Pretty Gate Machine

Updated Report

11/16/2018

Updated Description: Arithmetic Logic Unit with full data path.

~~As part of our Digital Logic and Computer Design course, we are going to realize a 4-bit Arithmetic and Logic Unit. An arithmetic logic unit (ALU) is a device that performs the basic arithmetic and logic operations of a processor.~~

~~Thus, our ALU can perform a minimum of 10 different operations on two 4-bit inputs, producing a single 4-bit output as well as an indicator of the output status (overflow, sign, zero, output hold).~~

The final ALU uses a 3-bit input termed the Opcode in Verilog to output one of nine functions. All operations are performed simultaneously, but the Opcode runs through MUX selectors to produce the desired output.

The main functions of an ALU are:

1. **Basic logical operations**. An arithmetic unit is used to perform the basic logical operations on numeric data (generally on 8 bits):
    - logical AND
    - logical OR
    - logical NOT, and
    - logical XOR (exclusive OR)
2. **Arithmetic operations**. An arithmetic and logical unit also makes it possible to carry out the operations on numerical data:
    - Addition,
    - Substraction,
    - ~~Division, and~~ **Division is not included in the final product.**
    - ~~Multiplication~~ **Multiplication is not included in the final product.**

3. **The binary comparison**: An arithmetic and logical unit also makes it possible to compare two numbers by indicating in a state register whether the result is larger, smaller or equal.

4. **The Shifter**. A shifter is formed of (n + 1) inputs D1,... , Dn, C (control bit) and n outputs S1, ..., Sn and operates a shift of 1 bit on the inputs D1, .. ., Dn:

- ~~If C = 1, this is a shift to the right,~~
- ~~And if C = 0, a shift to the left.~~
- **All operations are performed but are selected to be output near the mux.**

The various tasks of this project are composed but not limited to:

- Coordination and project management
- Development of truth tables
- Simplification of the equations
- Circuits design
- Choice of implementation mode (software discovery)
- Translation of the code to the implementation platform
- Implementation of the different parts of ALU
- Tests and validation
- Production of the project report
- **State Machine Diagrams**
- **Parts List**
- **Input Lists**
- **Output Lists**
- **Interface List**

- **Module List**
- **Mode List**
- **Output**

Updated Member Tasks:

Paul Jacobo:	Alex Palm:	Patrick Meyomesse:
Choice of implementation (Software Discovery)	Coordination and Project Management	Simplification of Equations
Translation of the code to the chosen platform	~~Development of Truth Tables~~	~~Circuit Design~~
Implementation of ALU Parts: Binary Comparison	Production of the Project Report	Tests and Validation
Implementation of ALU Parts: Basic Logical Operations	~~Implementation of ALU Parts: Arithmetic Operations~~	~~Implementation of ALU Parts: Binary Comparison~~
Homework: Verilog Portion Primarily	Homework: Split the Problems	Homework: Split the Problems
Circuit Design	**State Machine Diagrams**	**State Machine Diagrams**
Implementation of ALU Parts: Arithmetic Operations	**Input List**	**Mode List**
Output	**Output List**	**Interface List**
	Parts List	**Module List**

Updated Software Discovery:

For the Term Project Pretty Gate Machine is building an ALU with Verilog. The platform the Verilog code is compiling and simulating on is macOS Sierra version 10.12.6 running on a 2015 MacBook Pro. The Terminal accesses a specific file location to compile and run Verilog code. For ease, a split screen set up of The Terminal and text editor, Atom, is used to quickly code and run again. A unique extension is used while writing and editing code on Atom that allows the text editor to notify of any Verilog syntax errors.

Logisim will be used to create ideas and get a better sense of how modules are being built. Currently, Logisim is only being used for CS 4141 lab reports but is planned to be used for to plan the ALU.

Verilog was installed onto the platform via a GitHub Repository. In order to successfully complete the installation, the GitHub Desktop client was used to clone onto the platform.

Learning Verilog has been through trial and error. Resources such as http://iverilog.icarus.com/, stackoverflow.com, http://iverilog.wikia.com, are incredibly useful with plentiful information that leads the programmer in the way of solving the problem.

