Soft Error Prevention and Correction Methods Impact upon Energy Consumption

Joshua Weed
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

***Abstract*—This paper examines soft error prevention and correction techniques for memory cells with a focus upon the impact various approaches have on energy consumption. Errors within memory are a primary contributor to poor performance and as devices continue to shrink in size and operating voltage the options for designing to prevent these errors are becoming limited. Research in this area to combat the problem has shown multiple approaches can be taken to improve program reliability. A test program was created to evaluate the improved nonvolatile spintronic flip flop design, range repair of range matching content-addressable memory, and double node charge sharing SEU tolerant latch methods impact upon energy consumption. The test program is a word search tool written in assembly language to take in a user input body of text and a user supplied word to search the text body for. The program then outputs the number of occurrences of the word to be searched for as well as the index of the occurrences of that word. The most energy efficient design was the DNU latch that consumed fJ.**

Keywords— *reliability, soft-error prevention, single-event upset (SEU), memory elements, non-volatile flip-flop (NVFF), range matching content-addressable memory (RM-CAM), triple-modular redundancy (TMR), double node charge sharing SEUs (DNCS-SEUs)*

I. Introduction

The assembly language code used to explore the topics discussed within this report was written to function as a word search tool. This style of program was chosen due to the need to incorporate substantial memory access tasks in order to realize a word search tool. This provided a sample to use in comparing the various methods effect on energy consumption. To create the test program

This program takes in a series of words up to many sentences in length input by the user. After that string is stored to memory the user also inputs a word they would like to search the string for. The program then outputs the number of times the word occurred as well as the index (word number) where the words occurred. In order to test code functionality a baseline test case with multiple sentences made up of upper and lower case letters including spaces and numbers was developed. A secondary test included multiple occurrences of the search word on the ends of the text body with the goal of checking the indexing ability. This stress test includes many occurrences of the search word and tests the reporting ability

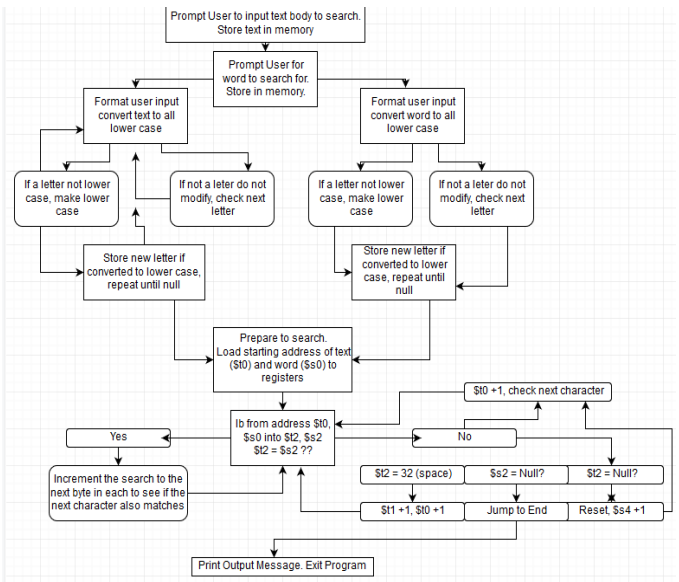

Fig.1: Flowchart of the assembly program.

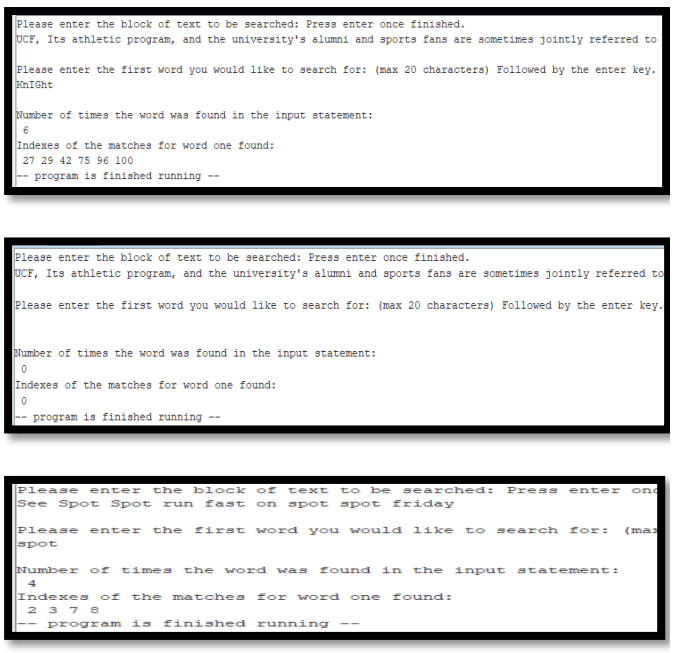

Fig.2: Sample outputs of the program.

of the word number and locations reported for accuracy. A final logic test involves no user input search word. In this case the continuity of the program is checked for when there is no test word to search for.

