MNIST CNN Accelerator Design on Zynq-7000 SoC

Presenter Names:

V. Ashish Nadh - VU22EECE0100389

P. Deepika - VU22EECE0100129

Y. Manohar - VU22EECE0100099

B. Aryan Ratnakar - VU22EECE0100175

Project Guide :

Dr. G. V. K. Sharma

Department of Electrical, Electronics and Communication Engineering

GITAM School of Technology,

GITAM (Deemed to be University), Visakhapatnam.

Table of Contents

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

Introduction & Problem Understanding

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

Application Context: This project implements a hardware accelerator for a Convolutional Neural Network (CNN) specifically designed for **handwritten digit detection and recognition**. The target model is optimized to achieve high accuracy on the **MNIST dataset**, which consists of 28x28 pixel grayscale images.

Technical Problem: The core technical challenge is accelerating the massive computational load of a CNN – particularly the convolution and fully connected layers – within the strict power and resource constraints of an edge device. Relying on the CPU alone is too slow, while a GPU is too power-hungry.

Project's Technical Goal: The goal is to design a low-power, high-efficiency hardware accelerator in the FPGA's Programmable Logic (PL). This accelerator will offload the heavy computations from the ARM Cortex-A9 processor, following a **hardware-software co-design** approach on the ZEDBoard. Success is defined by minimizing power-hungry external memory accesses through on-chip data reuse.

Methodology

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

Our proposed methodology follows a structured hardware-software co-design process, broken down into distinct phases. This approach allows for systematic development, testing, and validation of the CNN accelerator on the Zynq-7000 SoC platform.

Phase 1: CNN Model Design & Architecture :

- Design of Convolution Neural network CNN in detail (Convolution layer, pooling layer, Fully connected layers, Activation Functions etc.)

Phase 2: RTL Design & Functional Simulation

- **Modular Hardware Design:** Develop individual Verilog modules for each core function: Convolution Layer, Activation Function , Pooling Layer, Fully Connected Layer and a top-level controller.
- **Behavioral Simulation:** Create a Verilog testbench to simulate and verify the functional correctness of each module independently and then the integrated accelerator as a whole.

Methodology

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

Phase 3: System Integration with Zynq SoC

- **IP Core Creation:** Package the verified RTL design into an IP (Intellectual Property) core using Xilinx Vivado.
- **Block Design:** Integrate the custom CNN accelerator IP with the Zynq-7000's ARM Processing System (PS) using the standard AXI bus for communication.

Phase 4: Software Development & On-Board Validation

- **Software Development :** Develop a C/C++ application in the Xilinx SDK to run on the ARM processor. This application will manage data transfer to/from the accelerator and control its operation.
- **System Testing:** Generate the final bitstream, program the FPGA, and run the complete system to perform inference on MNIST test images.
- **Verification:** Validate the hardware's classification output against the results from a software model to confirm correctness.

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

CNN ARCHITECTURE

7-Layer Modified LeNet-5 Architecture

Why We Chose This Architecture?

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

- **Hardware Constraints:** The ZedBoard (Zynq-7000) has limited On-Chip Memory (BRAM) and DSP slices.
- **Bottleneck Analysis:** Standard models like VGG-16 (138M parameters) are too heavy and require slow external DDR memory access.
- **Our Solution:** We selected a customized 7-Layer Architecture (Modified LeNet-5).
- **Key Benefit:** Extremely lightweight (~60k parameters), allowing the entire model to fit inside the FPGA's fast internal BRAM for real-time processing.

Proposed 7-Layer CNN Architecture

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

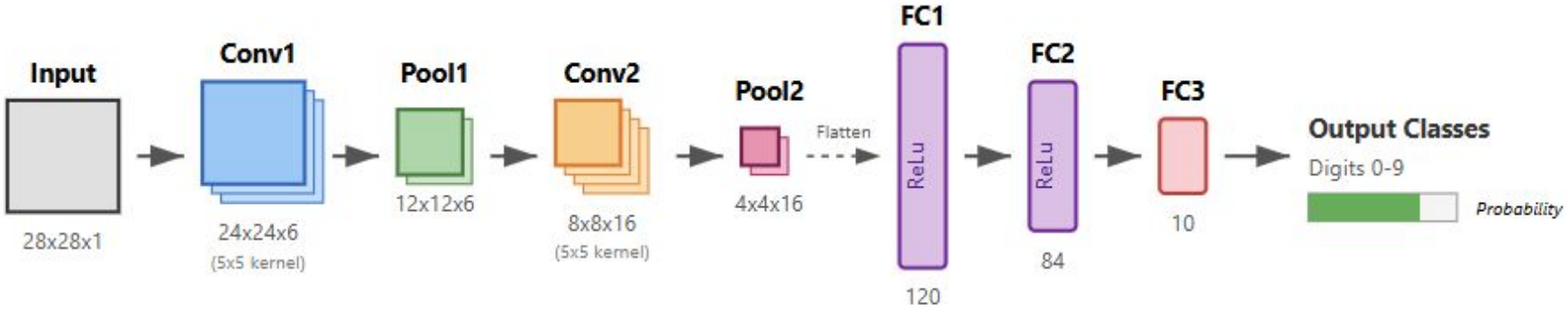
- **Input:** 28 X 28 Grayscale Image.
- **Feature Extraction:** 2 Convolutional Layers (to detect edges and shapes).
- **Downsampling:** 2 Max-Pooling Layers (to reduce data size by 75% at each step).
- **Classification:** 3 Fully Connected Layers (mapping features to the 10 digits).
- **Final Output:** Probability vector for Digits 0-9.

