ECE 128

Final Project: Calculator

1. Objectives

The objective of this project was to design and implement a 4-bit calculator on the FPGA board that supports a total eight operations. These operations include arithmetic functions (addition, subtraction, multiplication, and division) and logical functions (AND, OR, XOR, and NOT). The system was developed using a modular hierarchy in Verilog, with separate modules for calculator logic (ALU), a BCD converter, and a seven-segment display to enable real time output. A top level module integrated all submodules into a single functional design. A testbench was created to simulate and verify all calculator operations as well as the corresponding seven-segment display outputs. Finally, the completed design is validated through hardware testing by demonstrating a fully functional calculator operating on the FPGA board. Overall, the project provided insight into the bridge between digital logic theory and practice by applying fundamental digital logic concepts to the design of a functional calculator on an FPGA board.

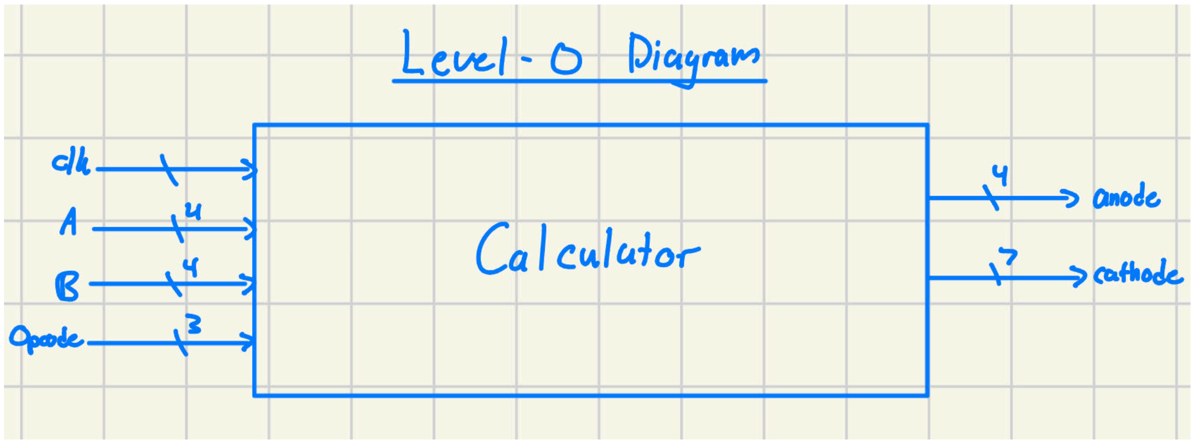
2. Introduction

Calculators are one of the earliest and most widespread applications of digital electronics. The core of calculators rely on hardware to perform arithmetic and logical operations. Whether in the context of the common handheld calculators or an internal calculator implemented within a modern processor, the fundamental building block of the computations is the ALU, or arithmetic logical unit. The ALU is a combinational digital circuit that performs arithmetic and logical operations on binary numbers. The inputs of an ALU are the operands, the data to be operated on, and the opcode which indicates the operation to be performed.

The direction of design for this project centered on implementing a simplified calculator to the FPGA board by following the same principles used in real digital systems. The first step was to identify a modular hierarchy for our system to separate the ALU, binary to BCD converter, and the seven-segment driver modules. As described, the ALU is the functioning component of the computational logic. The ALU module takes two 4-bit inputs, the operands A and B, along with the 3-bit opcode, to indicate one of the eight operations to be performed. Through combinational logic, the ALU assigns the output, result, of the module to opcode designated operation. The four logical operations, AND, OR, XOR, and NOT, are computed using the built-in bitwise operators. The four arithmetic operations, addition, subtraction, multiplication, and division, are derived from their respective submodules. The addition operation is computed using a Ripple Carry Adder (RCA). The RCA produces a 4-bit sum, with a carry output bit, through a cascaded calling of four 1-bit full adders with the carry output from each full adder connecting to the carry input of the following full adder. A combinational multiplier computes the multiplication operation. A combinational multiplier generates the 8-bit output in a single step, based solely on the inputs via generating partial products and shifting the product incrementally. The subtraction operation is computed via a subtractor module that inverts the bits of one operand and utilizes the RCA module to produce a 4-bit result with a "borrow bit." Finally, for the last arithmetic operation, the division module is designed to repeatedly subtract the operands and increment the quotient until the remainder is smaller than the divisor through a combinational loop.

