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: Part II

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. In the Summer, we worked on building some key components such as memory units (registers and RAM), the Arithmetic Logic Unit, and the Binary to decimal converter for the output. I continued the project this fall by completing the build of the RAM and by designing the Control Unit. It was the most challenging, yet enjoyable part to design. Not only did I have to discover how data flows in computers, but I also learned to decode an instruction into many steps. The details of the built will be explained bellow (A reminder that this project was entirely done with block design rather than using VHDL):

**Von Neumann Architecture: Essential parts of the computer**

The Von Neumann architecture describes the basic structure of a computer system and consists of a few key components. This architecture follows an execution model that is sequential, meaning that the instructions are fetched (collected) from memory one at a time, and executed in a step-by-step fashion. It has been the basis for the design of most general-purpose computers for decades. The key components of this model are:

* The Memory: In a Von Neumann architecture, both data and program instructions are stored in the same memory space. This memory is typically divided into two parts: data memory (for storing data) and program memory (for storing instructions).
* Central Processing Unit (CPU): The CPU is the brain of the computer and is responsible for executing instructions stored in memory. The CPU of the Von Neumann architecture uses two main units that allow data processing: The Arithmetic Logic Unit (ALU) which is responsible for performing arithmetic and logical operations like addition, subtraction, multiplication, and comparison; The Control Unit which manages the execution of instructions, ensuring that the right actions are performed at each step of the execution.
* Input and Output (I/O): This component facilitates communication between the computer and external devices. Input is a way to store instructions and data in the computer while the output is how the computer returns the result of the instruction it executed.

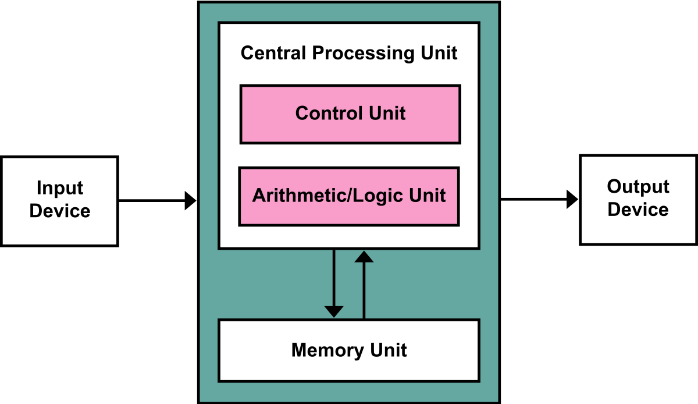
****

Figure : Von Neumann Architecture

In the first part of this project, we successfully built and tested registers (the core of the memory for our computer), the Binary-to-Decimal converter (serving as output), and many other components important to the integrity of our 8-bit computer such as the counters and the ALU (Key Component of the brain of the computer).

**Instruction Register:**

Before we start explaining how we designed the major components of the Von Neumann Architecture, we need to explain the use of the instruction register as it is a special memory unit where the next instruction to be executed is stored. As the Memory Address Register (MAR) and the output register, the instruction register holds the current instruction and sends it to the control unit for decoding. But unlike the others, it takes an 8-bits data from the bus as input and splits it into two parts: 4 bits for the instruction and 4 bits for the memory address. It sends the instruction to the control unit and the address into the Memory Address Register through the bus.

A diagram of a computer

Description automatically generated

Figure : Output [Q4..Q8] is instruction. Output [Q0..Q3] is Address

**The Microinstruction Counter**

The microinstruction counter is a counter like the Program Counter (PC) that goes through the individual steps needed to perform an instruction. Indeed, for each instruction, there are a set of actions that need to be performed in the right order because all the data cannot flow through the bus at the same time. The counter allows the computer to know what steps it is at in executing a specific instruction.

