A Comparison of Proposed Full Adder Designs and Their Impacts on the Energy Consumption of an Assembly Program

Rachel Goodman
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

***Abstract*—The objective of "A Comparison of Proposed Full Adder Designs and Their Impacts on the Energy Consumption of an Assembly Program" is to evaluate a few different proposed Full Adder designs. First, an assembly program was designed that counts the number of occurrences of an input word in an input statement and outputs that number and where the words are located in the statement. Then, the dynamic instruction count of that program was used to calculate the total energy consumption of that program for each individual full adder design. Finally, based on those calculations, it was concluded that the program consumed the least amount of energy (174,937.6 pJ) when the Majority Gate-based Full Adder from [1] was implemented in the ALU.**

Keywords—Boolean gates, Majority Gate-based Full Adder, Magnetic Adder, Non-Volatile Full Adder, SLIM-AD, Domain Wall, Magnetic Tunnel Junction, Energy Consumption

I. Introduction

A. Project Design

First, the program prompts the user for an input statement and a word (up to 10 characters) that the user would like it to search for in the statement. It then converts both the statement and the word to all lowercase so that the word can be found even if it occurs in the statement or word with different letters capitalized. All counters are initialized. Next, each character in the input word is stored into its own register. The program then begins to iterate through the long statement, checking conditions as it goes. If it encounters a space or a new line, the index counter is incremented by 1, indicating the start of the next word in the statement. If the character matches the first character of the input word, it checks each of the following characters to see if they match also. If it encounters a letter that does not match, it continues iterating through the statement searching for the first character again. If all the characters match and the end of the word is reached, the word counter is incremented by 1, and the value in the index counter is stored in an index array. At the end, the program prints the word counter and the index array, informing the user of the amount of times the word occurs in the statement, and the indexes it occurs at. Finally, the program exits.

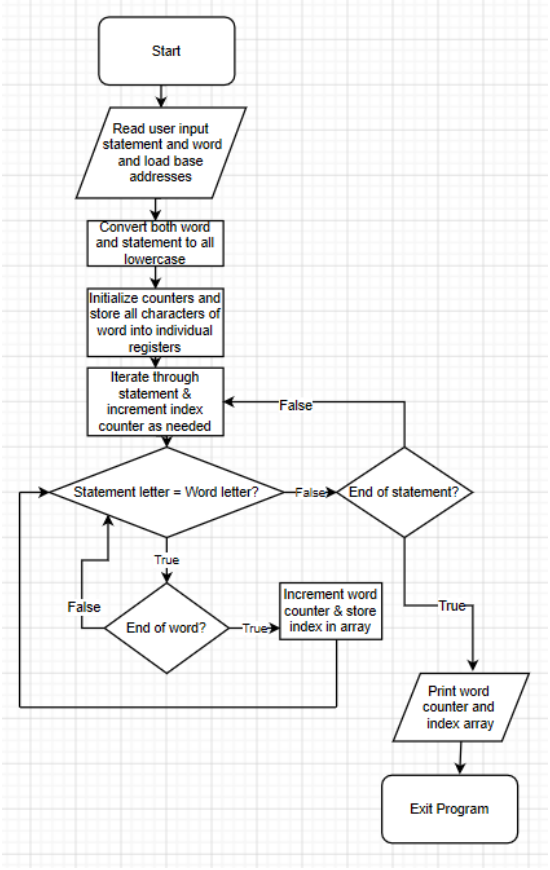

Fig.1: Flowchart of the assembly program.

B. Test Cases

For the first input, I used the statement about the Knights Graduation and Grant Initiative given in the task description, and I input the word "knight." I then compared it to the given correct result to ensure that my code worked as it should. This was a good way to test the code on a longer statement without having to count all of the words myself, because the correct count was provided in the assignment.

For the second input, I input a statement I made up. It goes as follows: " My name is Rachel. My friends call me Rachel, my parents call me Rachel, my extended family calls me Rachel. No nicknames, just Rachel." I input the word "RAcHeL." This was a good test to make sure that the code would work no matter if the capitalization differed between the word and the statement. Also, since it was a short statement, it was easy to count the words to make sure the numbers being output were correct.

For the third input, I input another short statement I made up: "ucf UCF uCf UcF UCf knights UCF uCF." This was another good test to make sure that the word would be counted correctly no matter the capitalization. It also made sure that the word would still be counted correctly even if it occurred over and over again, and if it occurred at the end of the statement.