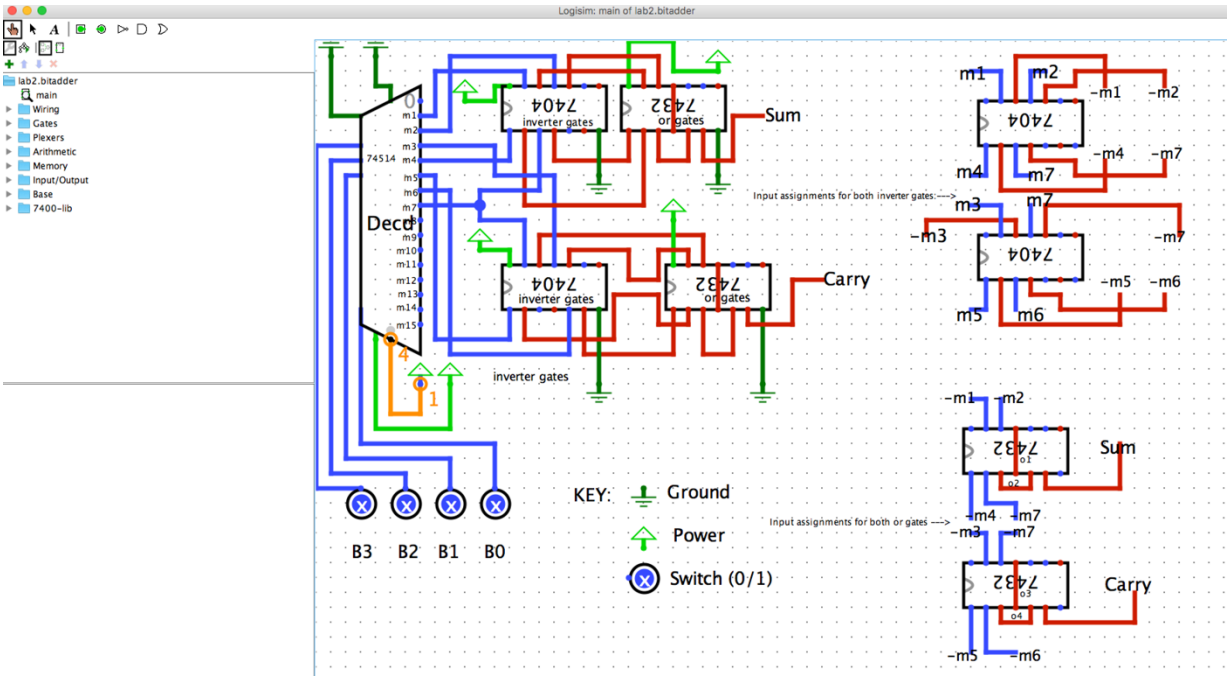
~~Implementing the Term Project's ALU is currently planned to contain multiple Verilog files with each being a complex module. The main source code will run include a module of all Verilog files' individual modules, creating the ALU as well as a test bench to run multiple delays and input.~~

All modules are included in one single file. No additional software was used however, Verilog issues were primarily solved on websites such as Stackoverflow.com and iverilog.wikia.com. Additionally, before

