LAB 3
ADDING CUSTOM IP TO THE SYSTEM

AIM

To create a **custom AXI4-Lite peripheral** using Vivado IP Packager, integrate it with the Zynq Processing System, connect it to on-board LEDs of the Zybo board, and generate the bitstream.

OBJECTIVE

1. To understand AXI4-Lite based custom IP creation
2. To use the IP Packager feature of Vivado to create a custom peripheral.
3. To modify the functionality of the custom IP
4. To integrate the custom IP into a Zynq block design
5. To assign pin location constraints for LEDs
6. To add AXI BRAM Controller and Block RAM
7. To generate and validate the hardware bitstream

HARDWARE AND SOFTWARE USED

Category	Description
FPGA Board	Xilinx Zybo (Zynq-7000)
Processor	ARM Cortex-A9
Interface	AXI4-Lite
Design Tool	Vivado Design Suite
Memory	BRAM (8 KB)

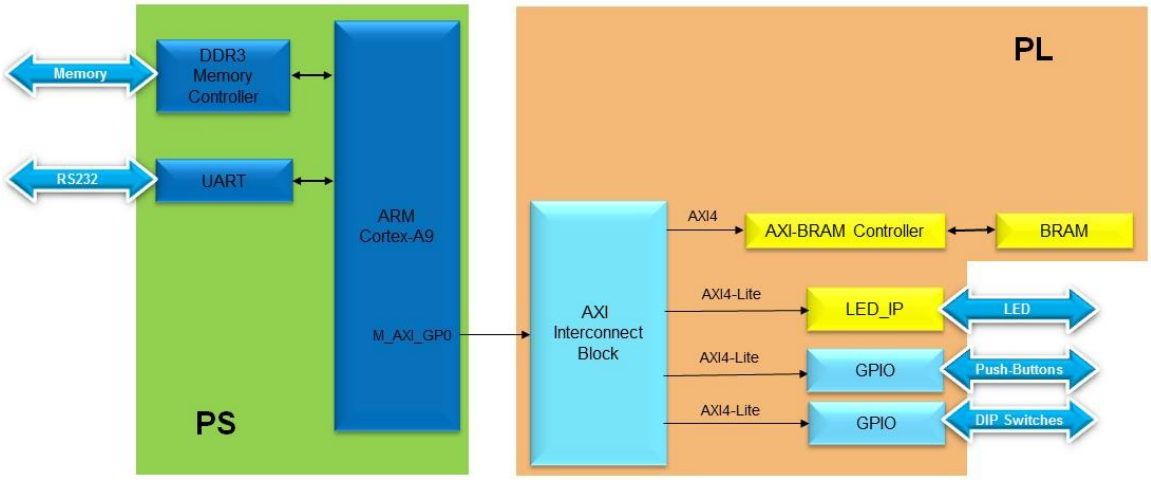
THEORY

The **Zynq-7000 SoC** integrates a **Processing System (PS)** with **Programmable Logic (PL)**, enabling hardware–software co-design. Software executes on the ARM Cortex-A9 processor in the PS, while custom peripherals are implemented in the PL.

In this lab, a **custom AXI4-Lite peripheral** is created using the **Vivado IP Packager**. AXI4-Lite is a lightweight, memory-mapped interface suitable for control and status registers. It allows the processor to write data to registers that control hardware logic implemented in the PL.

The custom IP is connected to the Zynq PS through the AXI interconnect and used to drive the **on-board LEDs** of the Zybo board. **Pin location constraints (XDC)** are added to map the IP output signals to the physical LED pins.

An **AXI BRAM Controller** and **Block RAM (BRAM)** are added to the system to provide on-chip memory, preparing the hardware platform for software execution in subsequent labs. Finally, the complete design is synthesized, implemented, and a **bitstream** is generated to configure the FPGA.



PROCEDURE

1. A Vivado project was created targeting the **Zybo (Zynq-7000)** board.
2. The **Zynq Processing System** was instantiated and configured using board presets.
3. A **custom AXI4-Lite peripheral** was created using the **Vivado IP Packager**.
4. The generated AXI IP template was modified to include LED control logic.
5. The custom IP was packaged and added to the Vivado **IP Repository**.
6. The packaged IP was instantiated in the block design and connected to the PS using **AXI4-Lite interface**.

