ECE 310
Project Report: Serial BCD ALU Implementation

Anish

November 26, 2024

Abstract

This report outlines the design, simulation, and synthesis of a Serial Binary-Coded Decimal (BCD) Arithmetic Logic Unit (ALU). The ALU performs addition and subtraction on two 16-bit BCD numbers using Serial-In Parallel-Out (SIPO) and Parallel-In Serial-Out (PISO) registers. The design emphasizes efficient data handling, modularity, and synthesis optimization. Simulation results confirm the functionality and accuracy of the ALU across various test cases, highlighting its ability to process serialized inputs and outputs effectively.

Q2.2 Pen-and-Paper/Digital Architecture and Synthesis Results

The pen-and-paper architecture includes four main components: the SIPO register, the controller, the datapath, and the PISO register. The SIPO collects serial input data, forming a 41-bit parallel value. The controller manages state transitions, including detecting start packets, triggering arithmetic operations, and controlling data serialization. The datapath executes digit-wise BCD addition or subtraction, applying necessary corrections for valid BCD output. Finally, the PISO serializes the computed results for output.

Synthesis results align with the conceptual architecture but include computer-generated optimizations. These optimizations reduce resource usage, such as minimizing registers and mapping arithmetic logic efficiently. The synthesized design mirrors the functionality of the pen-and-paper approach, with differences in layout and resource allocation for hardware efficiency.

Q2.3 Waveform Results and Discussion

Simulated waveforms illustrate key operations, including input loading, start packet detection, arithmetic processing, and result serialization. Test cases such as ‘3627 + 1287 = 4914‘ and ‘637 - 459 = 178‘ validate addition and subtraction operations, while edge cases like ‘9999 + 1‘ test carry handling. Key points in the waveforms include detecting start packets (‘0x5A‘), initiating arithmetic based on operation codes, and serializing results with the correct timing.

The ALU accurately processes continuous streams and resumes correctly after interruptions, as demonstrated in the waveforms. Timing analysis confirms synchronized operation, ensuring reliable processing of serialized inputs and outputs.

Q2.4 Code Discussion

The ALU is implemented using a modular design with clearly defined components: 1. **SIPO Register**: Serially collects input data and assembles a 41-bit packet. 2. **Controller**: Manages state transitions, including computation initiation and output serialization. 3. **Datapath**: Includes ‘BCDadd‘ and ‘BCDsub‘ modules for digit-wise BCD arithmetic with corrections. 4. **PISO Register**: Serializes and outputs the 28-bit result packet.

The design emphasizes pipelining, enabling continuous operation without interruptions. The controller coordinates the interaction between modules, ensuring data flows seamlessly from input to output. Nested modules simplify arithmetic operations, promoting reusability and clarity. Resource usage is minimized through synthesis optimizations, particularly for registers and arithmetic logic.