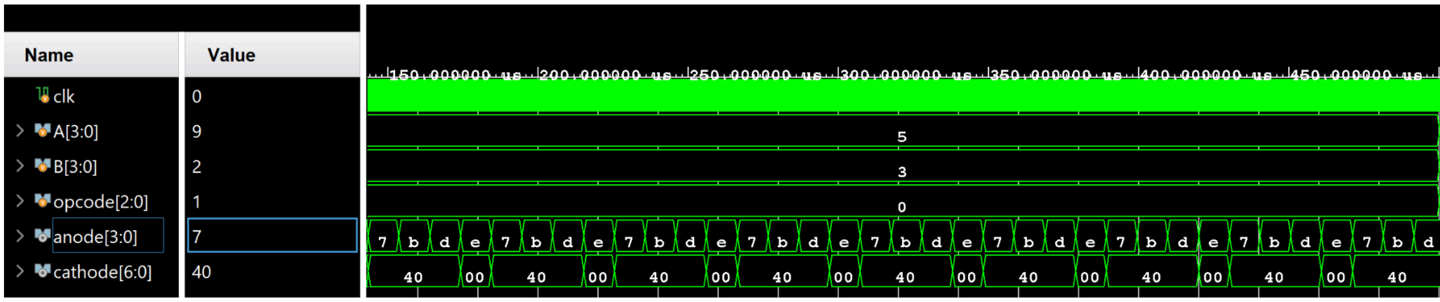
The remaining modules for our system design, excluding the top module to integrate the modules, are the binary to BCD converter and the seven-segment driver to ensure the displayed values of the FPGA's seven-segment are readable. The binary to BCD converter takes a 12-bit binary input and utilizes the double dabble algorithm in a sequential logic process to output a 16-bit value in binary coded decimal (BCD). Once completed, the converter relays a ready signal to the top module to begin the process of generating the correct anodes and cathodes for the seven-segment driver. The seven-segment driver module is able to display to multiple segments of the display using two submodules. The anode generator utilizes sequential logic to generate the respective anode for display based on the BCD value. The cathode module uses the first four bits of the BCD output from the anode generator to indicate the correct cathodes to display the value. Finally, a top module was implemented to drive the respective inputs, outputs, and ready signals to each module. Additionally, the top module aided in concatenating the bits from the ALU such that the values outputted would be of the correct register size when driven to the BCD converter.

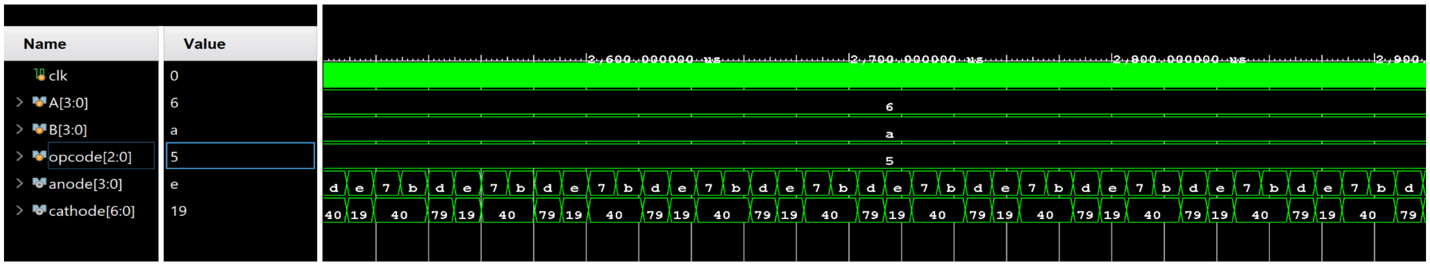
3. **Results**
    a. **Level-0 Block Diagram**