7. Address mapping for the custom IP was verified using the **Address Editor**.
8. An **AXI BRAM Controller** and **Block RAM (8 KB)** were added to the system.
9. LED signals were made external and **pin location constraints (XDC)** were applied.
10. The design was validated, synthesized, implemented, and the **bitstream was generated**.

CODE

User Logic for LED Control (user_logic.v)

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////
Module Name: lab3_user_logic


module lab3_user_logic # (
    parameter LED_WIDTH = 8
)
(
    input  S_AXI_ACLK,
    input  slv_reg_wren,
    input  [2:0]  axi_awaddr,
    input  [31:0] S_AXI_WDATA,
    input  S_AXI_ARESETN,
    output reg [LED_WIDTH-1:0] LED
);

always @(posedge S_AXI_ACLK)
begin
    if (S_AXI_ARESETN == 1'b0)
        LED <= 4'b0;
    else
        if (slv_reg_wren && (axi_awaddr == 3'h0))
            LED <= S_AXI_WDATA[LED_WIDTH-1:0];
end

endmodule
```

The module lab3_user_logic implements LED control using an AXI4-Lite interface. The parameter LED_WIDTH defines the number of LED outputs. On every positive edge of S_AXI_ACLK, the LED output is updated based on AXI write transactions.

When the active-low reset signal (S_AXI_ARESETN) is asserted, all LEDs are cleared. When a valid AXI write enable (slv_reg_wren) is detected and the address corresponds to register location 0x0, the LED output is assigned the lower bits of the AXI write data (S_AXI_WDATA).

