
Cohort Schmohort

Progress Report

October 15, 2019

Draft Description

In this project, we will create an Arithmetic Logic Unit with full datapath. Its use will be to exhibit functionality of all basic logical operations, and a series of arithmetic operations; this will display to the TA and professor the Cohort's mastery of necessary topics in the Digital Logic course. The ALU provides fundamental, essential arithmetic functionalities that are used in many circuits and devices. It will take a set of data inputs and a set of instruction inputs to perform an operation. A good example for a simple use of an ALU is for arithmetic calculations. The ALU will be designed to perform AND, OR, NOT, XOR, NAND, NOR, XNOR, and BUFFER logical operations to achieve Add, Subtract, and Multiply arithmetic operations. Moreover the ALU will be designed to handle left and right bit shifts.

Four main logic functions:

- AND two inputs
- OR two inputs
- XOR two inputs
- NOT one input

Detects errors:

→ Overflow

→ Divide By Zero

The ALU will take 2 8 bit integers for inputs and 1 2 bit input to determine the operation that will be executed. These operations, mentioned above, will be either addition, subtraction, or multiplication. The result of the operations performed will be saved into the memory register of the ALU.

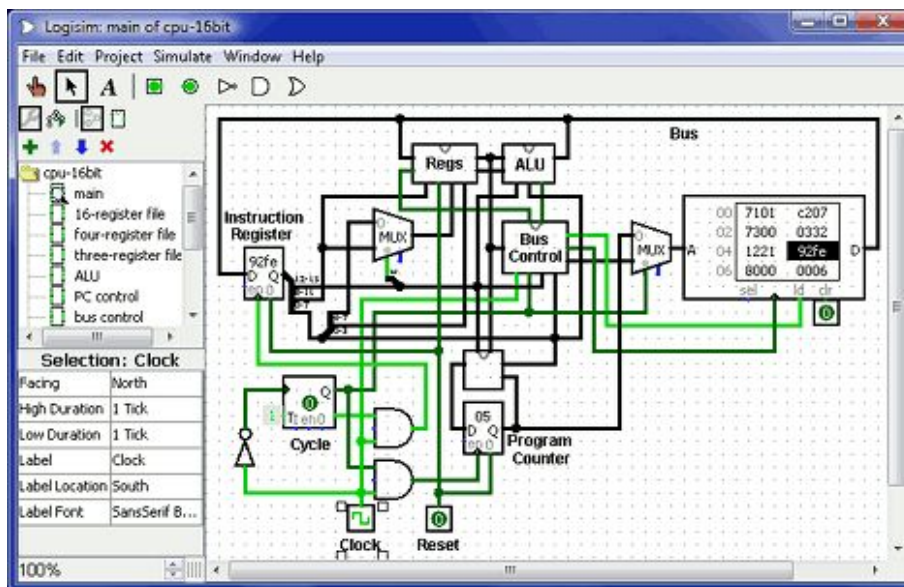
Draft Member Tasks

Name	Delegated Tasks
Jethjera Silasant (JJ) - jxs147030	Software Discovery AND/OR gate diagrams and Verilog Documentation of Inputs & Outputs Overflow Detection Logic
Jonathan Kim - jxk155930	Software Discovery NOR/XNOR gate diagrams and Verilog ADD/SUBTRACT function implementation OpMode Selection logic Documentation of Interface
Clayton Allen - cab160030	Draft Description NOT/XOR/NAND gate diagrams and Verilog OpMode Selection State Diagram State Machine Logic
Zhengyuan Wang - zxw170010	Draft Description BUFFER gate diagram and Verilog MULTIPLY function implementation Documentation of States, Modules, & Registers Divide by Zero Detection Logic
Tristen Even - tge160130	Organization of Deliverables Participation Census Final Circuit Diagram State Machine Logic Error Detection Logic Code Refactoring