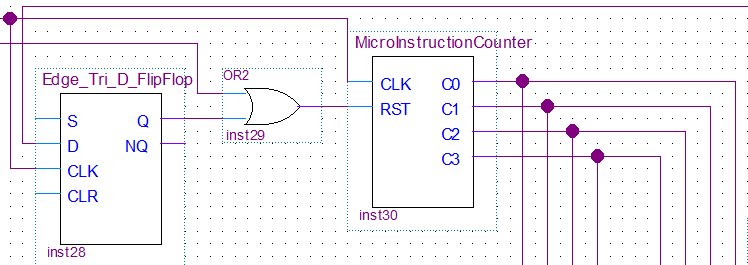


Figure : Micro Instruction Counter is a counter that will tell our control unit which step it is at to finish executing the instruction.

**Control Unit:**

The main thing about the control unit is its ability to take in an instruction and break it into many steps. The next images show how the instructions are broken down into the multiple steps (microinstruction) needed to run them. There are three decoders. The first one decodes the instruction (from the instruction register input), the second one chooses the step (from the microinstruction counter input), and the third one uses the flags for instructions that need them, such as Jump if Zero and Jump if Carry.

A close-up of a diagram

Description automatically generated

Together, let’s decode the specific actions that our computer needs to take to load the number 5 (stored in the RAM) and save it in the A-Register. We will call this instruction: LDA (short for load), and we will associate the binary code 0001 to this instruction. So, if the Control Unit receives the binary code 0001, it will set the LDA output of the Op-Code element to 1 (our computer knows that 0001 = LDA). Then, each time the clock ticks, the computer does a specific task that will allow us to read the specific address where the number “5” is stored, and move it to Register A. Each step is described in the table below:

|  |  |  |
| --- | --- | --- |
| Micro Counter | Step Number | Actions Taken |
| 0000 | 1 | PC Output ON, MAR input ON |
| 0001 | 2 | RAM Output ON, IR input ON, PC Increments |
| 0010 | 3 | IR Output ON, MAR input ON |
| 0011 | 4 | RAM Output ON, A-Register input ON |
| 0100 | 5 | RESET Microinstruction Counter to ZERO |

A close-up of a paper

Description automatically generated

Figure : Thought Process explaining the necessary steps behind other instructions.

A diagram of a machine

Description automatically generated

Figure : Animation Explaining How the LDA instruction is Executed

A table of test results

Description automatically generated with medium confidence

A diagram of a test

Description automatically generated with medium confidence

Figure : Alongside with providing the binary code for each instruction, this detailed table illustrates the necessary step for almost all the instruction that this computer can execute (Note: This computer can have a total of 15 instructions since each instruction only have 4 bits.)

**TESTING PROCESS:**

A diagram of a computer circuit

Description automatically generated with medium confidence

Figure : After building comes troubleshooting. It was an intensive process that required extreme patience and problem-solving skills. But when testing the ADD Instruction and other instructions, all the necessary steps were well executed. We used the SignalTap Analyzer to view an interactive state of all the components in the final design. It facilitated pinpointing the errors and solving them.

A screenshot of a computer

Description automatically generated

Figure : SignalTap Analysis

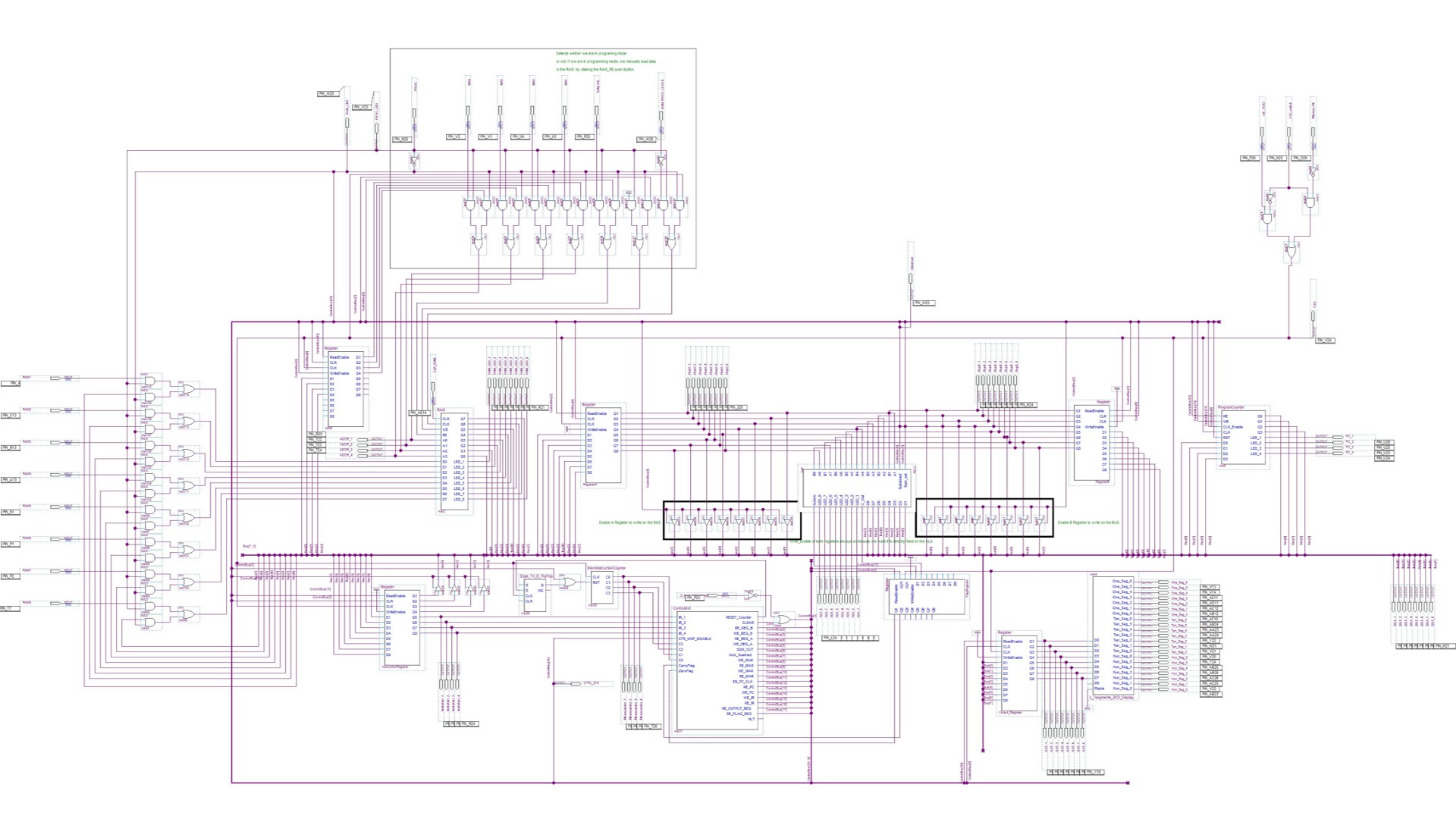


Figure : Fully functional Computer with all the parts connected together



Figure : Video demonstrating the computer executing 15+14=29 on the FPGA Board (Double click to play)



RAM CLK

Figure : Legend of how to use the DE2 Altera FPGA board to test that our computer was working as it is supposed to.

**HOW TO USE THE COMPUTER WITH THE DE2 BOARD:**

1. Put the Computer in Program Mode.

* Click “CLR Computer” to clear the messy data randomly loaded.
* “Clear RAM” switch up then down (to clear randomly loaded data)
* “Code RAM”, “PROG” and “CLK mode” up to code the RAM.

2. LOAD instruction and data in the desired address of the RAM

* In address 1, put LDA from address 15: 0001 0001111
* Click “RAM CLK”
* In address 2, Output the data loaded: 0010 11100000

RAM ADDRESS

* Click “RAM CLK”
* In address 3, put ADD to data in address 14: 0011 00101110
* Click “RAM CLK”
* In address 4, Output the result: 0100 11100000

RAM DATA

* Click “RAM CLK”
* In address 15, store the number 20: 1111 00010100
* Click “RAM CLK”
* In address 11, store the number 23: 1110 00010111
* Click “RAM CLK”

3. Put the Computer back in RUN Mode.

* Click “CLR computer” Computer to clear the messy data randomly loaded.
* “Code RAM” and “PROG” down go back in RUN mode.

4. Start executing your program by clicking on the “Manual CLK” slowly