Detailed Layer Configuration

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

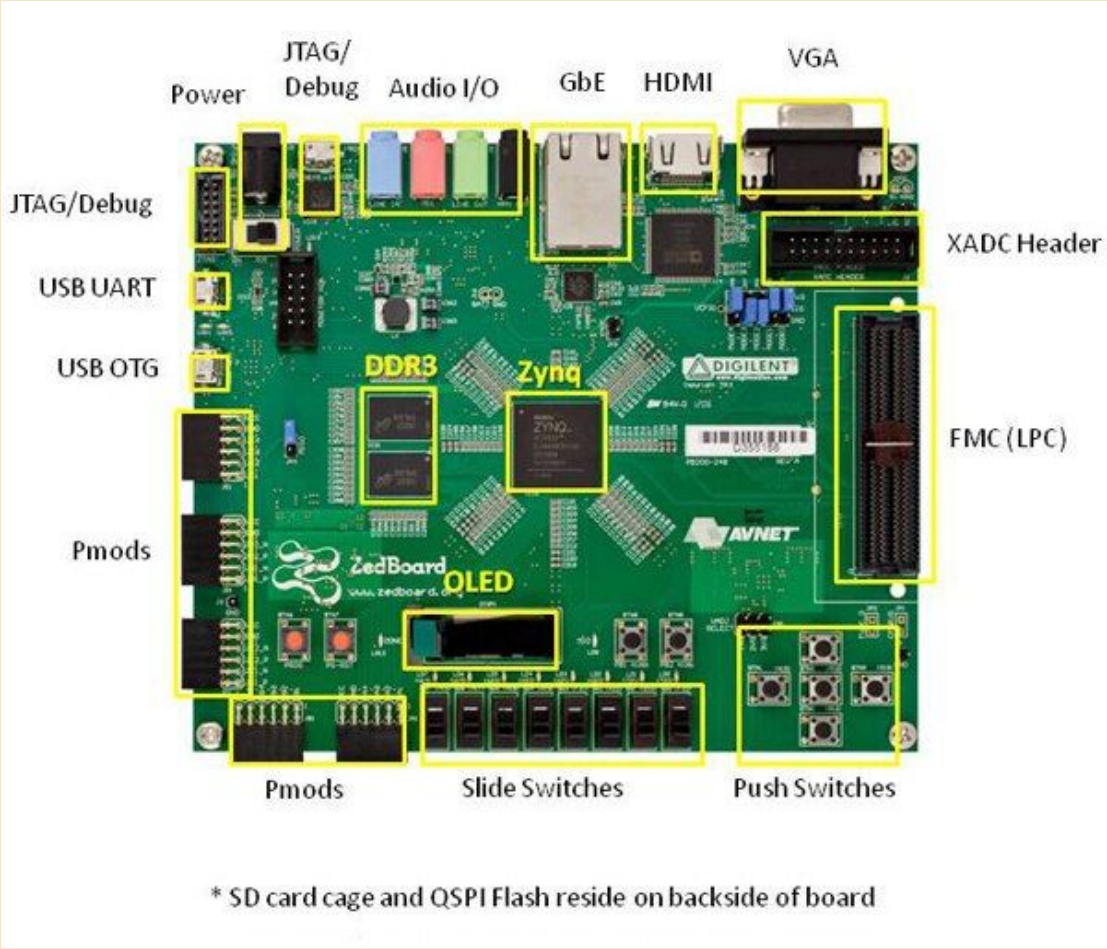
- **Layer 1 (Conv1):** 6 Filters, Kernel Size 5 X 5, Output: 24 X 24 X 6.
- **Layer 2 (Pool1):** Max Pooling, Window 2 X 2 (Stride 2), Output: 12 X 12 X 6.
- **Layer 3 (Conv2):** 16 Filters, Kernel Size 5 X 5, Output: 8 X8 X 16.
- **Layer 4 (Pool2):** Max Pooling, Window 2 X 2 (Stride 2), Output: 4 X 4 X 16.
- **Fully Connected Layers:**
  - **FC1:** 120 Neurons
  - **FC2:** 84 Neurons
  - **Output:** 10 Neurons (Softmax equivalent)

