Implementation of Non-Volatile Memory Designs with Full Adder

Abdulsalam Khan
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

***Abstract*—As predicated in Moore's law, the continued advancement in technology has led to miniaturization of our current electronics devices almost every two years to a scale that possess great challenges in terms of energy consumption, speed delay, and other areas dealing with accuracy, etc. To tackle some of the important challenges like energy consumption and issues with delays, there has been tremendous work done in the scientific community to come up with better solutions and efficient designs. As the title states, this report discusses the different designs for non-volatile memory devices (NVM) on a basic principle of a full adder. As required for this project, a program was developed to take user input and search the occurrence of user input search word and also display the indexes. First part of the report introduces the task of the program in detail with section A going through project design in detail. Section B discusses the test cases used validate accuracy. The second part of the report summarizes an analysis of Full Adder circuit using different designs. Then, part three of the report discusses the results of energy consumption and finally, a conclusion is provided on part five. The total energy consumption calculated of the best design being SLIM-ADC [1] is 15126 pJ.**

Keywords—SRAM, Non-Volatile Memory, Magnetic Tunnel Junct

I. Introduction

The task of this project was to accept unspecific amount of text and be able to search any type of word with a limit of 10 characters and display the indexes of where the searched word appears in the text. The project design of the code and test cases chosen to account for all possibilities to show the code executes correctly as per requirements.

A. Project Design

As shown in *Fig. 1*, the code begins with prompting the user for input statement and stores that as a label to access later easily. Then prompt the user for look up word and stores that as 'Search_Word' to also access later easily. As per requirement, the design of the code could be broken down into two parts. First, finding the number of times the search word appears in the input statement and second, finding the indexes of the searched word. During the process of the design, there was a problem with taking in search word as user input and displaying the search word in between the required sentence as it would enter second

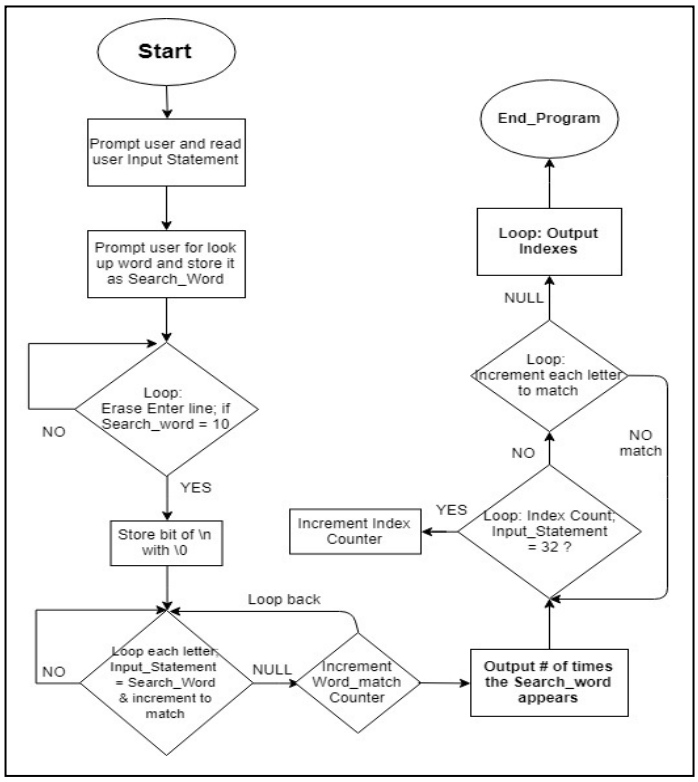

Fig.1: Flowchart of the assembly program.

line. After researching and debugging the code, the cause of the problem was determined to be an automatic print line '\n' whenever the user presses enter key after typing in a search word. I devised a solution that would erase the enter line by using a loop to go through each character of the search word until it equaled to ASCII value of 10 (New Line) and then store that bit with null (\0) character. Then using similar loop, the input statement and the search word was compared for each character until it matched. Once all letters were matched and there was a null character, the word match counter was incremented by 1 to indicate a search was found and then loop back to the previous loop too keep searching until final it went through whole input statement. Then word match counter was

used to output the number of times the search word appeared in the input statement.

