This is the design phase of building an 8-bit computer using an Altera FPGA D2 board and exploration of the future potential use of this project

Computer Design: 8-bit CPU

Faculty Advisor: Jacob Eapen

Zongo, Grace O

This honor project idea was inspired by the 8-bit computer built on a breadboard built by Ben Eater. I watched his videos countless times when I was back in Burkina Faso but lacked the means to realize his project. However, after getting to the USA, I searched for the components to build his 8-bit computer. After building the first module called “Clock Module”, I realized that they could cost a lot in time and resources for a student like me. Also, making the computer on breadboards with logic chips seemed like it would take up a ton of space and have issues with bad connections and troubleshooting. Even though it was a great experience for troubleshooting, it did not exactly push me toward my goal of understanding how computers work at the core. This led me to look for other ways to satisfy my curiosity: I could make the computer using an FPGA and software, and this would give me better skills that suit the technology nowadays instead of using outdated breadboards (even though they are great for learning purposes).

Even though I used an FPGA, I purposely didn't use the complicated programming language called HDL. Instead, I built the computer using simple logic gates. I also didn't use a lot of the pre-made designs in the Quartus program (that's the software for working with FPGAs) for the sole purpose of activating my brain to build and understand every single component that goes into the design of a computer.

Now, let’s dive into building our first computer.

1. The SR Latch: How do computers store single bits:

An SR latch, also known as a Set-Reset latch or a Flip-Flop latch, is a simple electronic circuit element that functions as a basic memory cell. It's made up of two cross-coupled NAND gates (or NOR gates) and has two inputs: Set (S) and Reset (R). The outputs of the latch are Q (the normal output) and Q' (the complemented output).

A diagram of a computer

Description automatically generated**SR Latch Truth Table**

Figure 1: SR latch made of AND gates.

|  |  |  |  |
| --- | --- | --- | --- |
| S | R | Q | NQ |
| 0 | 0 | / | / |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

SR latches are used in various digital circuits for simple memory storage, control, and synchronization purposes. However, they have a major drawback due to the potential of both inputs being '1' at the same time, causing unpredictable behavior. To address this limitation, more advanced flip-flops, like the D flip-flop and JK flip-flop, were developed.

These flip-flops have specific clock inputs that control when the inputs are allowed to affect the outputs, providing better control and avoiding the race condition issue. Despite their limitations, SR latches still provide a foundational understanding of digital logic circuits and are used in educational contexts to teach basic concepts of memory elements and digital design.

A diagram of a circuit diagram

Description automatically generated with medium confidence

Figure 2: D-flip-flop made with and SR latch

We can pair two D-flip-flop together to save the set signal only at the raising clock.

1. Registers: Were data is temporarily saved

Registers are fundamental digital electronic components used to store and manipulate data within a microprocessor or other digital systems. They are made of flip flops capable of storing the data of the input momentarily as described above. Registers are crucial for various tasks in a central processing unit (CPU) and other parts of a computer system. They are commonly used for data storage, arithmetic and logic operations, input and output operations, control unit operations, temporary storage, etc.

A diagram of a computer circuit

Description automatically generated

Figure 3: Snapshot of a part of a Register made of D-flip-flop. Since we are building an 8-bit computer, our registers need to be able to manage an 8-bit data, hence the use of eight D-flip-flops for a single register. Our register has 8 input pins and 8 output pins. Moreover, we added a system of logic gates that will allow us to read, write, and clear the data stored in the register.

1. How will the computer do the little operations like adding and subtracting: the Arithmetic Logic Unit (ALU)

All computers need to compute the data given to them. Even if the purpose of the computer is not to know how much is one plus one, it still needs to compute the data given and identify what are the instructions it has been given. For our computer, an Arithmetic Logic Unit made of eight will be used to compute the input data. An adder is a digital circuit used to perform addition operations on binary numbers. It's a fundamental building block in digital design and is used in various applications within computers and digital systems. Adders are crucial for performing arithmetic calculations, data manipulation, and logical operations. There are different types of adders, each designed to fulfill specific requirements. Also, adders come in various forms, including half adders, full adders, ripple carry adders, etc., each offering different trade-offs between speed, complexity, and efficiency. They are used across a wide range of digital systems, from simple calculators to very complex CPUs and digital signal processors.

A diagram of a block diagram

Description automatically generated

Figure 4: This is a 1-bit adder that we built to compute the data stored in the registers. The adder has two inputs pins and one output pin plus a carry pin (A + B = SUM; and we carry C\_out). The Carry\_in pin (C\_in) is here so that we can add two or three or four (...) adders to make an 8-bit adder

A diagram of a computer

Description automatically generated

Figure 5: The ALU is made of eight 1-bit adders. Therefore, it takes two 8-bit data, add or substract them, and gives an 8-bit ouput (sum) plus the carry\_out pin.