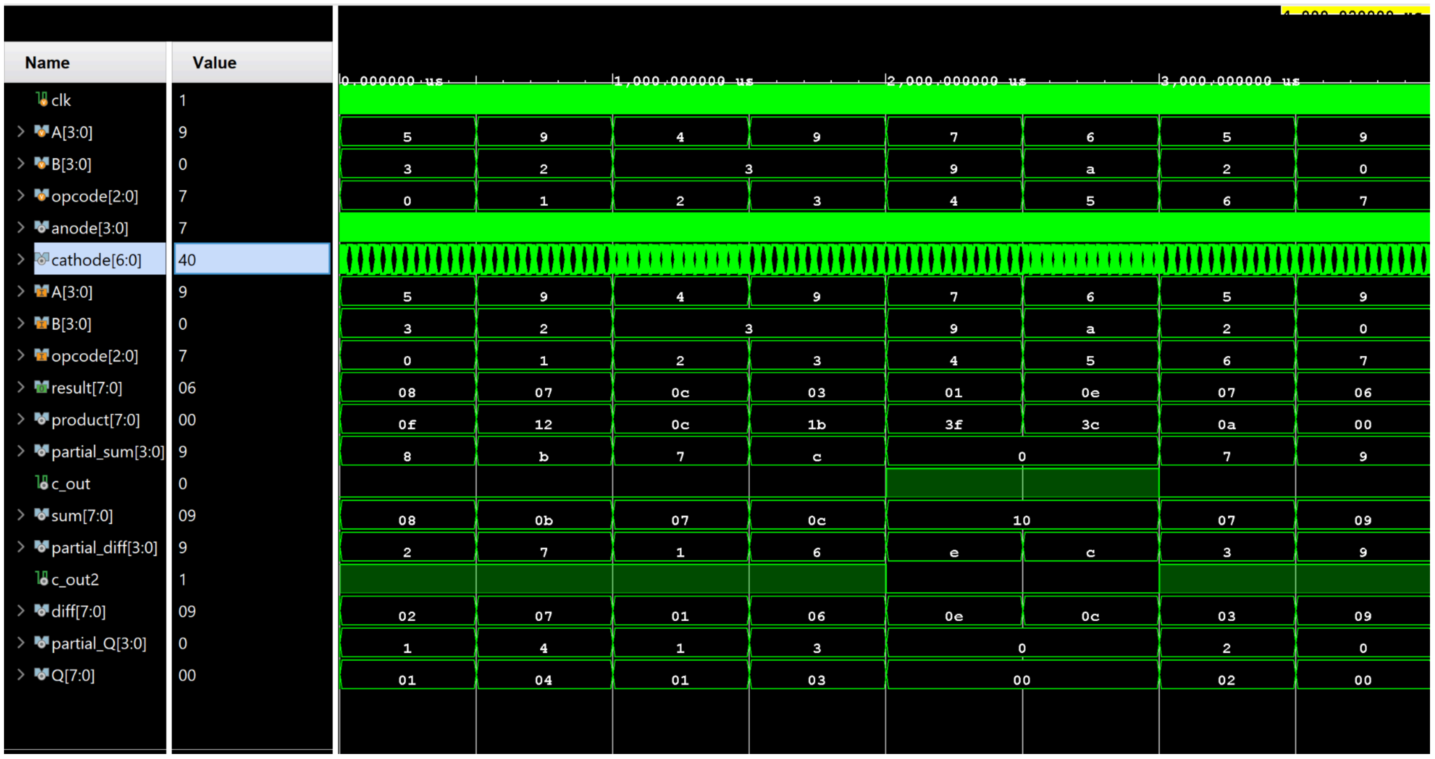
b. Simulation Waveform

Top Module Simulation Waveforms



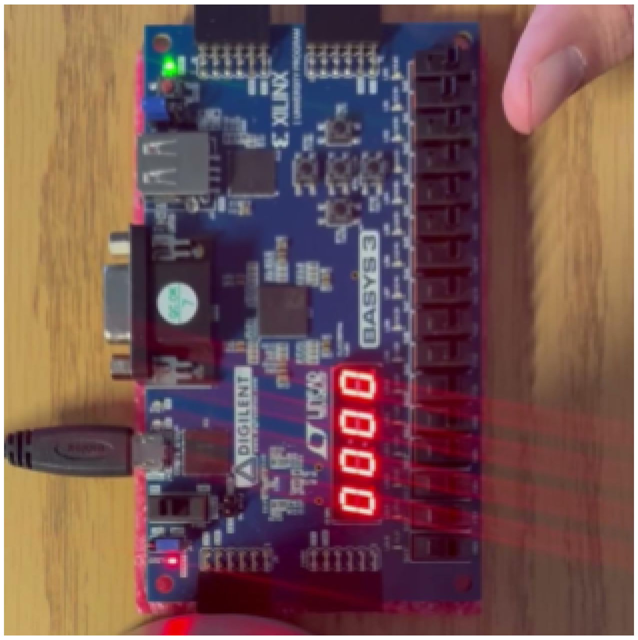


Simulation Waveform (with ALU signals)