7-Layer CNN Architecture for ZedBoard (Modified LeNet-5)



GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

ZEDBOARD ARCHITECTURE AND EXERCISES



ZEDBOARD ARCHITECTURE

ZED BOARD Exercises (1,2,3)

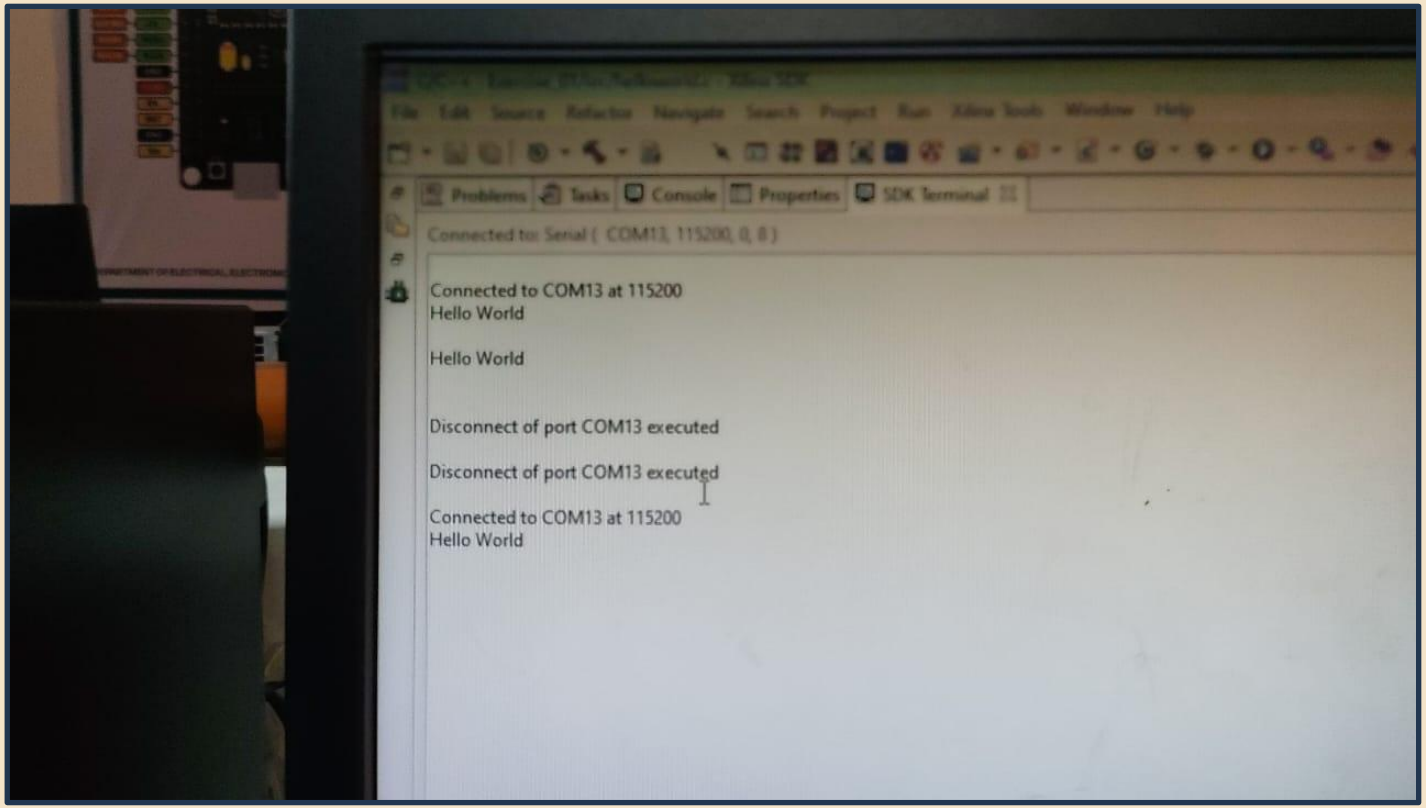
GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

Objective: To verify the toolchain (Vivado/SDK) and establish control over the ARM Cortex-A9 Processing System.