Thus, the processor can control the LEDs by writing values to the AXI register mapped to address offset 0x0.

OBSERVATIONS

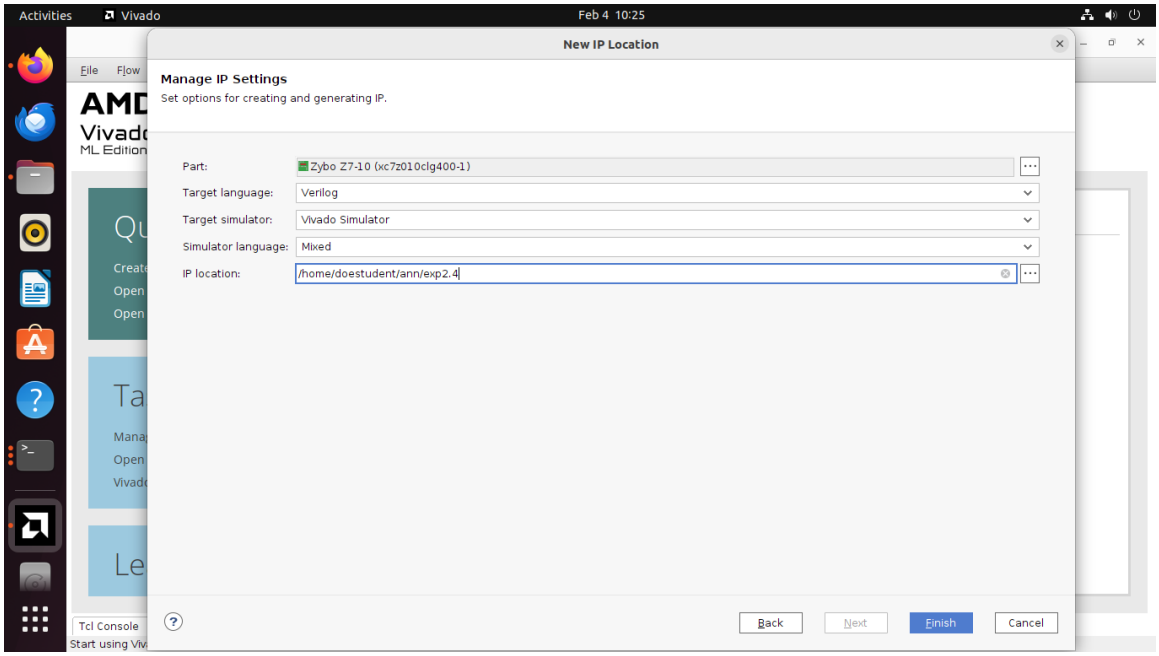


Fig 1: Manage IP Window

The custom AXI4-Lite LED IP was successfully created and packaged using Vivado IP Packager and appeared in the IP repository without errors.

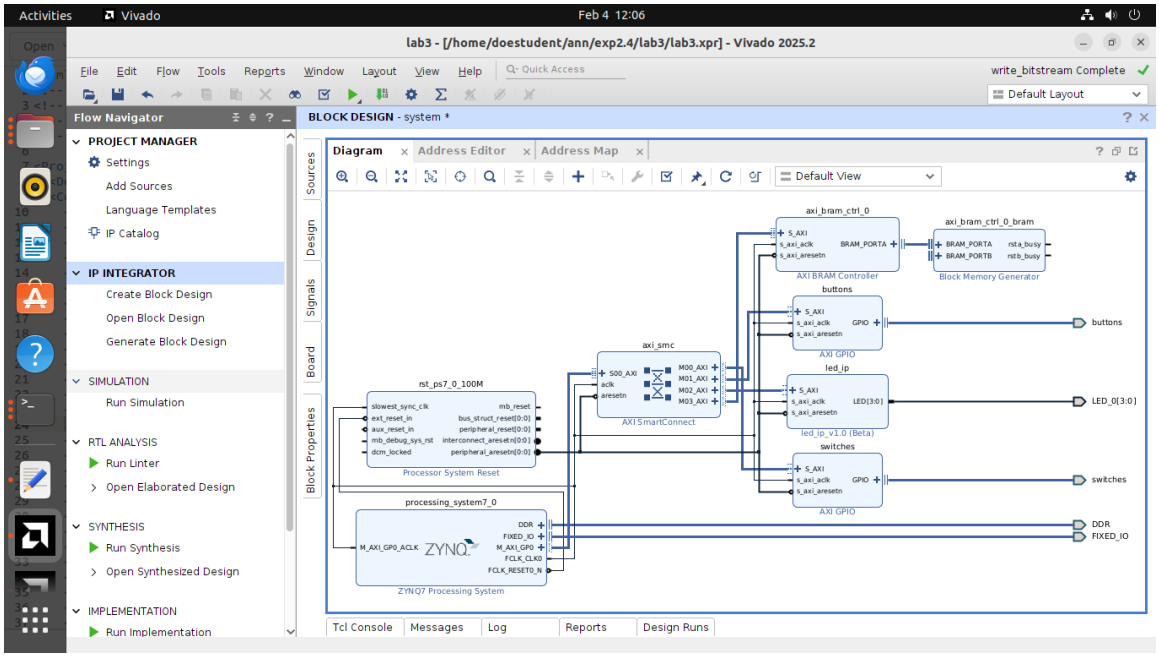


Fig 2: Block Design

The block design shows correct integration of the Zynq Processing System, custom LED IP, AXI interconnect, and AXI BRAM Controller.

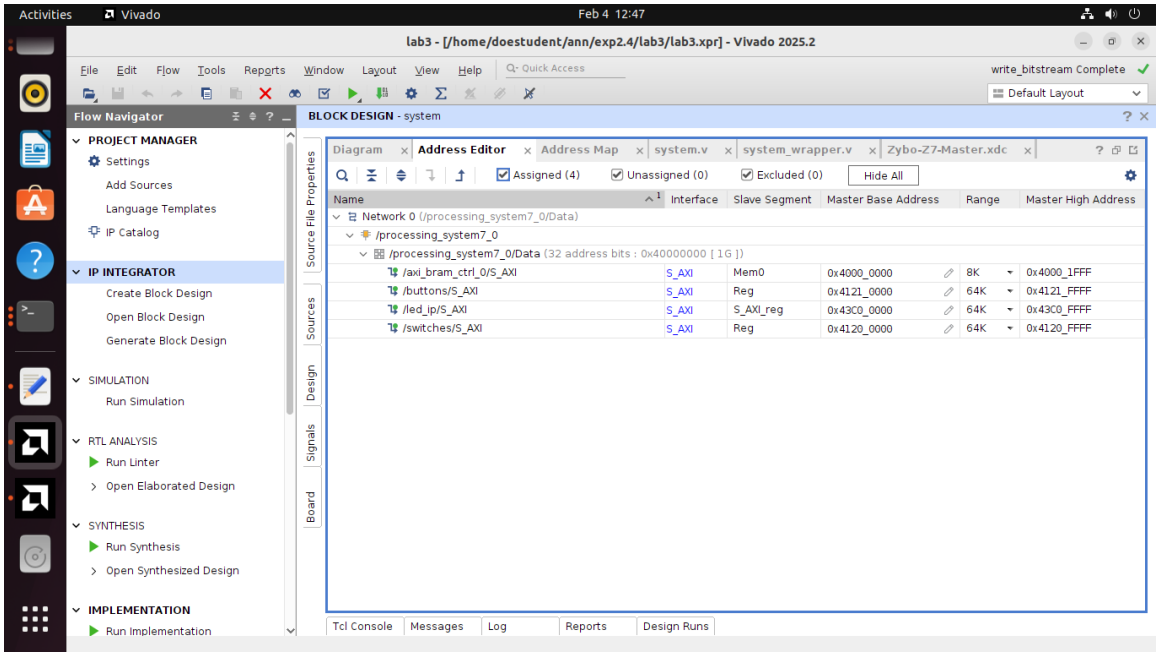


Fig 3: Address Editor