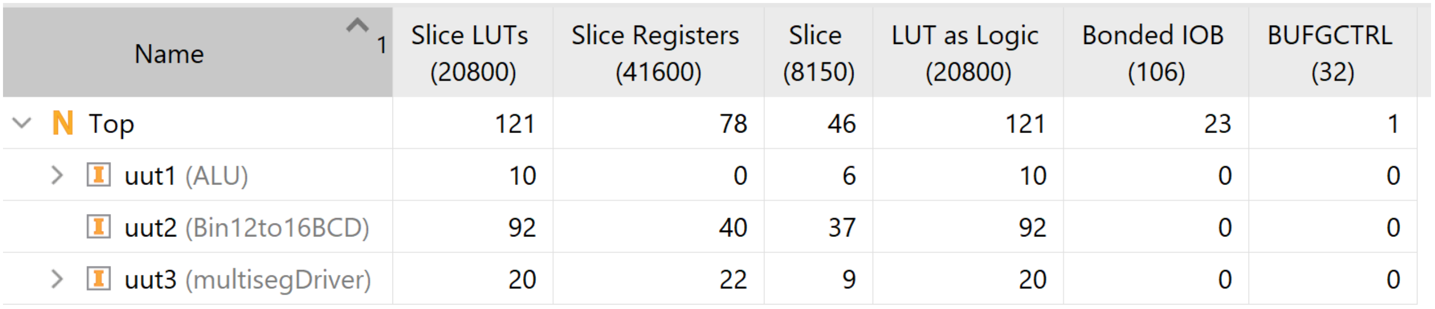


c. FPGA Board Implementation

Click image below for link to demonstration video:



4. Area Utilization



Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	Bonded IOB (106)	BUFGCTRL (32)
Top	121	78	46	121	23	1
uut1 (ALU)	10	0	6	10	0	0
uut2 (Bin12to16BCD)	92	40	37	92	0	0
uut3 (multisegDriver)	20	22	9	20	0	0

The design only used a small fraction of FPGA resources, highlighting the effectiveness of our design. Specifically, only 0.58% of the Look-up Tables, 0.19% of the Flip-flops, and 21.7% of the IO board components were utilized in the design implementation. With the efficient usage, it is apparent that additional calculator operations could be added to the design, along with general future expansion on the design.

5. Conclusion

In conclusion of our final project, we were able to successfully design and implement a fully functional 4-bit calculator on the FPGA board. The calculator system was able to support a total of eight arithmetic and logical operations. The system was designed using a modular hierarchical design in Verilog to produce a cohesive hardware design. The foundation of our computational logic was implemented through our ALU module, through which our arithmetic operations were calculated via their respective submodules. Furthermore, the incorporation of our BCD converter and seven-segment driver modules allowed for the FPGA board to display human readable and accurate values. Additionally, we were able to verify accuracy of the calculator system prior to implementation, through creating a Testbench and simulating various inputs for each operation. Finally, we synthesized and implemented our design before generating a bitstream and programming our FPGA board. Through in-depth testing, we were able to verify the effectiveness and accuracy of our design on the hardware level.

Throughout our project design, we faced some obstacles, but through addressing these challenges, the project reinforced key takeaways about digital logic design. For instance, in the ALU module, not all of the operations provided results with the same bit length, and when the values were passed to the BCD converter, errors would appear in the system. Specifically, the NOT operation, which acts solely on the first 4-bit input, would invert the other “empty” bits. To fix this issue, we ensured that all inputs and outputs of the system were appropriately concatenated with zero bits throughout all modules. Another challenge we encountered was with the simulation of our design and failing to get output values for the cathode and anode outputs. Through a series of debugging practices, we identified the issue as the short clock cycles we implemented for each test case in the testbench. We found that the sequential behavior of the BCD converter and seven-segment display modules needed an extended amount of clock cycles in order to effectively output their values. To amend the issue, we added a 0.5ms delay between each clock cycle. This additional step allowed us to improve our understanding of how the combinational modules interacted with the sequential logic modules. Overall, this project integrated core digital design concepts from throughout the course, while highlighting the complete engineering design process.

6. Contribution Chart:

	Frank Tamburro	Alex Hume
Prelab & Initial Research	X	X
Writing Code	X	X

Analyzing Result	X	X
Writing Report/Presentation	X	X

7. **GitHub Link: https://github.com/fjt227-debug/ECE-128-Final-Project**