Implementation:

- **Hardware:** Configured the **ZYNQ7 Processing System** with ZedBoard presets (DDR Memory & Clock generation).
- **Software (BSP):** Utilized Xilinx Board Support Package (BSP) drivers.
- **Debugging environment:** Explored the Debugging environment in Xilinx SDK
- **Key Result (Exercise 1 & 2):**
  - verified the functionality of the Zynq chip's Processing System (PS) on the ZedBoard through "Hello World" in the SDK terminal.
  - Bypassed the FPGA fabric to control **MIO Pin 7** directly from the Processor.
  - Successfully toggled User LED **LD9**.

ZEDBOARD Exercise 1



SDK Terminal showing "Hello World"

ZEDBOARD Exercise 2

LED (LD9) Blinking on the Processing System (PS)

ZEDBOARD Exercise 4

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

Hardware-Software Architecture

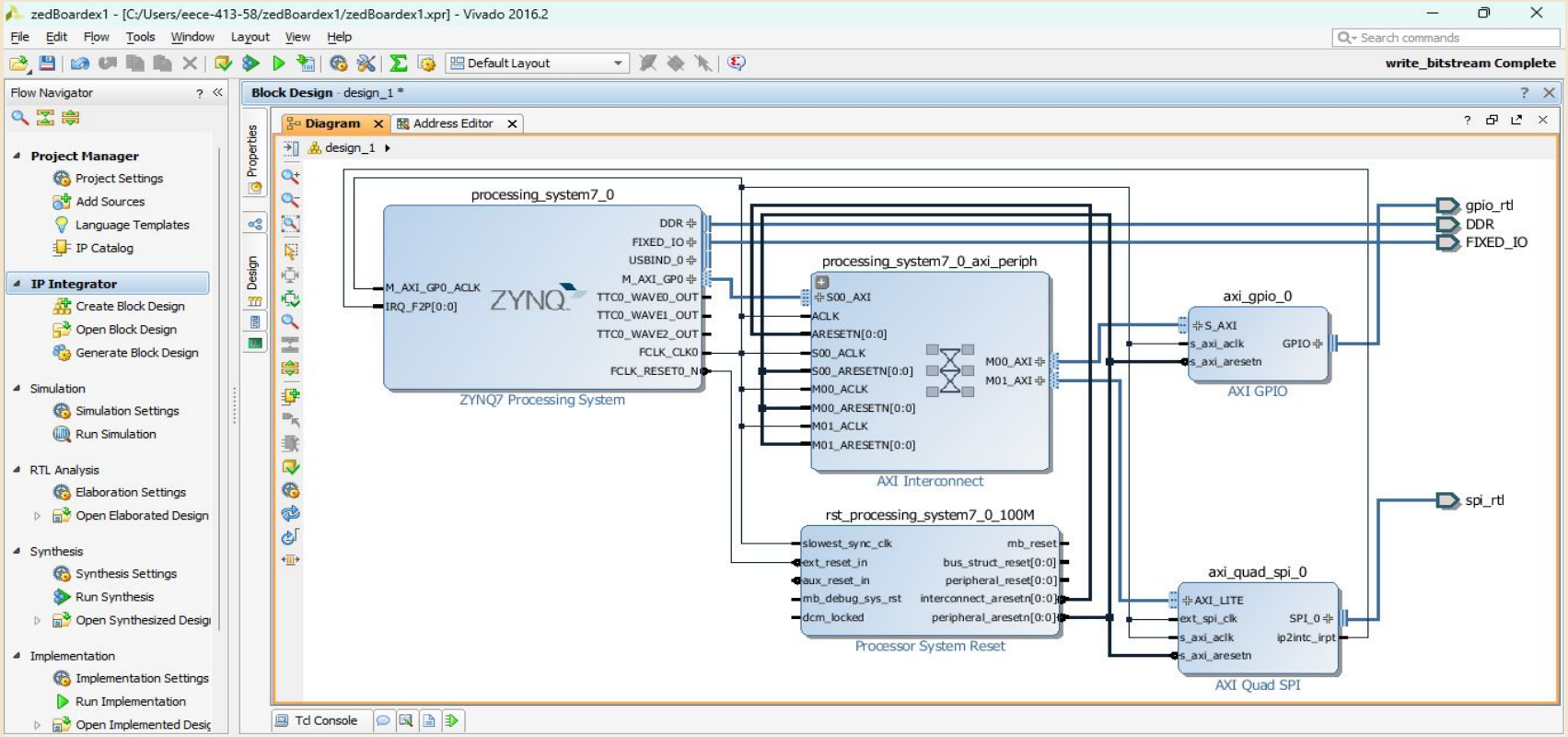
Objective:

To establish a communication pipeline between the ARM Cortex-A9 Processor (PS) and Custom FPGA Logic (PL).