Test 1:

```
# Please type in your input statement:
The Knights Graduation and Grant Initiative is a UCF award to help undergraduate students who cannot pay their tuition and their difficulty would
not allow them to finish their degree. The Knights Success Grant is the most
well-known program inside the Knights Graduation and Grant Initiative. In
order to be awarded the Knights Success Grant, you need to be referred but it
does not mean that all students who are referred will be awarded the grant. The
students who want to apply for the Knights Success Grant need to submit a
required application and complete the Knights Success Grant web course. For
more information, you can stop by their office in the Registrar's Office on the
main campus of UCF.
# Please type in a word (up to 10 characters) that you are looking for (e.g. Knight or KnIGhT, knight, &):
knight

# Number of times the word "knight
" was found in the input statement: 6

# Indexes of the matches found:2, 32, 42, 53, 85, 97,
```

Test 2:

```
# Please type in your input statement:
My name is Rachel. My friends call me Rachel, my parents call me Rachel, my extended family calls me Rachel. No nicknames, just Rachel.
# Please type in a word (up to 10 characters) that you are looking for (e.g. Knight or KnIGhT, knight, &):
RAcHeL

# Number of times the word "RAcHeL
" was found in the input statement: 5

# Indexes of the matches found:4, 9, 14, 20, 24,
```

Test 3:

```
# Please type in your input statement:
ucf UCF uCf UcF UCf knights UCF uCF
# Please type in a word (up to 10 characters) that you are looking for (e.g. Knight or KnIGhT, knight, &):
ucF

# Number of times the word "ucF
" was found in the input statement: 7

# Indexes of the matches found:1, 2, 3, 4, 5, 7, 8,
```

Fig.2: Sample outputs of the assembly program.

II. Full-Adder Circuit

Full adders are an extremely important component used in processors. A basic full adder takes 3 inputs and produces 2 outputs. The three inputs are the two bits to be added, as well as the carry in (C_I) bit. The two outputs produced are the sum bit and the carry out (C_o) bit [3]. Inside the full adder, Boolean gates perform the addition of the input bits.

The full adder designs discussed in this paper are a Majority Gate-based Full Adder [1], a Magnetic Adder [2], and a Non-Volatile Full Adder [3].

Each design performs the function of a full adder in its own way. The Majority Gate Full Adder performs its addition using majority gates, which output the value of the majority of their inputs. When it is implemented using the SLIM-ADC device proposed in [1], it outperforms other full adder designs in the areas of reduced power dissipation and delay. The Magnetic Adder in [2] is designed based on domain wall (DW) motion. Its inputs and outputs are stored in DW shift registers, which are non-volatile. This means that the circuit containing the adder can be powered on and off as needed without the adder losing its data. The Non-Volatile Full Adder discussed in [3] is designed using Magnetic Tunnel Junction and is faster than the proposed Magnetic Adder.

III. Results and Discussion

Table I: Energy consumption for a single ALU Instruction in the designs provided in [1-3].

Design	Energy Consumption For Each ALU Instruction
[1]	0.6 pJ
[2]	6.3 pJ
RTM-based [3]	1.67 pJ
STT-based [3]	1.61 pJ

In MARS, I ran my assembly program with the Test 1 input and obtained the dynamic instruction count. The total count was 12,884 instructions, with 5,031 ALU, 1,418 Jump, 4,920 Branch, 1,478 Memory, and 37 Other instruction types. Then, assuming that the ALU was implemented using a full adder, I calculated the total energy consumption of my program for each design using the given amounts for each type of instruction and the ALU energy amounts given in Table I. The results are given in Table II below.

Table II: Total Energy consumption for the assembly program using designs provided in [1-3].

Design	Total Energy Consumption
[1]	174,937.6 pJ
[2]	203,614.3 pJ
RTM-based [3]	180,320.77 pJ
STT-based [3]	180,019.91 pJ

According to the calculations, the design from [1], the Majority Gate-based Full Adder, was the most energy efficient. The Non-Volatile Full Adder designs from [3] used some more energy, while the Magnetic Adder design from [2] used the most by the largest margin.

IV. Conclusion

According to the source material and the calculations conducted within this report, the Majority Gate-based Full Adder proposed in [1] outperforms the other proposed designs in the area of energy consumption. It consumes 174,937.6 pJ when calculated using the dynamic instruction count of the Test 1 input to the assembly program described in Section I. The designs proposed in [3] come in a close second place, and the design proposed in [2] uses up the most energy of them all.

Technical topics I have learned from this project include:

- There are several different kinds of full adders, and many different ways they can be designed, each with its own pros and cons.
- An adjustment to a single component in a circuit can make a big difference in the amount of energy consumed by the circuit.
- I learned how to calculate total energy consumption using dynamic instruction count of a MIPS assembly program run in MARS.
- Non-Volatile Adders are the most convenient to use if the data within needs to be retained even when the circuit is powered off.
- I learned how to store an array of numbers as they were being counted and print the array using MIPS assembly code.

References

[1] S. Salehi and R. F. DeMara, "SLIM-ADC: Spin-based Logic-In-Memory Analog to Digital Converter Leveraging SHE-enabled Domain Wall Motion Devices," Microelectronics Journal, Vol. 81, pp. 137–143, November 2018.

[2] H. P. Trinh, W. S. Zhao, J. O. Klein, Y. Zhang, D. Ravelsona, and C. Chappert, “Magnetic adder based on racetrack memory,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 60, no. 6, pp. 1469–1477, Jun. 2013.

[3] K. Huang, R. Zhao and Y. Lian, "A Low Power and High Sensing Margin Non-Volatile Full Adder Using Racetrack Memory," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 62, no. 4, pp. 1109-1116, April 2015.