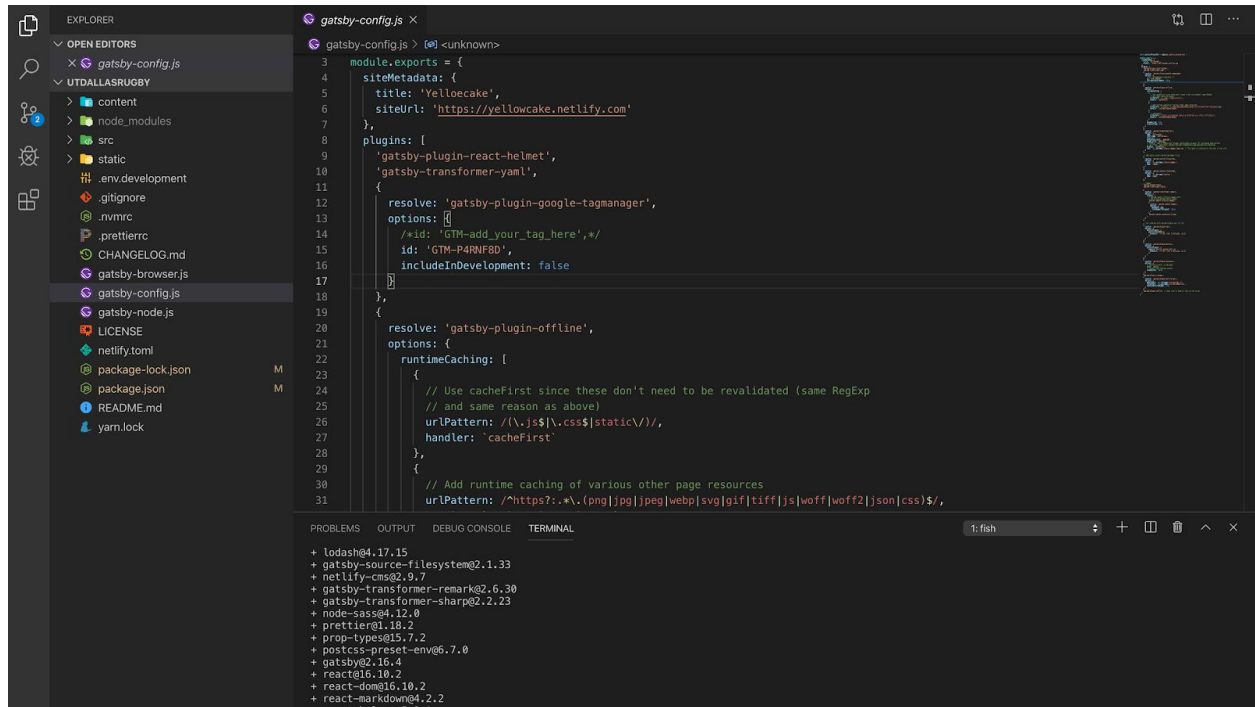
Draft Software Discovery

The software(s) that will be used for the project are *Logisim* for the drawing of the ALU, *VSCo* for writing the code, *iVerilog* for compilation and testing of code, and GitHub for version control. These software(s) are open source and easy to use. Moreover, they offer many online resources to better understand their use and implementation. *Logisim* simulates connections between circuit components and if inputs are entered incorrectly, it allows a way to check the work. For example, if one connection for the selection of the multiplexer is not written correctly, the program throws an error message such as “incorrect amount of bits.” This helps with visualization of the components we will be implementing, and generic troubleshooting during the initial phases of design.

