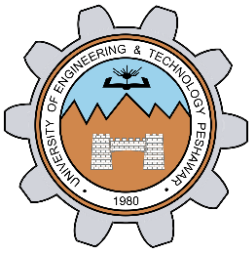
Lab 14

Introduction to MicroBlaze processor



Spring 2025

Submitted by: **Mohsin Sajjad**

Registration No: **22pwsce2149**

Class Section: **A**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Student Signature: _____

Submitted to:

Engr. Faheem Jan

Month Day, Year (17 07, 2025)

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Introduction to MicroBlaze processor

Objective:

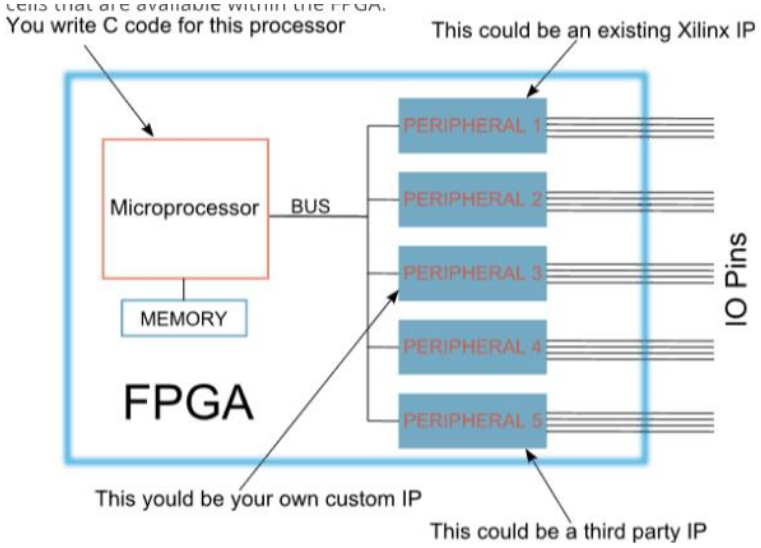
To learn how to use Xilinx MicroBlaze Microcontroller

TOOLS AND SOFTWARE USED

- Xilinx ISE Webpack 14.7
- Xilinx Platform Studio (XPS)
- Xilinx Software Development Kit (SDK)
- Mimas V2 FPGA Development Board (Spartan 6 LX9)
- Base System Builder (BSB) Wizard for Mimas V2
- Serial Terminal Software (TeraTerm or PuTTY)

THEORY

MicroBlaze is a 32-bit RISC soft processor core designed by Xilinx for its FPGA devices. Unlike traditional microcontrollers, MicroBlaze is implemented entirely using the configurable logic within an FPGA, offering tremendous flexibility for custom embedded systems. It supports standard interfaces, customizable peripherals, and is compatible with industry-standard C programming. This lab leverages the MicroBlaze soft core to implement a simple 'Hello World' embedded system project, bridging the gap between HDL-based FPGA designs and traditional C-based embedded systems.