For the second part of the design to find the indexes, use of multiple loops and counters was utilized to display the desired outcome. A loop is used to simultaneously compare letters of input statement and searched word while checking for a space between words with an ASCII value of 32 to increment the index count. The loop branches into another loop if no space is found to increment load byte of input statement and match letters. If no match is found, it jumps back to the first loop to check for space and if found increments the index. With this devised method, the index is incremented and once a match word is found, the index of that matched word is printed out. After each matched index print, a comma is also printed with a use of a counter to keep track until it matches with the total number of searched word appearance. Then a period is printed on the last index as required in the assignment output example. After outputting two critical requirements with formatting, the program finally calls for an end.

B. Test Cases

The three test cases shown in *Fig. 2* show the ability of the code to meet the requirements as tasked in *Project 3: Capstone Report Option*. The first screen shot shows input of four same words with different input styles to test the code if it can read different styles of inputs and correctly output the result. As shown in the screen shot, the program correctly outputs the results and meets the requirement handling different kind of input styles. Second screen shot shows an input of large text which is used from the sample provided in the requirement to test if it can intake large quantity of characters and go through all characters and output results correctly. As shown in the second screen shot, the results match with sample example provided and proves that it meets the requirement. Finally, the third screen shot shows a random input of sentence but this time to test the limit of search word which is set to 10 as per requirement. The search word used is “Crackerjack” as it is 11 characters long which is greater than 10-character limit. As shown in the third screen shot, the program only takes in first 10 characters of the word “Crackerjack” and displays 0 matched word and no indexes as it was not found in the input statement. Finally, this last requirement is met and all three test cases prove to be sufficient to test the code against the requirements and validate its accuracy.

II. Full-Adder Circuit

For the purpose of fully understanding full adder and the later designs implemented, one must understand the basics. A full adder is a digital circuit that performs addition on three one-bit numbers. Taking in three inputs and carry value to produce a sum and carry out to carry bit to the next adder.

To overcome some of the important challenges such as power consumption and interconnection delay with miniaturization of the circuitry, the complementary metal-oxide-semiconductor technology has been surpassed by the newer designs of Magnetic Tunnel Junction (MTJ). The many reasons including the non-volatility has paved a way to better designs such as NVM spin-based devices. One being known as Spin-based Logic-In-Memory or (SLIM-ADC) which provides faster logic operations due to non-volatility which in return reduces energy consumption. The energy consumption is

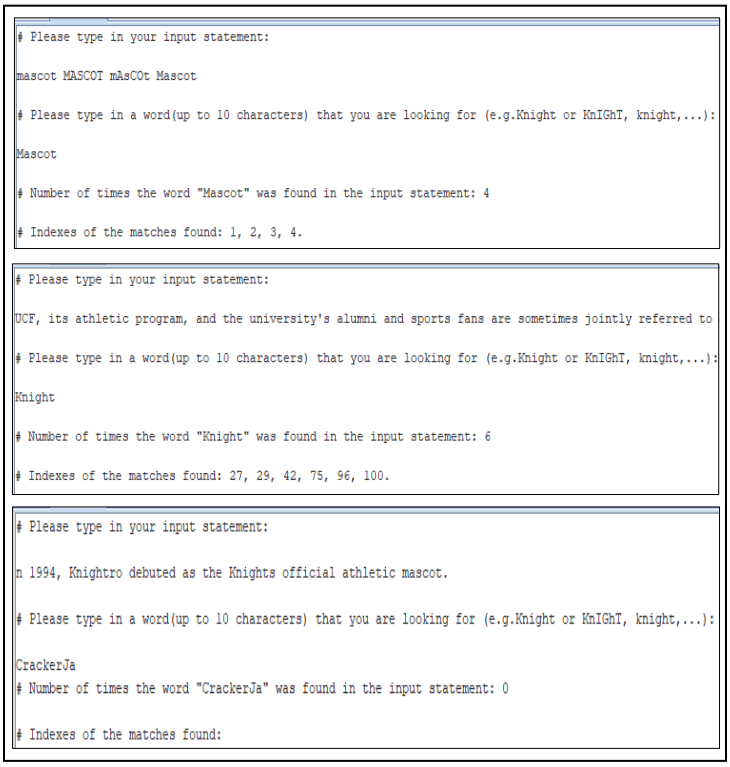

Fig.2: Sample outputs of the assembly program.

reduced due to the quick on/off operation without the use of backup storage. The MTJ devices can be used as different function such as OR gates, Majority Gates (MG) and AND gates. With the three inputs combined, 1 bit full adder is realized.