System Architecture:

- **The Bridge (AXI Interface):**
  - Enabled **M_AXI_GP0** (General Purpose Master) port on the Zynq.
  - This converts software commands into hardware signals.
- **The Hardware (AXI GPIO IP):**
  - Integrated a **16-bit AXI GPIO** block into the PL fabric.
  - Configured as a **Bidirectional Bus** to handle both input and output data simultaneously.
- **Physical Mapping (Constraints):**
  - **Input Map:** Bits [0-7] ⟹ Physical DIP Switches (F22 - M15).
  - **Output Map:** Bits [8-15] ⟹ Physical LEDs (T22 - U14).

Vivado IP BLOCK Design



ZEDBOARD Exercise 5

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

Interactive System Logic & Results

Control Algorithm (The "Loop"):

The system operates on a continuous feedback loop:

1. **INPUT (Hardware ⟹ Software):**
   - CPU reads the state of the **8 DIP Switches** from the PL addresses.
2. **PROCESS (Software Logic):**
   - CPU performs a **Left Bit-Shift (<< 8)** operation.
   - This moves the data from the "Input" bit positions (0-7) to the "Output" bit positions (8-15).
3. **OUTPUT (Software ⟹ Hardware):**
   - CPU writes the processed value back to the **LED Registers**, instantly lighting the corresponding LEDs.

Experimental Outcome:

- **Latency:** Real-time response (Zero perceptible delay).
- **Verification:** Verified that SW[0] ON ⟹ LED[0] ON, confirming successful bidirectional data transfer.

ZEDBOARD Exercise 5

LEDs Blinking by flipping DIP Switches on the Programmable logic (PL)

ZEDBOARD Exercise 6

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

Memory Interfacing & Direct Register Access

Goal: To understand the memory-mapped architecture of the Zynq-7000 SoC and perform direct I/O operations without high-level drivers.