<http://www.cburch.com/logisim/>



VS Code will be used for writing the verilog code. It is a free and easy to use tool that also offers a Verilog support extension to allow better code design and troubleshooting. <https://code.visualstudio.com/download>



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a file tree for a project named 'UTDALLASRUGBY'. The file 'gatsby-config.js' is selected and open in the main editor. The code in the editor is a Gatsby configuration file with the following content:

```
3 module.exports = {
4   siteMetadata: {
5     title: 'Yellowcake',
6     siteUrl: 'https://yellowcake.netlify.com'
7   },
8   plugins: [
9     'gatsby-plugin-react-helmet',
10    'gatsby-transformer-yaml',
11    {
12      resolve: 'gatsby-plugin-google-tagmanager',
13      options: {
14        /*id: 'GTM-add_your_tag_here',*/
15        id: 'GTM-P4RNF8D',
16        includeInDevelopment: false
17      }
18    },
19    {
20      resolve: 'gatsby-plugin-offline',
21      options: {
22        runtimeCaching: [
23          {
24            /* Use cacheFirst since these don't need to be revalidated (same RegExp
25             * and same reason as above)
26             */
27            urlPattern: /^(\.js|\.css)$static$/,
28            handler: 'cacheFirst'
29          },
30          /* Add runtime caching of various other page resources
31           */
32          urlPattern: /^https?:.*\.(png|jpg|jpeg|webp|svg|gif|tiff|j|s|woff|woff2|json|css)$/,
33        ]
34      }
35    }
36  ]
37 }
```

At the bottom of the editor, the TERMINAL panel is open, showing a list of installed dependencies:

```
+ lodash@4.17.15
+ gatsby-source-filesystem@2.1.33
+ netlify-cms@2.9.7
+ gatsby-transformer-remark@2.6.30
+ gatsby-transformer-sharp@2.2.23
+ node-sass@4.12.0
+ prettier@1.19.2
+ prop-types@15.7.2
+ postcss-preset-env@6.7.0
+ gatsby@2.16.4
+ react@16.10.2
+ react-dom@16.10.2
+ react-markdown@4.2.2
```

iVerilog compiles the verilog code written. The way that we will use the compiler is to fix the errors in our work, and also create the verilog code for the ALU. This will require coding experience, debugging, and designing the ALU. As for VsCode (Visual Studio Code), it allows linting for verilog. Therefore these will show possible errors or spelling errors without having to buy an actual Verilog component tool. <http://iverilog.icarus.com/>