Another design leading to solve many of challenges is a multi-bit magnetic adder based on domain wall motion. Similar to the SLIM-ADC design, the circuit could be turned off safely without the need of data backup but this design efficiently tries to overcome standby power issue. The data moved between logic and the memory is greatly reduced due to integration of memory and logic circuit together. This also begs the question of interconnection delay and this design surely time.

In the continuum of discovering better designs, another design studied uses the Racetrack memory (RTM) with non-volatile full adder. This design results in smaller area requiring reduced power, and delay speed. The technique of this design stores the inputs and outputs in perpendicular fashion, hence, the racetrack memory.

To better the understanding of nonvolatile full adder based on logic in memory architecture [4], a deep dive is done to gain in-depth picture. The use of MTJ with metal oxide semiconductor transistor is used to fabricate full adder logic in memory architecture. The advantage of high tunnel magneto-resistance ratio on magnesium oxide barrier is used to confirm the fabrication of basic operable full adder.

III. Results and Discussion

In the section below, energy consumption is calculated using the provided values for branch, jump, memory, other, and ALU in Table 1.

1) *ALU = Refer to Table I*
2) *Branch = 3 pJ*
3) *Jump = 2 pJ*
4) *Memory = 100 pJ*
5) *Other = 5 pJ*

Based on different technologies proposed in [1-3], the energy consumption values for ALU were used from the Table I. In order to calculate the total energy consumption listed on Table II, dynamic instruction count of the program was found using MARS4.5 Tools > Instruction Statistics. Dynamic count for each instruction is multiplied by the energy consumption per instruction provided in Table I, and added together with rest of the instruction types. The following formula below illustrates the calculation for energy consumption:

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

Energy consumption = (ALU x 315) + (J x 24) + (B x 160) + (Mem x 144) + (Other x 37)

As show in Table II, the results indicate the total energy consumption for the program using the four different designs. It is noticed that if the input statement from the user is increased, the overall energy consumption also increases but for this part of calculation, a shorter input was used. The magnetic adder based on racetrack memory [2] design seems to consume most energy while the SLIM-ADC [1] design consumed the least amount of energy. The RTM-based [3] and STT-based [3] designs consumed more energy than the SLIM-ADC but not as much as magnetic adder RTM design. Overall, the SLIM-ADC [1] seems to perform better than other designs in terms of energy consumption.

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

Design	Total Energy Consumption
[1]	15126 pJ
[2]	16921.5 pJ
RTM-based [3]	15463.05 pJ
STT-based [3]	15444.15 pJ

IV. Conclusion

The paper has presented an understanding of different non-volatile memory designs and implementation of the program to analyze the energy consumption. Many of the topics covered included complementary metal-oxide-semiconductor (CMOS), Magnetic Tunnel Junction (MTJ), Non-Volatile Memory (NVM), Racetrack Memory (RTM), and Spin-based Logic in Memory (SLIM). The analysis of the different designs creates a deeper understanding on challenges faced such as energy consumption and interconnection delay. The best design being the SLIM-ADC consumed energy of 15126 pJ compare to other designs.

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 et al., "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.

[4] Matsunaga, Shoun, et al. "Fabrication of a Nonvolatile Full Adder Based on Logic-in-Memory Architecture Using Magnetic Tunnel Junctions." *IOPScience*, Applied Physics Express, 22 Aug. 2008, iopscience.iop.org/article/10.1143/APEX.1.091301/meta.

[5] A.M. Shams ; T.K. Darwish ; M.A. Bayoumi "Performance analysis of low-power 1-bit CMOS full adder cells"Volume: 10, Issue: 1, Feb. 2002.

[6] Sverdlov, Viktor, et al. "CMOS-Compatible Spintronic Devices." *Ieeexplore.ieee.org*, IEEE