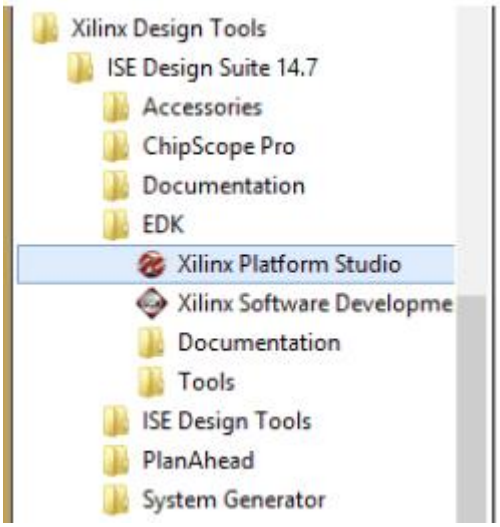


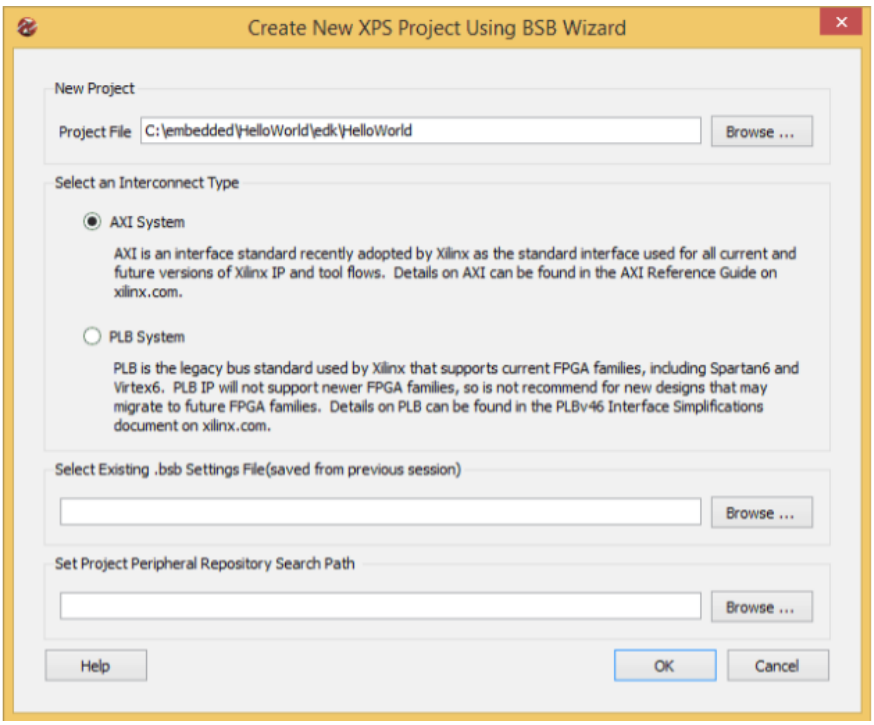
PROCEDURE

Step 1: Create MicroBlaze System in Xilinx Platform Studio

1. Launch Xilinx Platform Studio.
2. Create a new BSB project: File > New BSB Project.
3. Set project directory (e.g., C:\embedded\HelloWorld\edk) and name the project

'HelloWorld'.
4. Select 'Numato Lab Mimas V2' as the development board.
5. Choose UART as the only peripheral.
6. Set UART baud rate to 19200 by configuring the IP core.



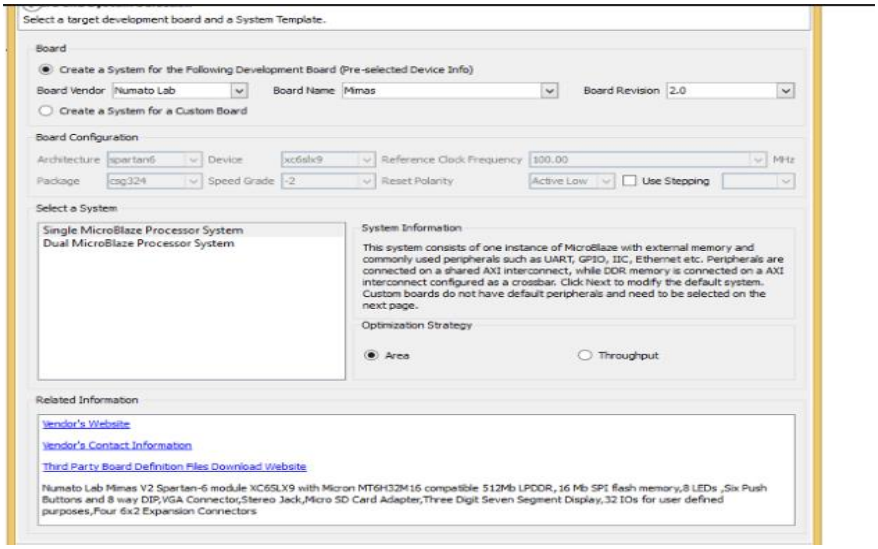


Step 2: Set Startup Clock

1. Go to Project > etc/bitgen.ut.
2. Change '-g StartUpClk:JTAGCLK' to '-g StartUpClk:CCLK'.
3. Save and proceed to bitstream generation.