The Address Editor confirms that a valid and non-overlapping base address range was assigned to the custom AXI peripheral and the BRAM.

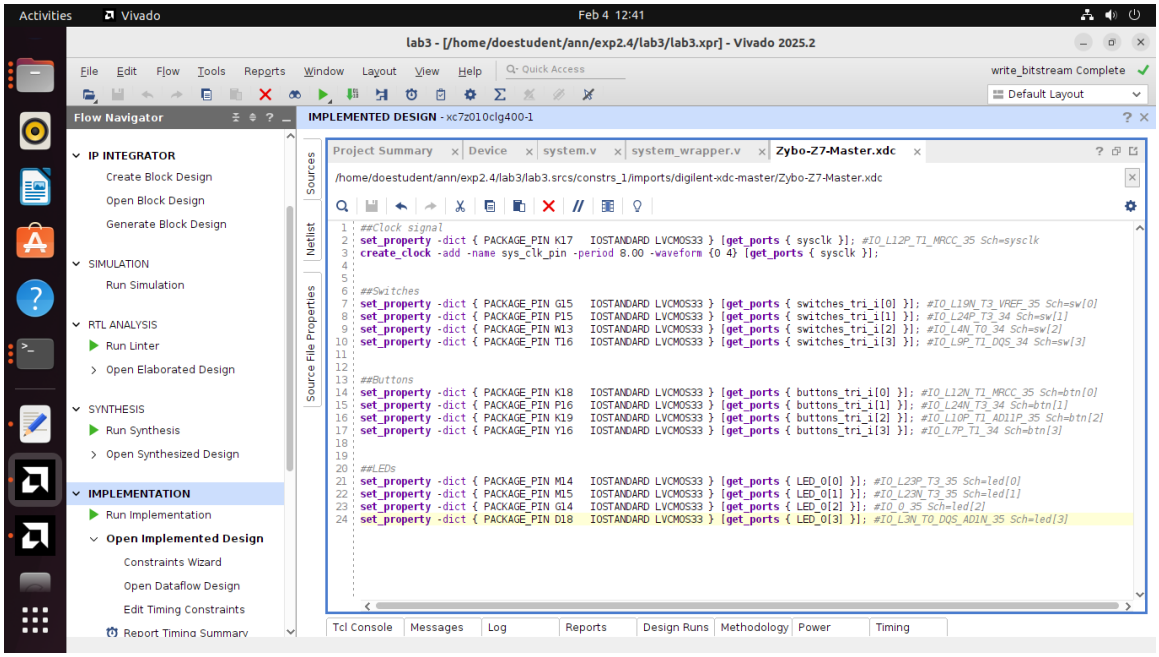


Fig 4: Constraint File (XDC)

The constraints file correctly maps the LED output ports of the custom IP to the physical LED pins of the Zybo board.

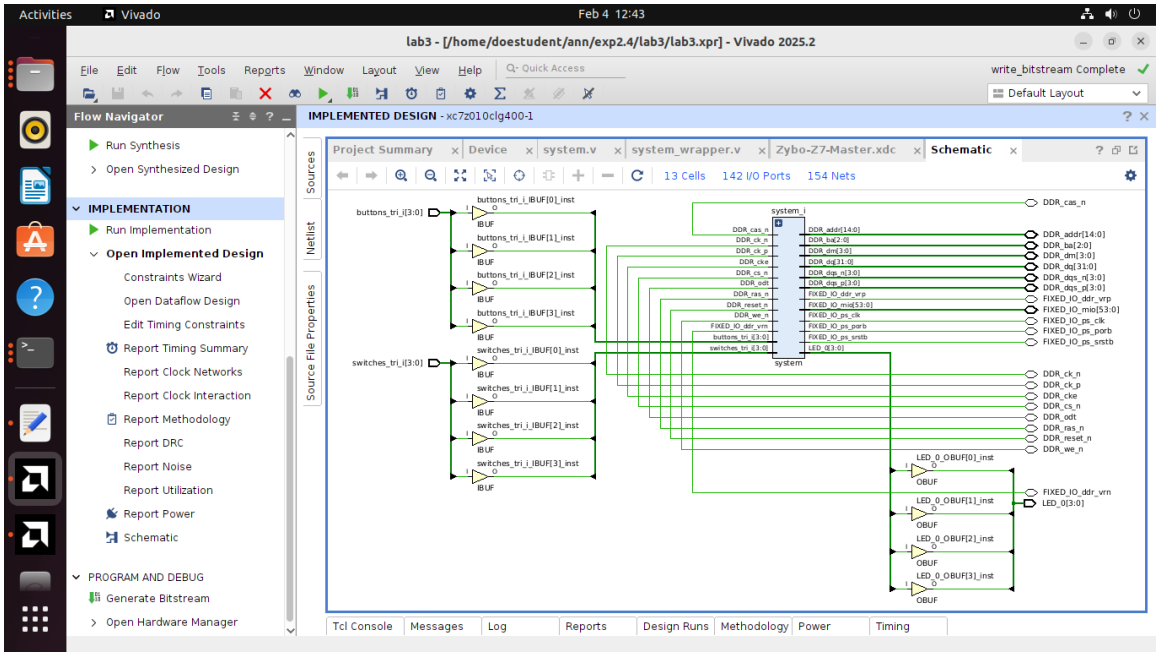


Fig 5: Schematic View

The schematic view verifies correct signal connectivity between AXI registers, user logic, and LED output ports.

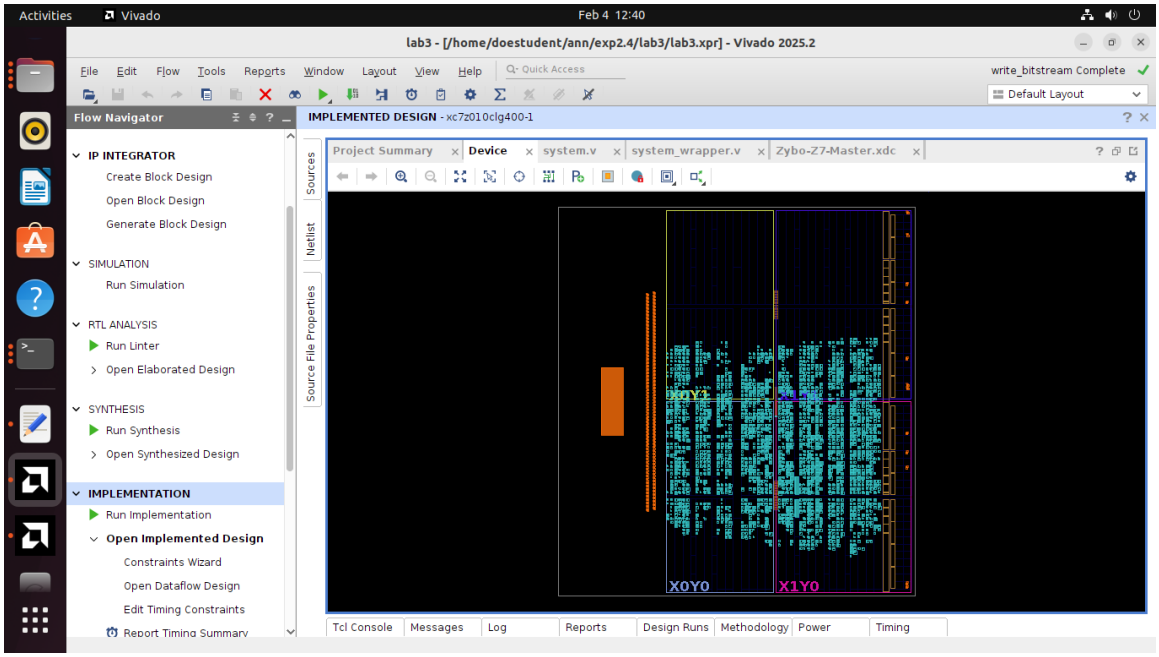


Fig 6: Implemented Design

The implemented design view confirms successful synthesis and implementation of the complete hardware system.

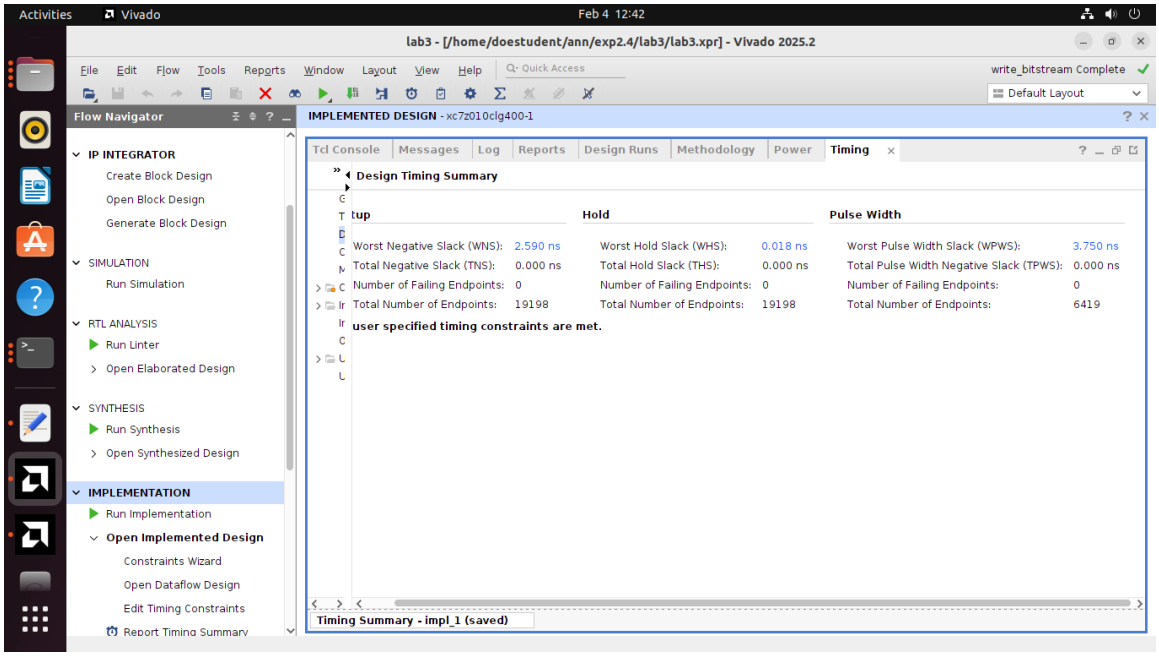