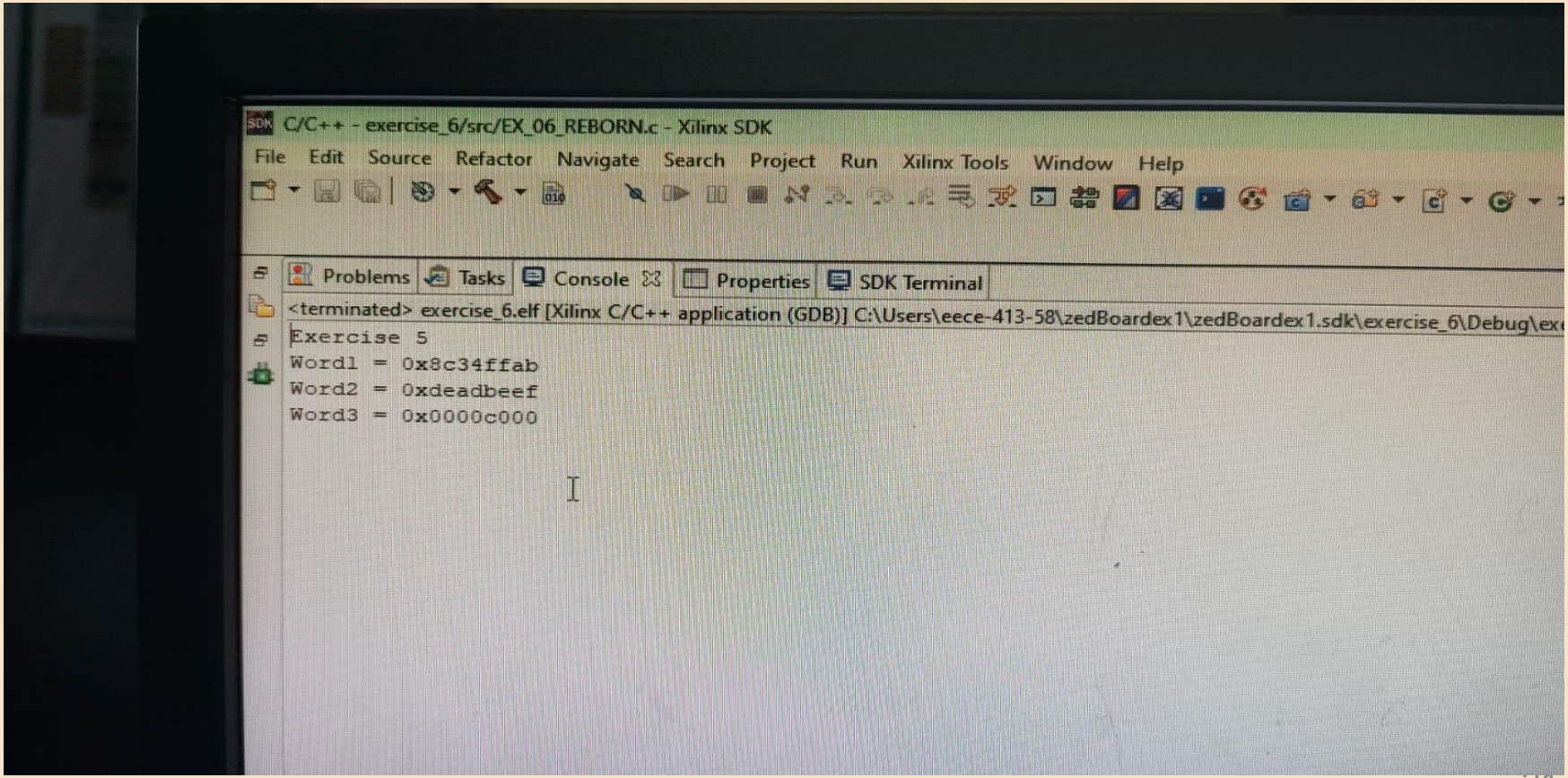
Technical Details

- **Memory-Mapped I/O:** Every peripheral and memory block (OCM, DDR, GPIO) is assigned a unique physical address. We used `xil_io.h` for low-level access.
- **On-Chip Memory (OCM):** Explored the high-speed 256KB RAM inside the Processing System (PS).
- **Direct Address Manipulation:** We targeted address `0xE000A064` to read the raw state of the PS-side GPIO pins.

Experimental Results (Output)

- **Word 1:** `0x8C34FFAB` - Successfully wrote and verified a 32-bit pattern in OCM.
- **Word 2:** `0xDEADBEEF` - Verified successful word-aligned memory transactions.
- **Word 3 (Interaction):** `0x0000C000` - When **Push Buttons (PB1/PB2)** were pressed, we observed real-time changes in **bits 14 and 15** of the register.

ZEDBOARD Exercise 6



ZEDBOARD Exercise 7

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

Hardware Timers (Polled Mode)

Goal: To achieve precise time measurement using the Private SCU (Snoop Control Unit) Timer instead of software delay loops.

