A blue and black logo

Description automatically generated

Project 4: System-on-Chip Design (SoC)

**Team Members:** Yousef Alfailakawi, Samonorom Chum

**Course:** EECE.4500/5500 Advanced Digital System Design

**Due Date:** 12/12/2024

**Professor:** Dr. Orlando Arias

# Summary

Project 4 aims to design and implement a custom Avalon Memory-Mapped (MM) agent to drive seven-segment displays on the DE1-SoC FPGA board. The project involved creating a custom IP core, integrating it with the HPS (Hard Processor System) using Platform Designer (Qsys), and verifying the functionality through simulation and hardware testing. An optional simulation was also set up to verify pin displays.

**(30pt)**

* A report summarizing your design, experimentation, and discussing your results.
* The report must include your hardware overhead of your design.
* You must also include a photograph of your board displaying the temperature.
* For graduate students, the report must be at least five (5) pages long and must contain a high level diagram of your design.
* Reports must be typeset as a single spaced, single column on a 12pt serif typeface (e.g. Computer Modern, or Times).
* Any code in the report must use a monospace typeface.

# Experimentation

In this project there are four main sections that we will explore. The sections are IP (Intellectual Properties) Core Design, Test Simulation, Integration and Hardware Testing. The first section, IP Core Design, relates to designing our own IP cores using VHDL. The second section, Test Simulation, involves testing the VHDL IP Core files by using a Python wrapper to examine functionality and outputs by simulating a seven segment display. The third section is about integrating the IP cores into the Cortex-Ap Subsystem using Platform Designer (formerly Qsys.) and Quartus. Lastly, the fourth and final section covers testing the files on a physical test system.

## IP Core Design

* + 1. **Input: (address, write, read, writedata) (Agent’s ports)**

The project begins with the creation of the seven\_segment\_agent.vhd file, which serves as the top-level VHDL module. This module defines the entity containing generics and port interfaces to receive data and control settings from external inputs. The input data, including address, write, read, and writedata, is handled as binary values by the agent.

* + 1. **Change Trigger: Feature Query (`get\_features`) → readdata**

Once the input is received, it is processed by the **Change Trigger**. This process manages resets and read/write operations. Additionally, it uses the get\_features function to query and encode the agent's supported features, which are then returned via the readdata port.

* + 1. **Data Processing: (`data\_driver`)**

The input data is processed within with data\_driver , based on the control settings. The Double Dabble function is used here to convert the binary input into a decimal value for easier processing.

* + 1. **Digit Assignment: (`assign\_digit`) → lamps\_off, lamps\_negative, get\_hex\_digit**

The processed data is mapped to the seven-segment display configuration by the assign\_digit function. This function determines how each digit should be displayed based on the selected features. Helper functions like lamps\_off (to blank digits), lamps\_negative (to show a negative sign), and leading\_blank\_zeros (to handle leading zeros) are used to configure the display appropriately.

* + 1. **Output Generation: (`concat\_function`) → digits**

The **Concat Function (concat\_function)** combines the individual digit configurations into a single output signal called digits. This function integrates various settings, such as decimal mode and negative sign, to prepare the final output for the display.

* + 1. **Physical Display Update: (7-segment display)**

Finally, the digits signal determines which segments (a through g) of each digit in the physical seven-segment display should light up. This drives the visual representation of the input data.

## **Simulation**

* **Test Fixture Setup**:
  + Python-based cocotb testbench simulated the IP core.
  + Pygame visualization displayed seven-segment outputs dynamically.
* **Simulation Tests**:
  + Verified hexadecimal and decimal modes.
  + Tested signed and unsigned numbers.
  + Validated leading zero blanking functionality.

## **System Integration**

* **Platform Designer (Qsys)**:
  + Packaged the IP core with \_hw.tcl and added it to the Qsys project.
  + Connected to the lightweight AXI bus of the HPS.
  + Assigned memory addresses as specified in the lab instructions.
* **Quartus Project**:
  + Integrated the Qsys-generated .qip into the Quartus project.
  + Completed pin assignments for seven-segment outputs and other peripherals.

## **Hardware Testing**

* **Programming**:
  + Generated a .sof file and programmed the DE1-SoC.
* **Verification**:
  + Interacted with the HPS using a terminal emulator (e.g., PuTTY).
  + Validated seven-segment display output for various control register configurations.

# ****Result and Analysis****

* The IP core correctly handled all tested scenarios, including hexadecimal and decimal modes, signed numbers, and leading zero blanking.
* Simulation and hardware testing aligned, confirming the reliability of the design.
* Proper integration with the HPS and Platform Designer ensured seamless operation.

# Design-Discussion-Experimentation

## **Challenges Faced**

* **Simulation Setup**: Initial difficulties in configuring the Python-based test fixture were resolved by reviewing the cocotb documentation.
* **BCD Conversion**: Debugging errors in the to\_bcd function required detailed test cases.

## **Future Improvements**

* Implement additional features like configurable refresh rates for the display.
* Explore optimization techniques to reduce hardware resource usage.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Expected Output** | **Observed Output** | **Result** |
| Hexadecimal display (0x1A2B3C4D) | Correct segments for hex digits. | Segments match input hex values. | Pass |
| Decimal display (12345678) | Correct BCD conversion displayed. | Matches expected BCD conversion. | Pass |
| Signed number (-12345) in decimal mode | Negative signs displayed. | Matches expected output. | Pass |
| Leading zero blanking enabled | Leading zeros not displayed. | Leading zeros blanked correctly. | Pass |
| Magic number read-back (0x12345678) | Magic number matches. | Matches expected value. | Pass |

***Table 1: …..Name***

# Conclusion

Project 4 successfully demonstrated the design, implementation, and integration of a seven-segment display controller using VHDL. The core functionality of the seven\_segment\_agent was developed to process binary inputs, manage control settings, and output the appropriate seven-segment configurations. By leveraging modular design principles and reusable components defined in the seven\_segment\_pkg, the agent effectively supported features such as Decimal Mode, Signed Number Support, and Blank Leading Zeros.

Throughout the project, the functionality was rigorously tested in simulation and hardware. The use of ModelSim enabled precise verification of the agent's behavior, while Quartus Prime ensured seamless integration with FPGA-based systems. Key challenges, including the handling of multiple display configurations and feature encoding, were addressed through well-structured processes like data\_driver, assign\_digit, and concat\_function.

The project culminated in a working implementation capable of driving a physical seven-segment display, accurately rendering numbers with features such as negative signs and zero suppression. The design's flexibility and modularity make it a robust and reusable IP for future digital system designs.

Overall, Project 4 provided a comprehensive learning experience in digital design, from conceptualizing and coding to integrating and debugging in an FPGA environment. It highlighted the importance of modular design, reusable code, and simulation-driven development for successful hardware implementation.

# References

1. University of Massachusetts Lowell. (2024). Advanced Digital System Hardware Design. [Online]. Available: https://lowell.umassonline.net/ [Accessed: Dec. 8, 2024].
2. O. Arias. (2024). “Project 4: System-on-Chip Design” University of Massachusetts Lowell, Assignment, 2024. [Online]. Available: https://lowell.umassonline.net/ [Accessed: Dec. 8, 2024].
3. Intel, "Quartus Prime Lite Edition Documentation," [Online]. Available: https://www.intel.com. [Accessed: Dec. 8, 2024].
4. Intel, "Avalon Interface Specifications," [Online]. Available: https://www.intel.com. [Accessed: Dec. 8, 2024].
5. Terasic, "DE1-SoC User Manual," [Online]. Available: https://www.terasic.com.tw. [Accessed: Dec. 8, 2024].