Step 3: Generate Bitstream

Click the 'Generate Bit File' button. Upon successful completion, a message will confirm bitstream generation.



Step 4: Export to SDK

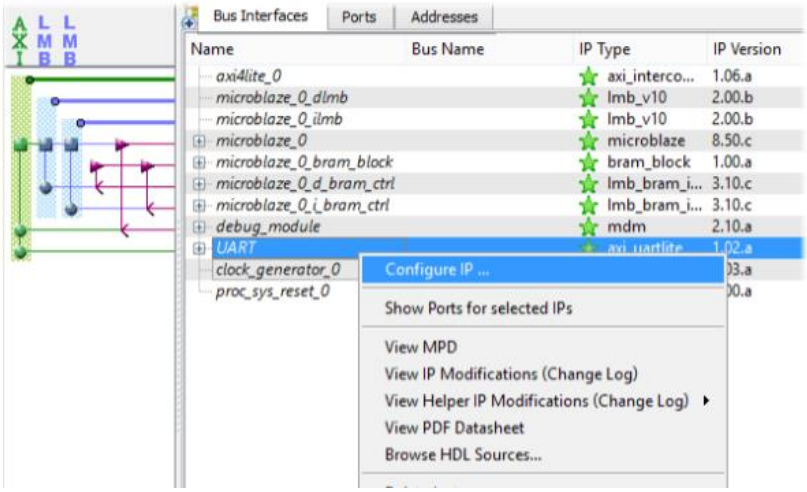
Click 'Export Design' > 'Export Only'. This exports hardware design files needed for application development.

Step 5: Create Application Project in SDK

1. Launch Xilinx SDK and choose workspace folder.
2. Create a new Application Project: File > New > Application Project.
3. Name the project 'HelloWorld'.
4. In 'Target Hardware', click 'Create New' and select HelloWorld.xml from the SDK export directory.
5. Choose 'Hello World' template and finish.

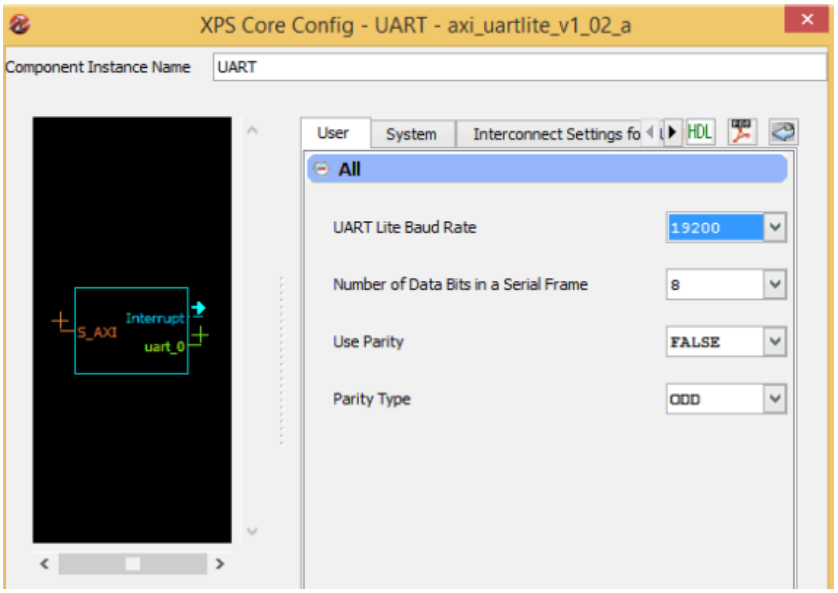
Step 6: Build and Merge ELF File

1. Build the SDK project to generate .elf file.
2. Execute the following command in the XPS console:
   source ipcore_dir/microblaze_mcs_setup.tcl
3. Merge the .elf with the bit file:
   microblaze_mcs_data2mem sdk/HelloWorld/Debug/HelloWorld.elf
4. Generate binary file using:



