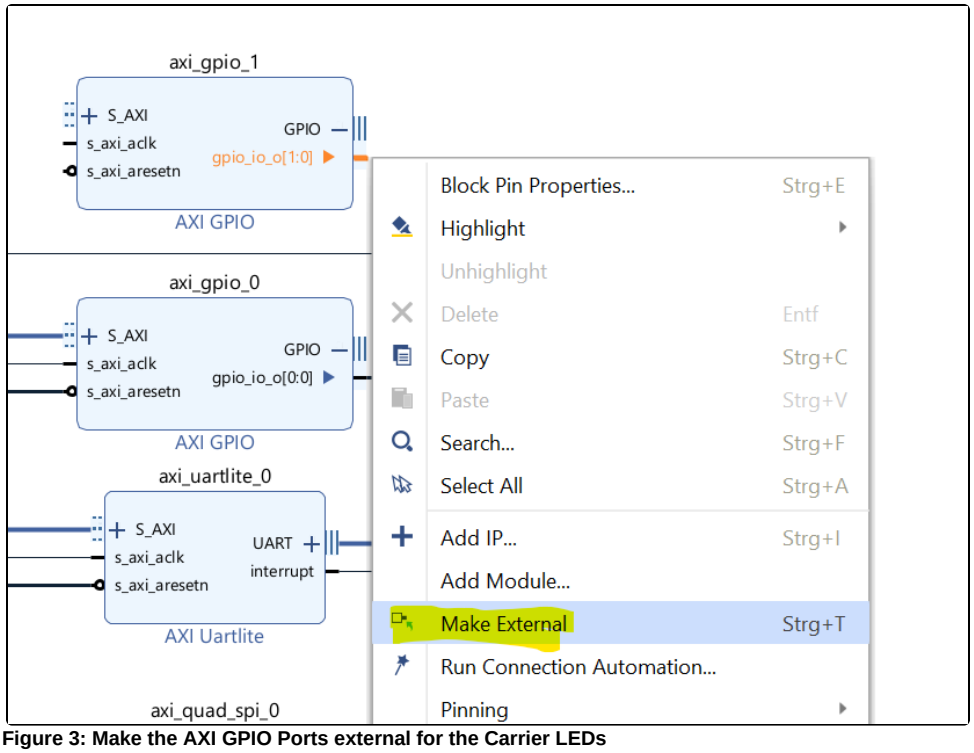

Figure 3: Make the AXI GPIO Ports external for the Carrier LEDs

5. Run the connection automation. The AXI GPIO IP Core gets connected to the MicroBlaze via the AXI Interface.

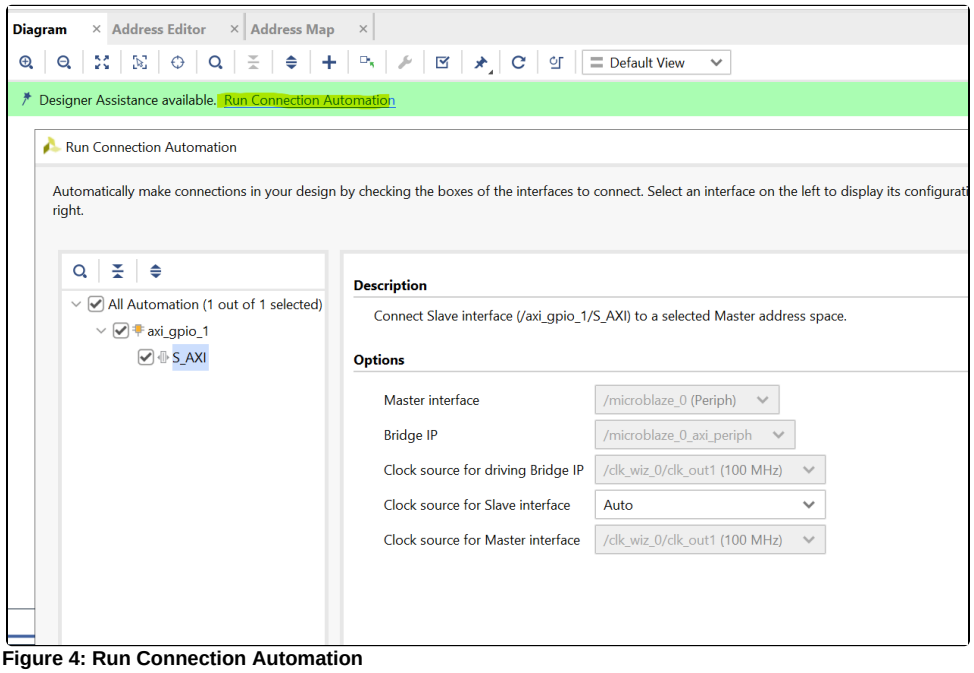

Figure 4: Run Connection Automation

6. Now the address mapping in the address editor should be set like:

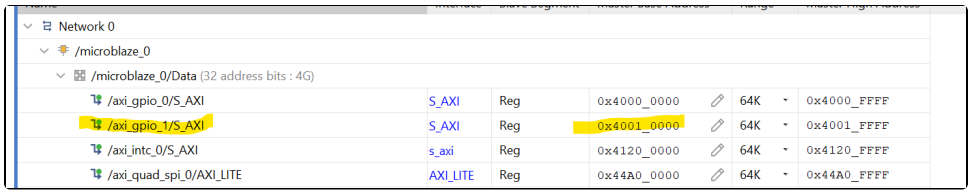

Figure 5: Address Editor - AXI GPIO 1

7. Next, the constraints for the two leds have to be set in the constraint file.

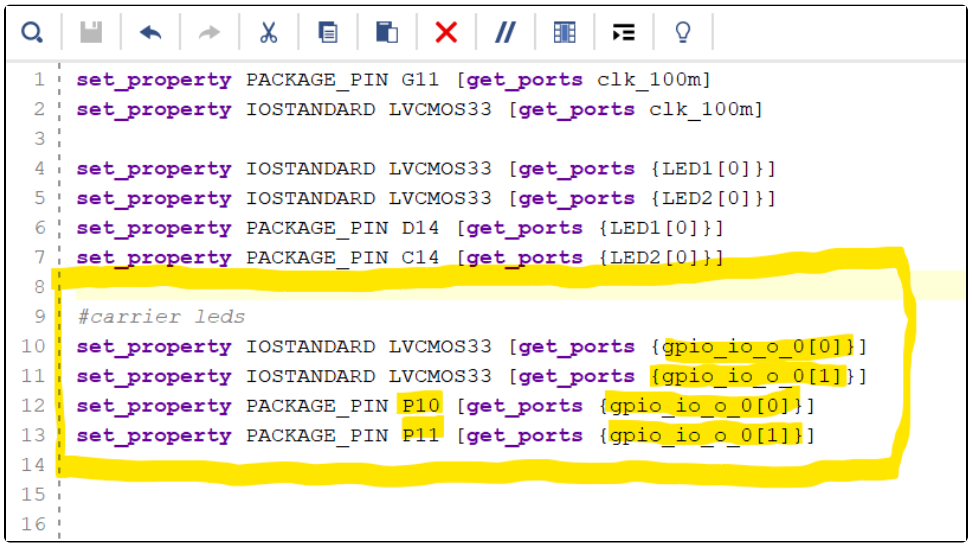