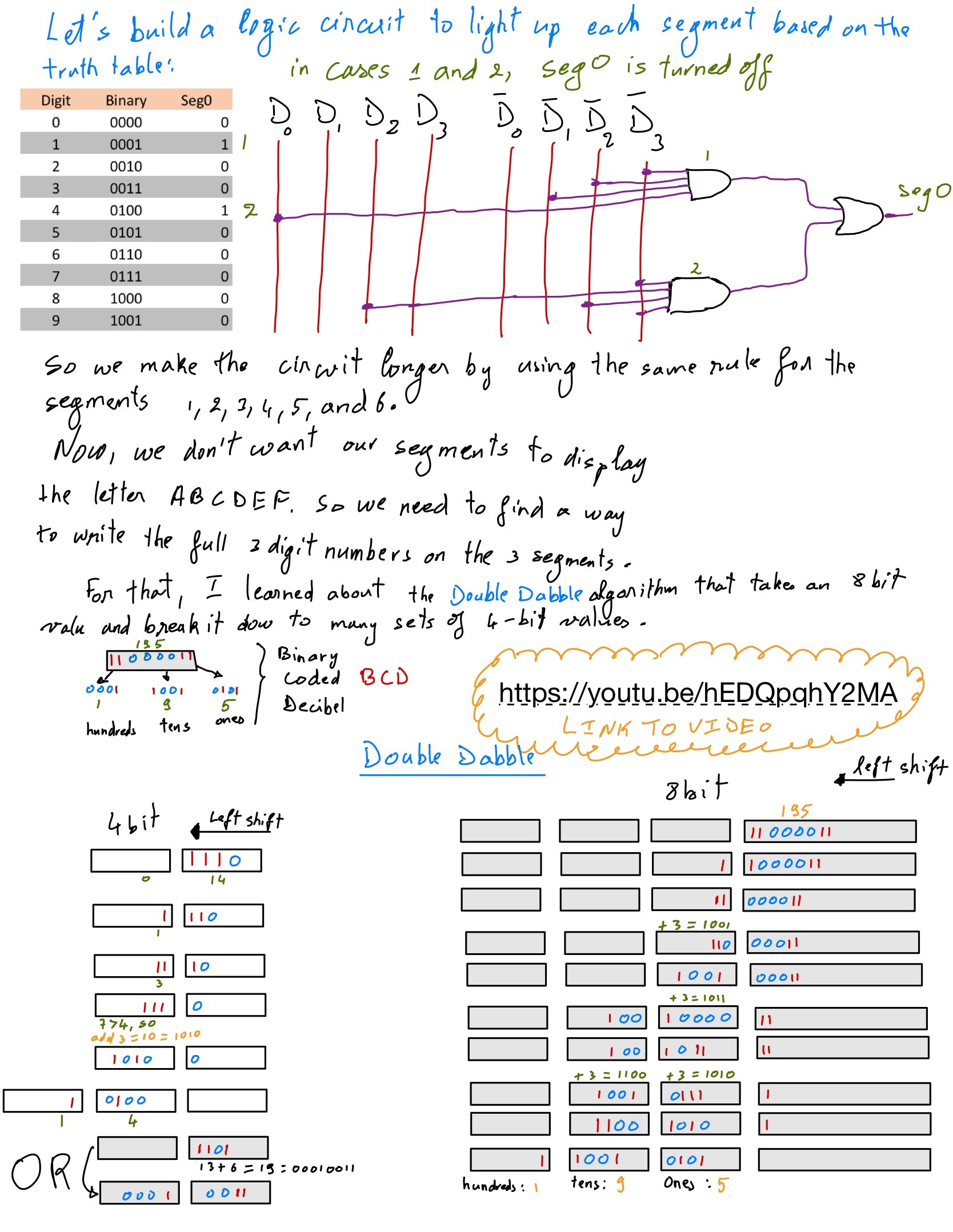
1. Testing the ALU

After building the ALU, we wanted to test it and make sure it was working as it was supposed to. We used the switches of the FPGA as input, and the LEDs as outputs. However, we would love to see the computing happening by using the seven segments displays. The testing process is explained below.

A computer screen with numbers and text

Description automatically generated





A diagram of a computer

Description automatically generated

Figure 6: Logic of the Dabble Algorithm

A diagram of a computer network

Description automatically generated

Figure 7: This schematic allows us to display a number with 3 digits

A computer screen shot of a computer

Description automatically generated

Figure 8: Final schematic of ALU connected to the segment displays, allowing us to clearly see the operations being made by the computer

1. What we will do in the future:

Now that we have built the ALU and the segment display chip, we will begin building the memory chips. This includes the RAM and the memory address, all made of the registers that we already built previously. After building the memory chips, we will assemble the parts of the 8-bit computer and put together a way to program the RAM. Then, a simulation of the computer will take place to prove its functioning, and we will identify different uses for this 8-bit computer.