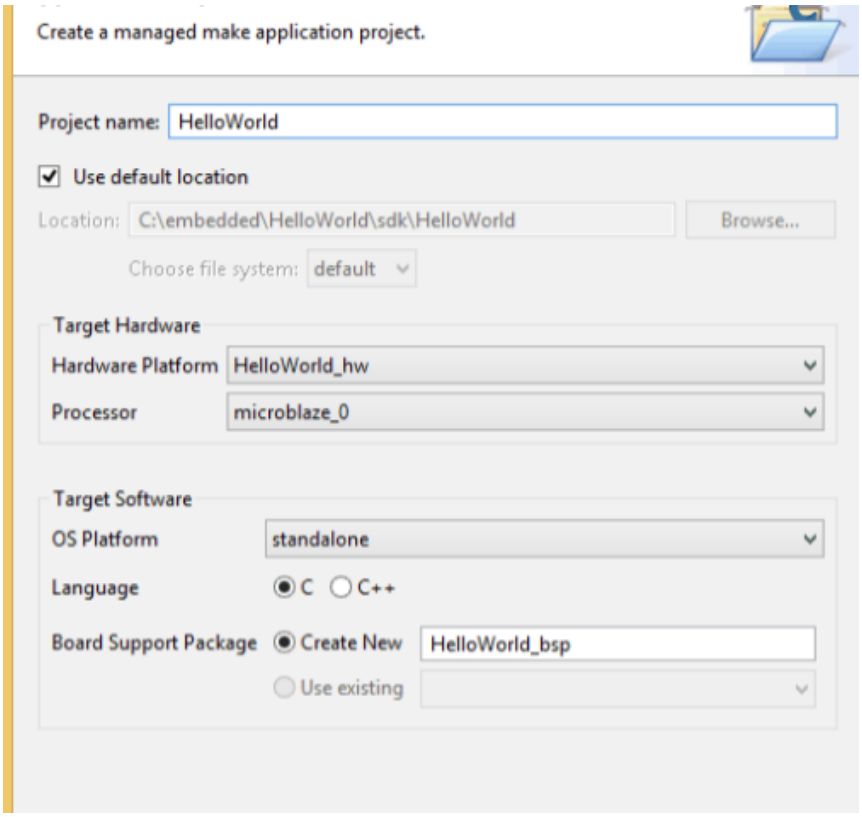
promgen -w -p bin -u 0x0 helloworld.bit -spi -o helloworld

5. Download to the FPGA and monitor 'Hello World' via serial terminal.



RESULTS

The FPGA was successfully programmed with a MicroBlaze-based embedded system. Upon running the application, the message 'Hello World' was displayed on the serial terminal connected to the Mimas V2 board.



DISCUSSION

This lab provided insight into designing embedded systems using FPGA. The integration of the MicroBlaze soft processor allows designers to combine the power of programmable logic with software programmability. The ability to customize peripherals and buses offers greater flexibility than fixed-function microcontrollers. However, this flexibility comes at the cost of increased design complexity and resource usage. Understanding how to use Xilinx's Platform Studio and SDK prepares students to develop scalable and customizable embedded solutions.

CONCLUSION

Through this lab, students learned how to configure and implement a soft processor on FPGA using Xilinx tools. The experience bridged traditional C-based programming with FPGA hardware design, culminating in a working 'Hello World' system. The lab reinforced concepts of hardware-software co-design and introduced the workflow for building embedded systems with MicroBlaze.