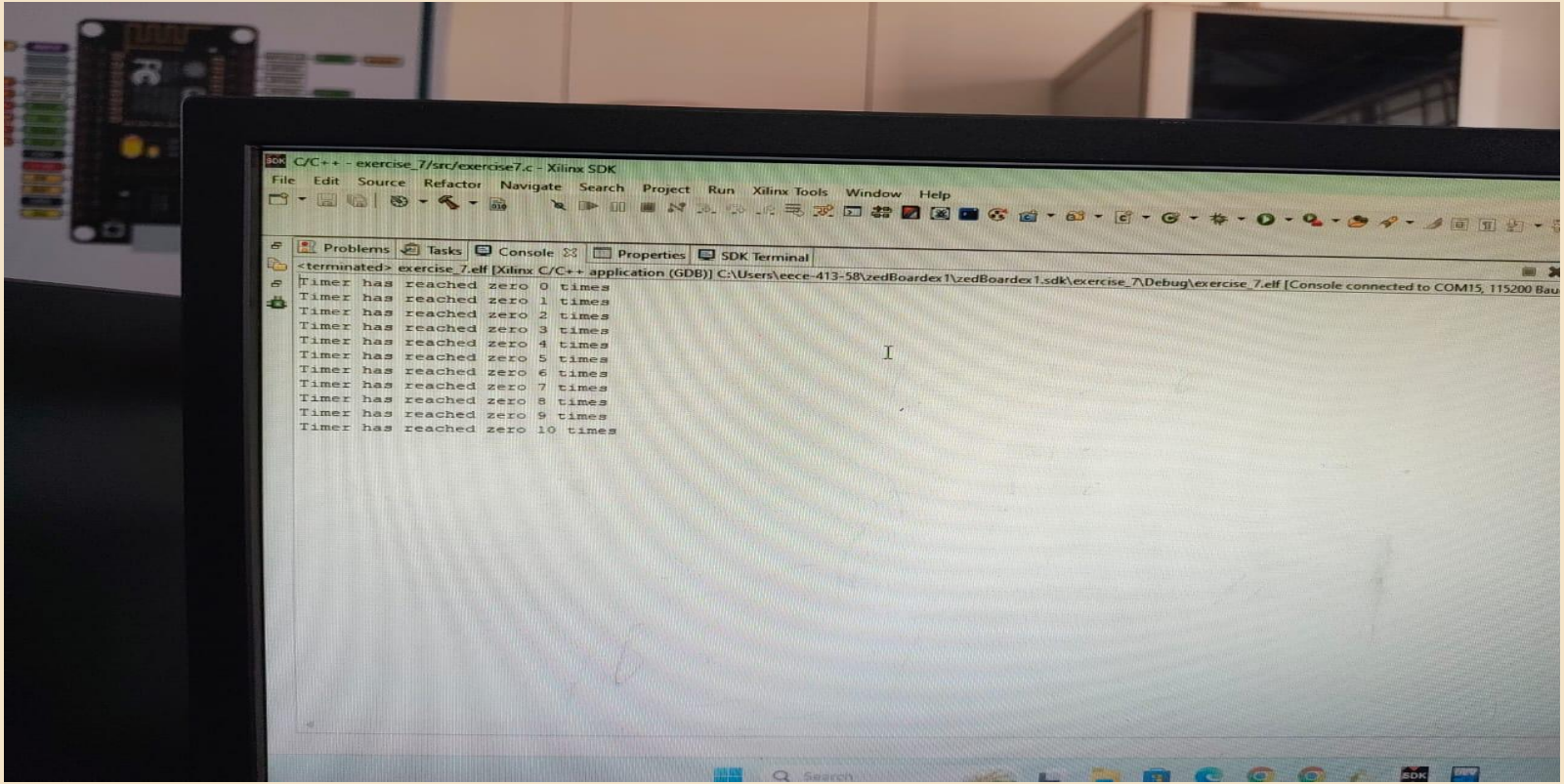
Technical Details

- **SCU Timer:** A 32-bit hardware counter that decrements at half the CPU clock frequency (333 MHz on the ZedBoard).
- **Polled Mode Logic:** The CPU enters a loop that continuously checks the "Timer Counter" register. Once it reaches zero, the event is triggered.
- **Calibration:** Used `XScuTimer_Config` to initialize the hardware and set a precise 1-second (1Hz) interval.

Experimental Results (Output)

- **Terminal Display:** "the timer has reached zero 0 times" ... up to "the timer has reached zero 10 times."
- **Precision:** Unlike software delays, this output remained perfectly consistent at exactly 1000 ms intervals.

ZEDBOARD Exercise 7



ZEDBOARD Exercise 8

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

Real-Time Interrupt Handling

Goal: To implement an efficient system where the hardware "notifies" the CPU of events, allowing for true multitasking.