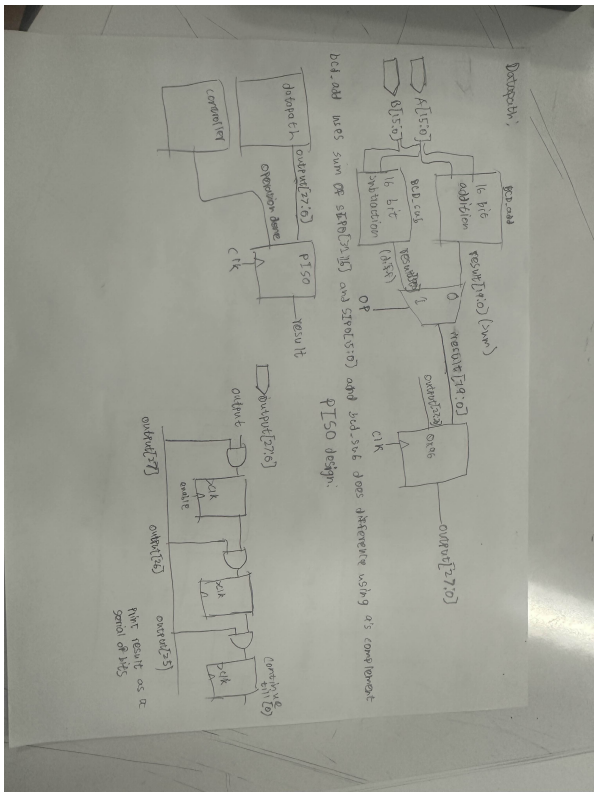

Figure 1: Datapath and PISO modules

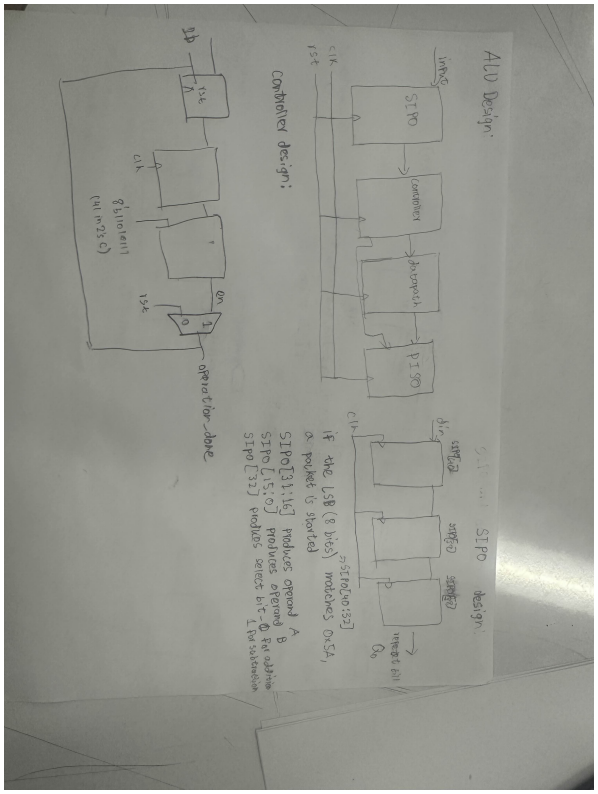

Figure 2: ALU desgin along with SIPO and controller modules

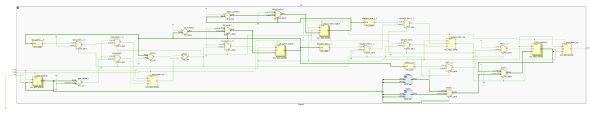

Figure 3: Synthesized Design

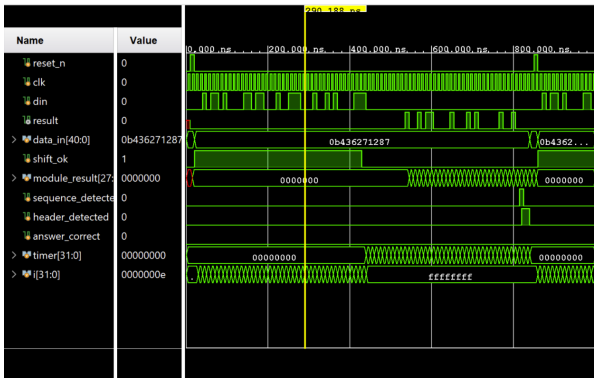

Figure 4: Waveform with test cases

Conclusion

The Serial BCD ALU successfully implements addition and subtraction for 16-bit BCD operands, leveraging efficient serialization techniques for input and output handling. The design balances modularity, resource efficiency, and functional accuracy, as confirmed by simulations and synthesis results.

Future enhancements could include: 1. Support for variable operand lengths. 2. Additional arithmetic operations like multiplication and division. 3. Enhanced error detection and handling mechanisms.

This project demonstrates the importance of modular design and synthesis in developing robust digital systems.