Fig 7: Timing Report

The timing analysis report indicates that all setup and hold timing constraints are met, showing a timing-clean design.

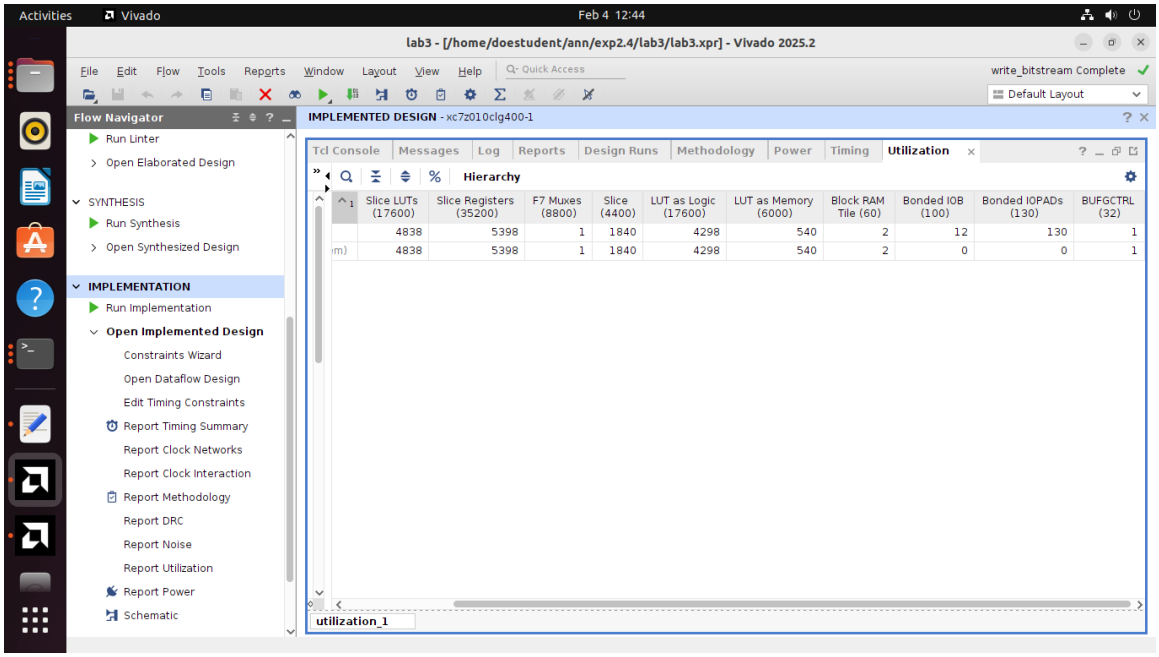


Fig 8: Utilization Report

The utilization report shows acceptable usage of LUTs, flip-flops, BRAMs, and other FPGA resources within device limits.

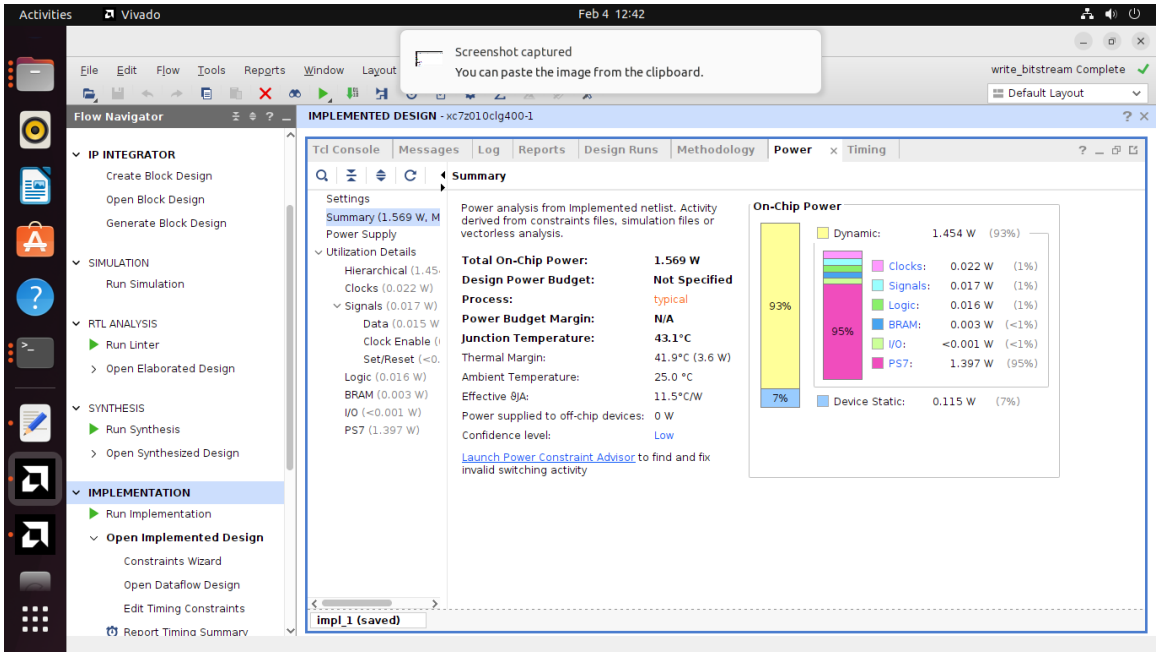


Fig 9: Power Report

The power analysis report indicates that the total on-chip power consumption is within safe operating limits.

RESULTS

Custom AXI4-Lite peripheral was successfully created, packaged, and integrated into the Zynq-7000 system. The peripheral was connected to on-board LEDs using proper pin constraints, BRAM was added to the design, and the complete system was synthesized, implemented, and a valid bitstream was generated successfully.

CONCLUSION

This lab successfully demonstrated the process of creating and integrating a custom AXI peripheral using Vivado IP Packager. The experiment validated effective communication between the Processing System and Programmable Logic through AXI4-Lite interface. Resource utilization, timing, and power analysis confirmed that the design meets implementation constraints, and the hardware platform is now ready for software development and debugging in subsequent labs.

REFERENCE

https://xilinx.github.io/xup_embedded_system_design_flow/lab3.html