CLJD

Updated Report

<2019-11-21>

Updated Description

The Arithmetic Logic Unit, or ALU as descripted in the draft description, still fulfils the requirements of taking a set of data and instruction inputs to perform different arithmetic operations. The ALU calculates these inputs, as if it were a calculator. The ALU functions fulfilled are add, subtract, multiply, and divide. The size of the numbers being calculated are 16 bit integers, thus we have carry-over and overflow errors that the ALU handles. The ALU reflected from this report can perform the following basic logic functions: ANDing, ORing, XORing, NOTing, NANDing, NORing, and XNORing.

**The obvious but unforeseen carry-over error that was added is handling carry-over in the multiply module.** Another unforeseen error handle which was not explicitly stated in the first progress report is **handling a divide-by-zero error**. The divide module currently reflects this error handle. **We have also defined a decoder, multiplexer, and a state machine** which was not originally defined in the draft description.

Our development workspaces differ but we are still using Verilog version 10.1 as the central language.

The overall purpose of the ALU is to perform simple calculations via a combinational digital logic circuit, and to follow the requirements specifications listed in Dr. Becker’s project description.

Updated Report

Updated Member Task List

Draft Member Tasks

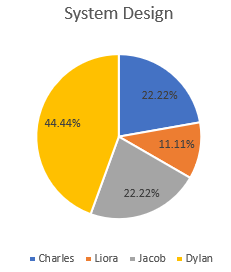
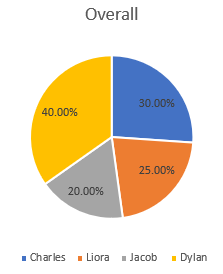
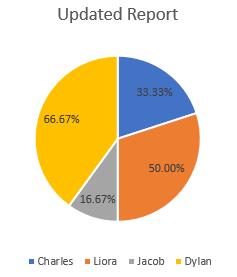
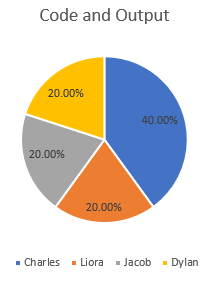
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Charles** | **Liora** | **Jacob** | **Dylan** |
| Updated Report | | | | |
| Title Page |  |  |  |  |
| Updated Report |  |  |  |  |
| Updated Description |  |  |  |  |
| Updated Member Task List |  |  |  |  |
| Updated Software Discovery |  |  |  |  |
| Updated Participation  Census |  |  |  |  |
| System Design | | | | |
| Title Page |  |  |  |  |
| Parts List |  |  |  |  |
| Input List |  |  |  |  |
| Output List |  |  |  |  |
| Interface List |  |  |  |  |
| Module List |  |  |  |  |
| Mode List |  |  |  |  |
| Circuit Diagram |  |  |  |  |
| State Machine Diagram |  |  |  |  |
| Code and Output | | | | |
| Add |  |  |  |  |
| Subtract |  |  |  |  |
| Multiply |  |  |  |  |
| Divide |  |  |  |  |
| Bit Shift |  |  |  |  |

Updated Software Discovery

We are no longer choosing to use Logisim as the main circuit development software tool for our project. Although Logisim serves well to automatically generate circuits and k-maps based on Boolean logic and truth tables, we felt using draw.io would be a better fit for our team. Draw.io is also an open source technology used by millions of people, but what ultimately makes draw.io better and more efficient than Logisim is its ability to enable team members to work on projects collectively at the same time from different machines. We found it difficult to find time in our schedules to meet face to face outside of class, so a feature like this held more value over any other seen in other open source softwares currently on the market. Our draw.io circuit diagram is linked to our Google drives, which serves as a safe backup, and is also how we share and work on the circuit diagram between team members.

Furthermore, unlike Logisim which was last updated on October 11, 2014, draw.io is continuously being updated. The current version of draw.io is 12.2.8 which was updated on November 21, 2019. Draw.io is a very current form of open source software that is being widely used by many, that with its continuous updates and versatility in diagrams, serves as a useful tool to be familiar with.

In terms of Verilog, we are still using iVerilog version 10.1 for our project.

 Updated Participation Census