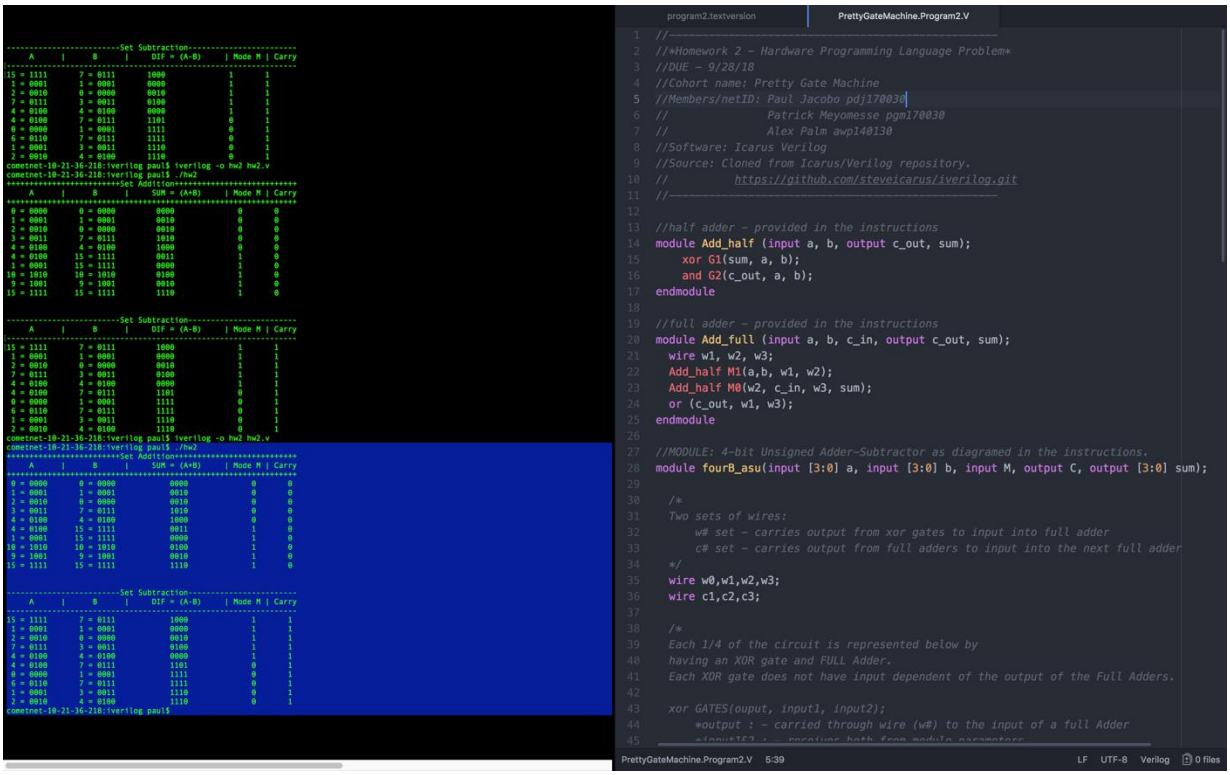
implementing any systems into code, models were simulated on Logisim to accurately display results.

SCREEN CAPTURES OF THE PLATFORM SETUP AND LOGISIM DIAGRAM HAVE BEEN INCLUDED

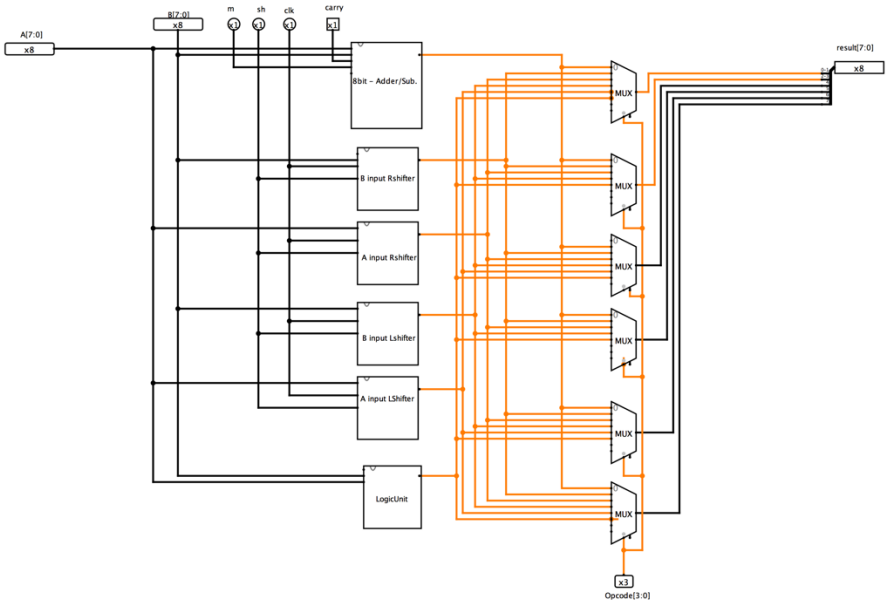
Logisim Diagram for 2nd CS 4141 Lab Report



Platform setup



UPDATED – Final Circuit Diagram



Updated Participation Census: