Updated Progress Report

November 22, 2019

Cameron Buchman

Jiang Bowen

Joshua Guzman

Albey Kappil

Swamynathan Singaravelu

Updated Description

An arithmetic logic unit (ALU) is often referred to as the “figuring” unit of a computer providing for the computational capability of hardware. Data buses connect an ALU to storage elements such as registers in order to perform computations that are efficient and complex. In the past, performance and cost were driving factors behind the design of ALUs. However, with the advent of mobile devices, power usage has slowly overtaken other attributes to become the most significant one in the design of ALUs. Numerous studies and experiments have illustrated that the most commonly used arithmetic operation is usually the bottleneck in terms of speed of the ALU. The ALU that will be developed in the project by the Beckstreet Boys will perform the following functions: Math, Logic, and Error-checking. The Math functions will include ADD, SUBTRACT, SHIFT, MULTIPLY, and DIVIDE. The Logic functions will include AND, OR, NAND, NOR, NOT, XOR, and XNOR. The Error-checking functions will include Overflow, Carry, Divide by 0, and out of memory. Since all these functions will be included as part of the ALU, the Beckstreet Boys have decided, after much debate, that the ALU will be best perform in image and video processing environments. Images and videos will be composed of picture frames that, when decomposed, will have pixels. Generally, image and video processors perform at the pixel level such as adjusting the brightness of a pixel or transparency of a pixel. Therefore, the ALU and the related circuit developed by the Beckstreet Boys will be useful for performing efficient calculations in image and video processing.

<u>Updated Member Tasks Lists</u>

Josh Guzman	• ~~Sketch circuitry of Add/Sub modules. Give sketches to Bowen~~ • ~~Write Verilog code for Add/Sub module~~ • ~~Create state machine visuals~~ • ~~Write Verilog code to handle the following errors: DivideByZero, and Out of Memory~~ • Write Verilog code for Shift Left and Shift Right modules • Debugged the Verilog code • Sketch circuit diagram for ALU
Albey Kappil	• Write the Verilog code for ControlLogic and Main multiplexer • Wrote the descriptions of modules and parts of the ALU • ~~Sketch circuitry of Shift Left and Shift Right modules. Give sketches to Bowen~~ • ~~Write Verilog code for Shift Left and Shift Right modules~~

	~~• Write Verilog code to handle the following errors: Overflow, and Carry-Over~~
Bowen Jiang	• Wrote the Verilog code for accumulator • Create the AND, NAND, and NOT modules • Creates circuit diagrams of all modules sketched by other members using the program ~~Circuit Diagram~~ • Accountable for ensuring that documentation is put together • Write Verilog code and circuit diagram for modules that take in the 8-bit input(s) and send them to operation modules
Cameron Buchman	~~• Sketch circuitry of Multiply module. Give sketch to Bowen~~ ~~• Write Verilog code for Multiply module~~ • Write Verilog code for the OR, NOR, XOR, and XNOR gates. ~~• Sketch circuitry for clock for overhead state machine~~ ~~• Write Verilog code of the clock for the state machine~~ • Help combine the individual modules into full working application
Swamy	• Write Verilog code for Add/Sub module

Singaravelu	• Write Verilog code to handle the following errors: Overflow, and Carry-Over • Create state machine visuals • ~~Sketch circuitry of AND, OR, and NOT modules. Give sketches to Bowen~~ • ~~Write Verilog code for AND, OR, and NOT modules~~ • ~~Sketch circuitry of the module that handles the output~~ • ~~Write the Verilog code for the module that handles the output~~

Updated Software Discovery

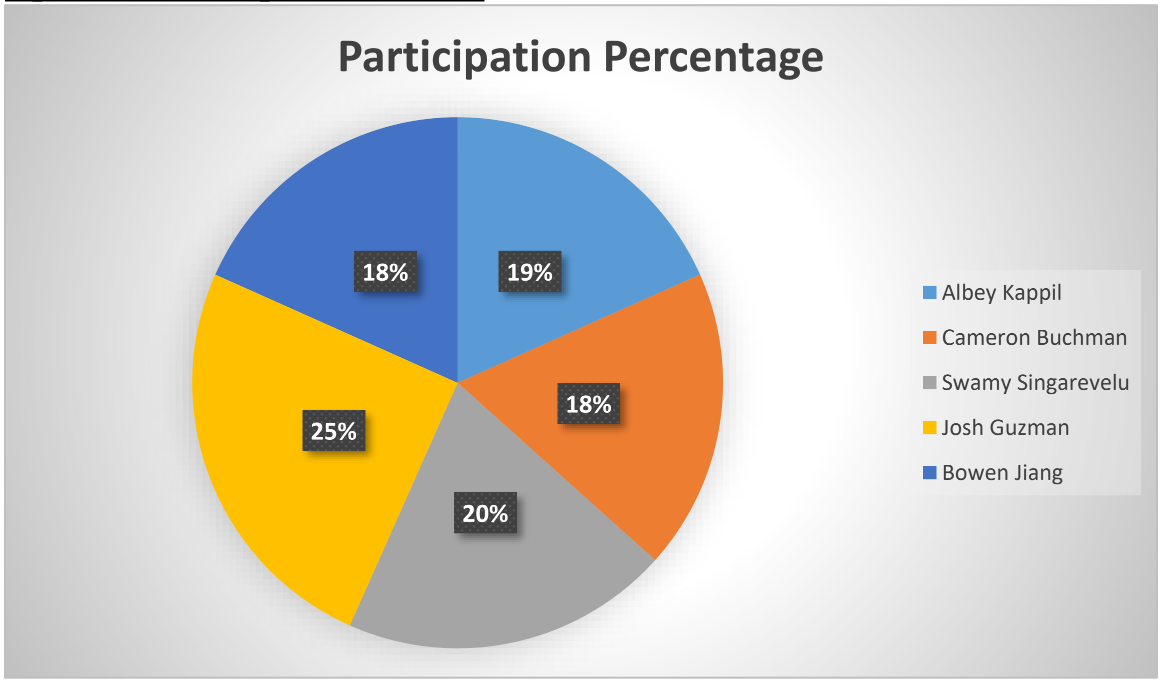
In order to show the circuit representation for our project, we have to select a piece of software that is easy to use and represents a circuit in a way that is easy to comprehend. ~~For this purpose we have chosen Circuit Diagram (https://www.circuit-diagram.org/). There was a multitude of reasons we chose this piece of software over all others, with the two main ones being that it is free to use and very simple to operate. Circuit Diagram can be used both in browser or downloaded and used offline allowing all the team members to work with it in whichever way is most convenient regardless of what operating system they are using. No matter how complicated our application gets it can accommodate us because Circuit Diagram~~

~~also offers free extensions that allow for different logic units to be represented.~~ We decided to use draw.io instead because it was just as easy to use and all of our team members have used it before and it allowed for us to more easily make the state diagrams.

A software that we used to design the state diagram is JFLAP. JFLAP is software for experimenting with formal languages topics including nondeterministic finite automata, nondeterministic pushdown automata, multi-tape Turing machines, several types of grammars, parsing, and L-systems. JFLAP allows for testing and constructing examples of state machines.

The next piece of software that is required for us to complete the project is choosing an HDL language to use to produce our ALU. The software we chose for this purpose is **Icarus Verilog (**http://iverilog.icarus.com/**)** which is the most recommended piece of software to work with Verilog. It operates as a compiler, compiling source code written in Verilog (IEEE-1364) into some target format

Updated Participation Census


Participation Percentage
Albey Kappil
Cameron Buchman
Swamy Singarevelu
Josh Guzman
Bowen Jiang