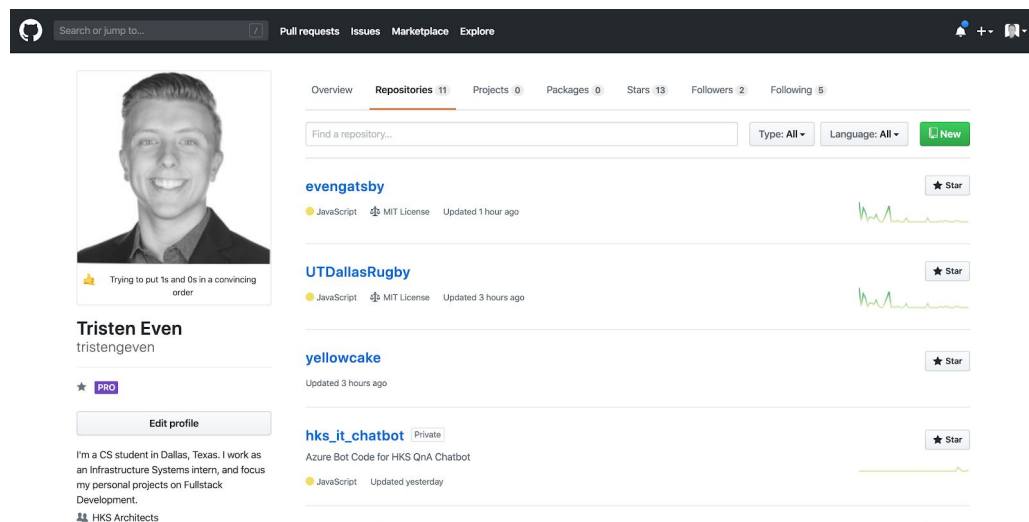
```
Command Prompt

C:\iverilog\bin>iverilog -o mydsign mux2.v mux2tb.v

C:\iverilog\bin>vvp mydsign
      0out=x,ctrl=0,in1=x,in2=x
      1out=0,ctrl=0,in1=0,in2=0
      2out=0,ctrl=0,in1=1,in2=0
      3out=1,ctrl=0,in1=0,in2=1
      4out=0,ctrl=1,in1=0,in2=1
      5out=0,ctrl=1,in1=0,in2=0
      6out=1,ctrl=1,in1=1,in2=0
      7out=0,ctrl=1,in1=0,in2=1

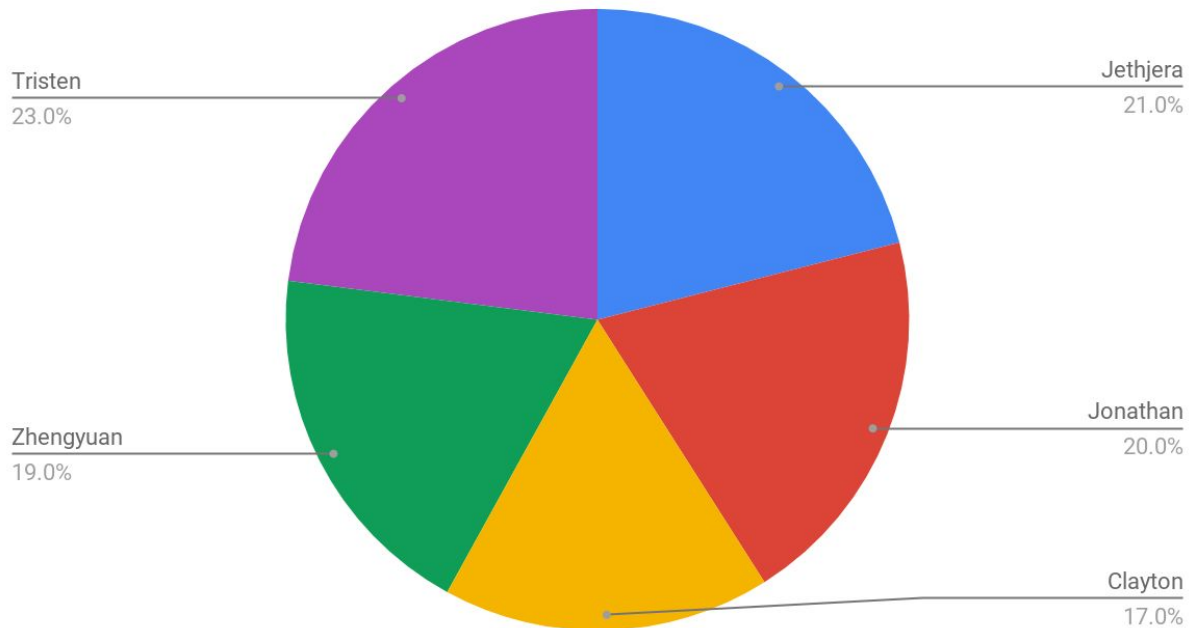
C:\iverilog\bin>
```

GitHub will be used for version control to make sure that we can create branches of each path that we choose without changing the master file unless the branch is properly represents the way that we want to build the ALU. These pull request will keep members from committing directly to the master folder and keeping version control for the project. <https://github.com>



Participation Census

Participation Census



Date	9/19	9/26	10/3	10/10	10/17	10/24	10/31	11/7	11/14	11/21	
Meeting	1	2	3	4	5	6	7	8	9	10	%
Tristen	o	o	o	o	o						
Jethjera	o	o	o	o	o						
Jonathan	o	o	o	o	o						
Zhengyuan	o	o	o	o	o						
Clayton	o	o	o	o	o						

* meetings will be held weekly on Thursday after class, and participation will be recorded.