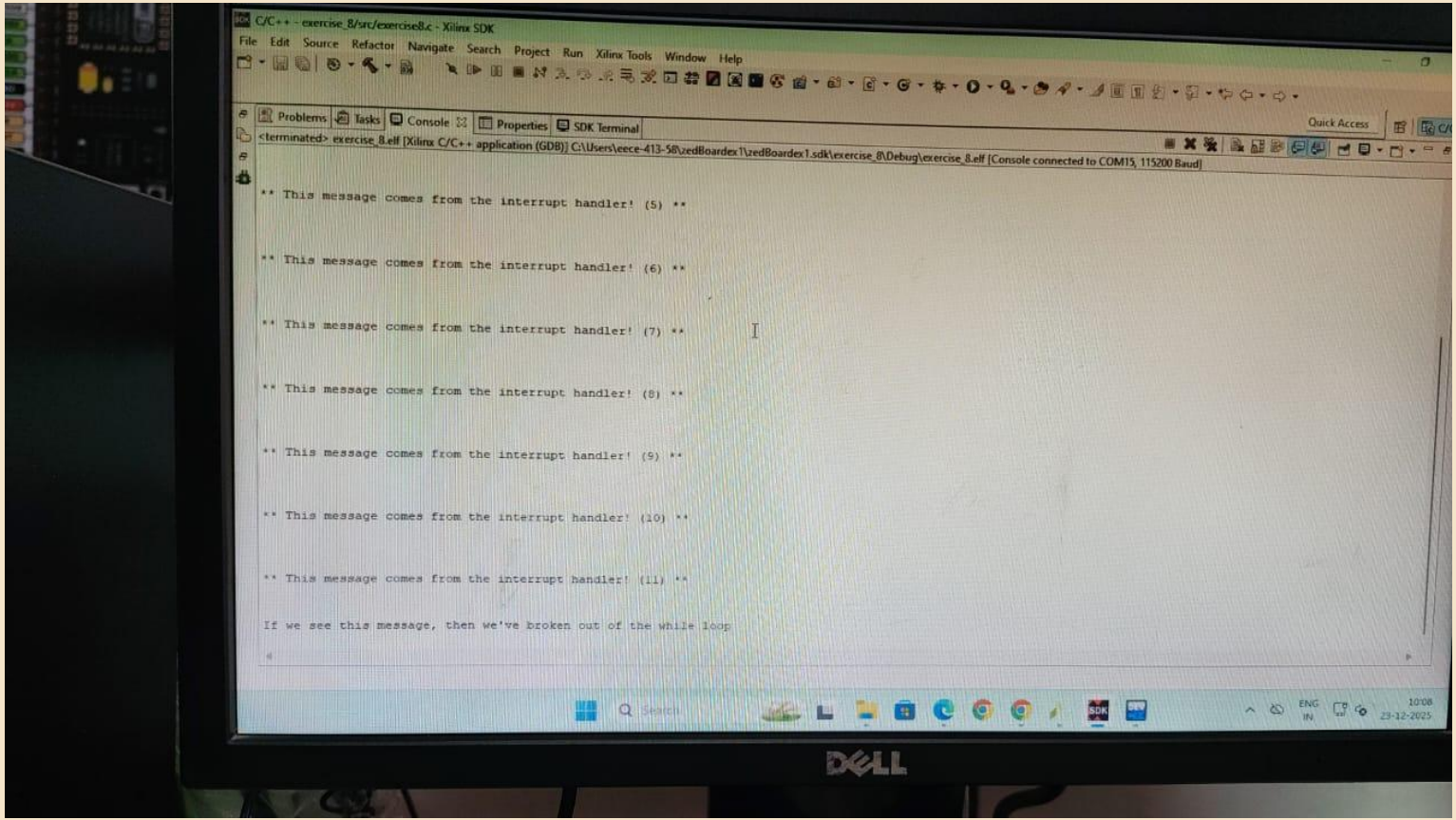
Technical Details

- **Interrupts vs. Polling:** In polling (Ex 7), the CPU is stuck in a loop. With interrupts, the CPU is free to do other work until the hardware signals it.
- **GIC (General Interrupt Controller):** Configured the GIC to route signals from the SCU Timer to the ARM core.
- **ISR (Interrupt Service Routine):** Developed a handler function (`my_timer_interrupt_handler`) that executes only when the timer expires.

Experimental Results (Output)

- **Multitasking Success:** 1. **Task A (Background):** Printed "message comes from interrupt handler" to the UART terminal at 1Hz. 2. **Task B (Foreground):** Continuously updated the **8 LEDs** in real-time based on the position of the **DIP Switches**.

ZEDBOARD Exercise 8



Future Work

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

Our plan for completing this project is a direct execution of our methodology. The following steps outline our roadmap leading up to the final project delivery.

- **CNN Model architecture and Design - completed**
- **RTL Implementation & Simulation**
  - **Goal:** To translate the CNN architecture into hardware and verify its logic.
  - **Tasks:**
    - Develop modular Verilog code for the convolution engine, pooling/activation units, and the fully connected layer.
    - Write a comprehensive testbench to simulate the data flow through the entire accelerator.
    - Debug and validate the RTL design to ensure it is functionally correct.

Future Work

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

System Integration

- **Goal:** To integrate the verified hardware accelerator with the Zynq processing system.
- **Tasks:**
  - Package the Verilog design into an AXI4-compliant IP core in Vivado.
  - Create the complete SoC block design, connecting our IP to the ARM processor.
  - Synthesize and implement the design, ensuring it meets timing and resource constraints.

Future Work

GITAM
SCHOOL OF ELECTRICAL, ELECTRONICS AND COMMUNICATION ENGINEERING

Software Development & On-Board Validation

- **Goal:** To control the accelerator with software and demonstrate the complete, working system.
- **Tasks:**
  - Write a C/C++ application to send image data and control signals to the accelerator.
  - Deploy the bitstream and software application to the Zynq board.
  - Perform end-to-end testing with MNIST images and verify the classification results on a terminal.

References

1. *Jiang, J., Zhou, Y., Gong, Y., Yuan, H., & Liu, S. (2025). FPGA-based Acceleration for Convolutional Neural Networks: A Comprehensive Review. arXiv preprint arXiv:2505.13461.*
2. *Srilakshmi, S., & Madhumati, G. L. (2023). A Comparative Analysis of HDL and HLS for Developing CNN Accelerators. 2023 International Conference on Intelligent Systems and various other Computer Applications (ICISVCA).*
3. *García-López, J., et al. (2020). A hardware accelerator for the inference of a Convolutional Neural Network. Ingeniería, 25(1), 107-118.*
4. *Khan, M. A., et al. (2022). A Step-by-Step Re-Configurable CNN Engine Design on FPGA. Electronics, 11(23), 3883.*

Thank You