Figure 6: LEDs constraints

8. Now you are all set in Vivado and you can build the bitstream + export the project to Vitis with the following command in the TCL-console(only possible in the script-based reference design from trenz).
   Manual steps would be to Generate the Bitstream and export the hardware platform.

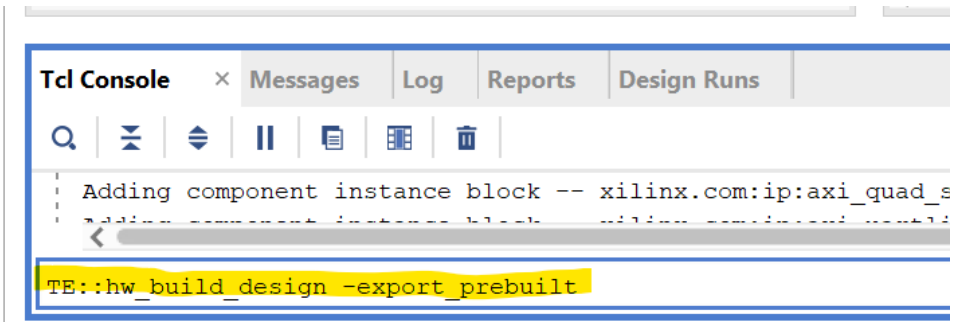

Figure 7: Build bitstream and export project

9. After that was successful you can test the functionality of the VIO Core by programming the FPGA and controlling the onBoard LED2.
   Otherwise continue with Step 10.

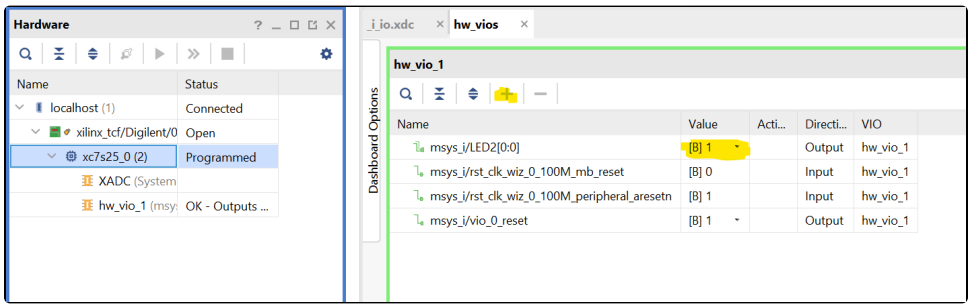

Figure 8: Program FPGA and control onboard LED through the VIO IP Core.

10. Now you can build the Vitis project with the provided C/C++-Applications with the following command. Manual steps would be to Open Vitis(Tools Launch Vitis IDE) and import the project from the project directory.

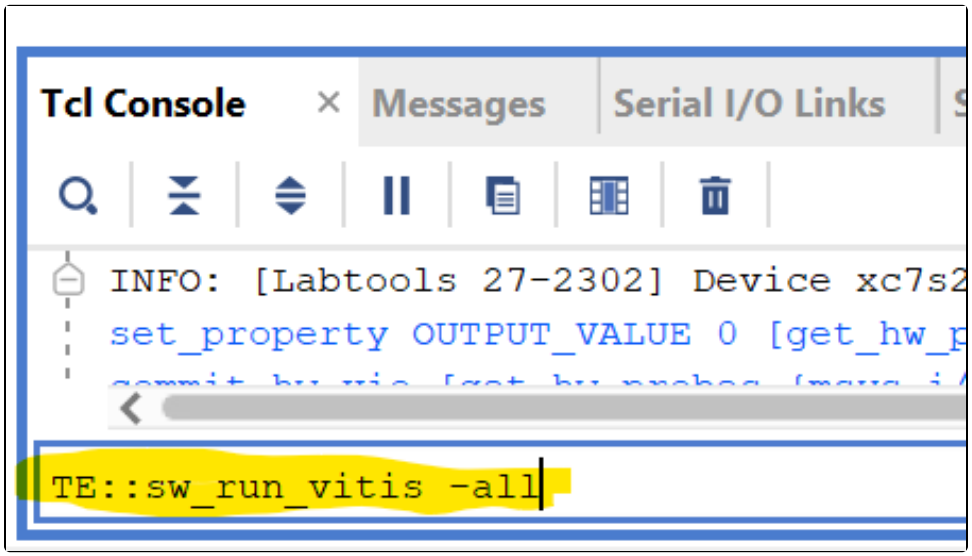

Figure 9: Build Vitis project with built vivado design

11. In Vitis open the hello_te0717 application:

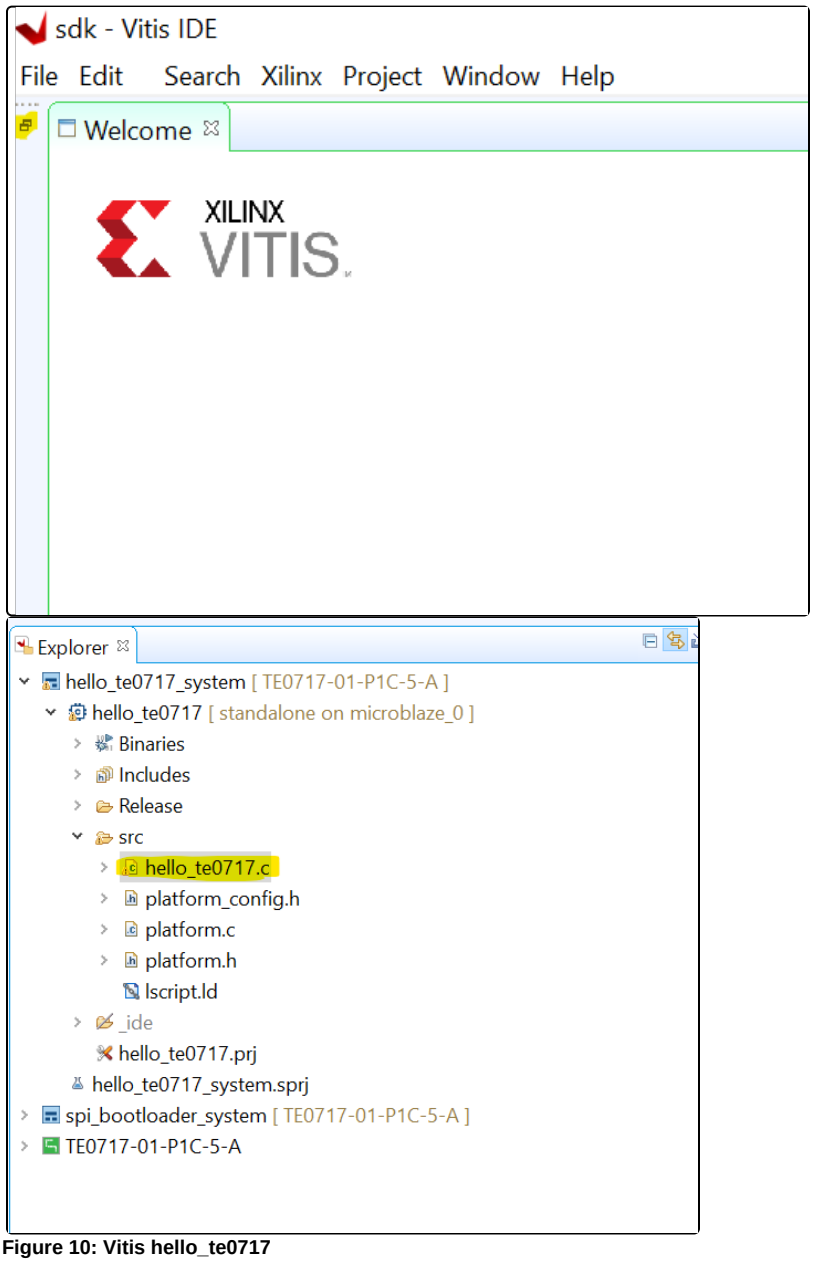

Figure 10: Vitis hello_te0717

12. Add the following lines of code to also make both of the LEDs blink:

```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include <xil_io.h>
#include "xparameters.h"

int main()
{
    init_platform();
    int print_index = 0;

    xil_printf("\n\rHello Trenz Module TE0717, Program stops after 10 minutes due to  HyperRAM IP time limit\n\r");

    while(1){

      print_index++;
      xil_printf("Hello Trenz Module TE0717 (Loop: %i)\n\r", print_index);

        Xil_Out32(XPAR_AXI_GPIO_0_BASEADDR, 0xF); // turn LED on
        sleep(1);
        Xil_Out32(XPAR_AXI_GPIO_0_BASEADDR, 0x0); // turn LED on
        sleep(1);

        // Carrier LEDs
        Xil_Out32(XPAR_AXI_GPIO_1_BASEADDR, 0xF); // turn LED on
        sleep(1);
        Xil_Out32(XPAR_AXI_GPIO_1_BASEADDR, 0x0); // turn LED on
        sleep(1);

    }

    cleanup_platform();
    return 0;
}
```

Figure 11: hello_te0717 adding code for controlling carrier LEDs

13. Save and Build the project
14. Right-Click on the application and choose "Run Configuration"

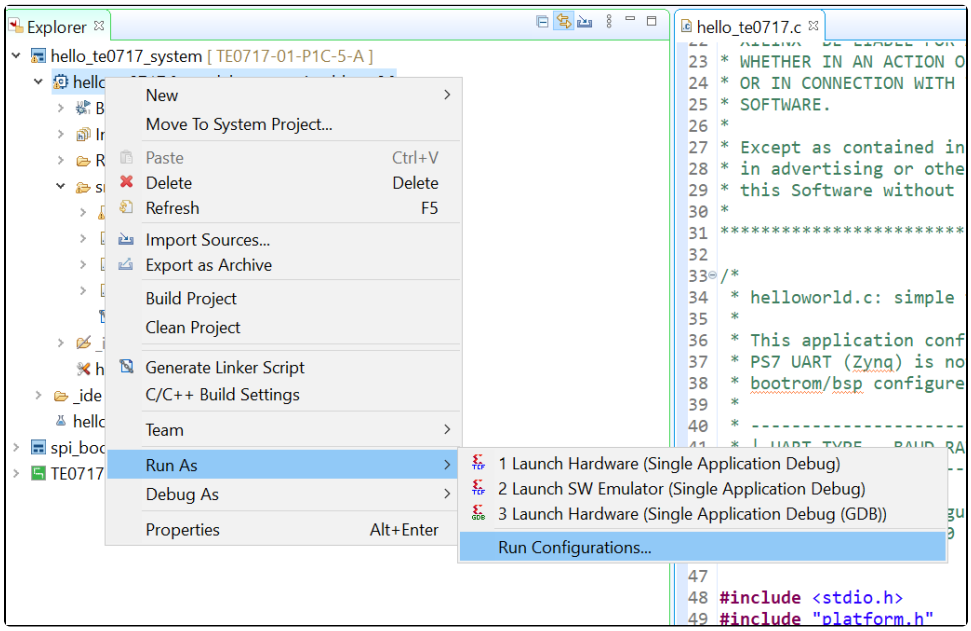

Figure 12: Run Configuration

15. Since we already programmed the FPGA, uncheck the following:

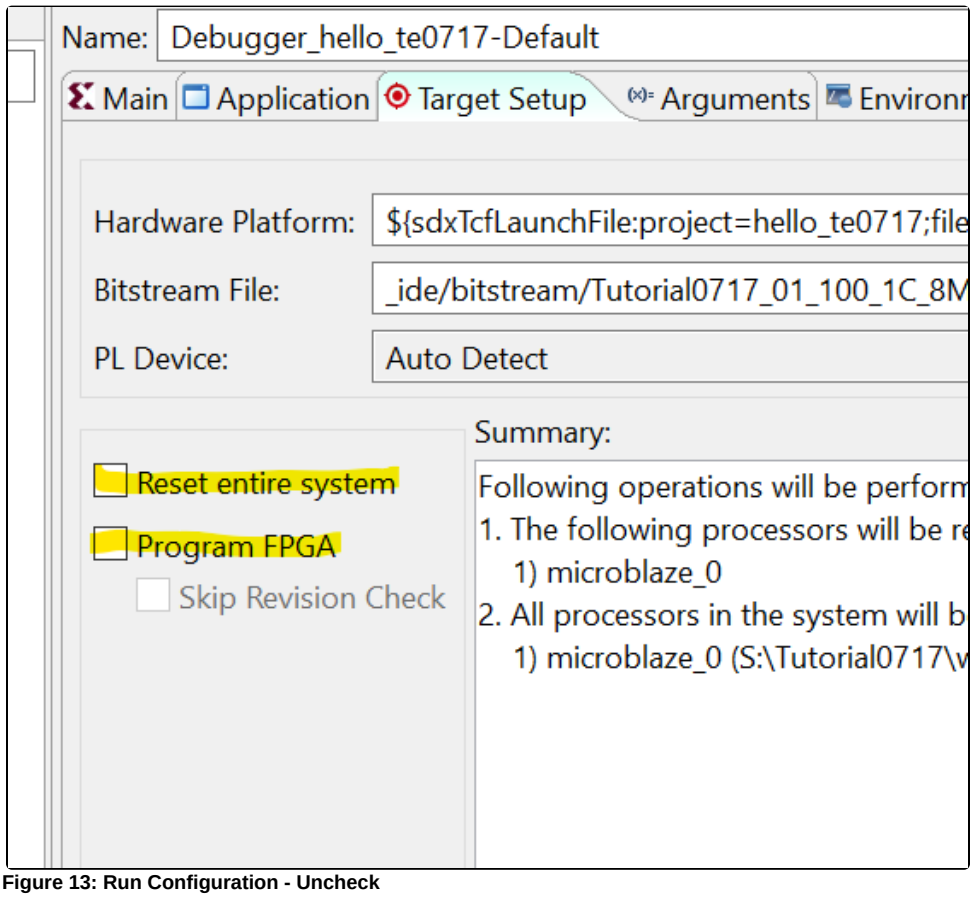

Figure 13: Run Configuration - Uncheck

16. Click on Apply and Run to execute the program on the MicroBlaze. You Should now see the LEDs of the module and carrier blinking in sequence. Also if you open up a serial connection to the board with a program like PuTTY, you should see output