A. *Project Design*

The program begins by prompting the user for a block of text that they would like to search for a particular word. They are then asked to enter the word they would like to search for. Both inputs are allocated space in main memory to be called upon later. The program then converts both user supplied inputs to all lower case in order to ease operations during the search phase. This conversion is done through manipulation of the asci character codes where those that are upper case are altered and those that are lower case or other symbols/ numbers are untouched. Once the conversion is complete the registers are all cleared and the address of the first byte of both the word and the text body are loaded into registers $s0 and $t0 with the data located at those addresses being stored in $s2 and $t2. A series of branch then jump statements are covered to search the text body and react accordingly to the various combinations of situations that can occur. Beginning with a comparison of the first character within the text body to the first character of the word, the program proceeds byte by byte checking each character in the text body. If the two are not equal the program looks to see if the character in $t2 is asci code 32 to represent a space. If this happens it is tallied to count the word number using the formula word number is equal to the number of spaces plus one. Then the program will increment $t0 by one and continue the search sequence. Another condition to check if the $t2 and $s2 bytes aren't equal is if either the null terminator. If the text body has a null terminator it means that the string has been fully searched. If $t2 holds the null terminator it indicates that the word was found within the text block and this must be recorded and the value in $s0 must be rest to the beginning address of the word being searched for so that the program can continue. Once the text body has been searched, the exit sequence begins by printing the format for the output. The output states the number of times the search word occurred as well as an index of where the search word occurred in the text body. A brief overview to show relationships can be seen in Fig 1.

In short, this design was constructed through the use of sequential and looping conditional statements. These statements were grouped into defined segments based on the actions needed to perform the word search. These stages can be categorized as input storage, input organization, search algorithm preparation, algorithm execution, and output presentation.

B. *Test Cases*

In order to verify functionality of the written MIPS code three tests were performed. To begin a control input was selected that included both upper and lower case characters, symbols, numbers, spaces in a representative length for text segments expected to be processed. This manner of control input would test the conditions set forth in the project requirements for length and input parameters. Once the ability to complete the control was exhibited a second continuity test was performed to evaluate the logic behind the search algorithm. The second test examines what happens when no user supplied word to search for is given. Another trial stresses the index tracking feature and the data output method by including numerous occurrences of the search word within the text body where those occurrences are on the ends of the text segments memory. This test would have shown potential flaws in the ability to count spaces for indexing purposes. An error in output could also be observed here had not enough space been allocated to store the index values or if the logic had counted on conditions similar to the control input.

II. Memory Bit-Cells

As semiconductors continue to shrink in size following the trend set forth in "Moore's law", technology is reaching a point where reliability of chip level processes is becoming an increasing concern. Reliability in simple terms is a device's ability to function as designed over a time interval. As the gate size shrinks further in CMOS devices the physics involved begin presenting problems that affect the ability to manufacture and operate without error [5], [2], [3]. Soft error that effects memory is the main focus of this discussion. Triple Modular Redundancy (TMR) is a technique that has long been in use to work with the presence of errors. TMR is a system of three separate units that each completing the same task where the units then compare the outcomes and "vote" on what the correct answer should have been with majority vote ruling. When taking this approach a random soft error such as a single event upset can be filtered out with the odds of each experiencing an error in the same area being slim. This leads to a an increase in the chance for correct operating due to decisions being made based on correct and not corrupted data. It comes at the expense of greater energy consumption due to the fact that the same task is run three times, two more than would otherwise be required. When acting as a part of the critical data path the TMR voter will have the final say in which bits are deemed the proper bits to be crunched. If the vote does not align with what the true value should have been the resulting decisions made on that data will be incorrect, this can cascade and cause further errors. Depending upon the type of error (whether it is the interjection of noise that can be interpreted as an unnecessary bit or the mixing of a one or a zero) the severity of consequences for circuit output can be determined [4].

Delving deeper into the physics behind the challenges to reliability brought about by the nanometer scale CMOS devices one must look at radiation hardening. Radiation hardening is measures taken to prevent the ionization radiation that has the energy to upset the workings of a semiconductor by freeing electrons in a manner not accounted for in the design. As devices shrink they become tougher to apply traditional radiation hardening techniques to. The spintronic memory devices can tolerate ionization radiation, the semiconductor based circuits they work with are still effected [1]. When

prevention by shielding and hardening is not possible, error prevention must be built into the device circuitry. These techniques require a system of checks and filters to verify that the information being sent is valid prior to writing to memory. These methods allowing for the presence of errors within the code are known as defect tolerance approaches [2]. Some designs actively work to correct errors that are detected with error detection and correction techniques being a growing area of development [2], [3].

III. Results and Discussion

In order to compare the energy consumption of the program implemented via the various design approaches mentioned above the energy consumption for all tasks other than memory access were kept constant. See table III for a breakdown of the calculations performed. The equation used was the summation of the number of each type of instruction executed multiplied by the femto Joules of energy consumed per instruction. Table I shows the deciding factor for the calculations for energy consumption with the results displayed in table II.

Table I: Energy consumption for a single bit-cell memory in the designs provided in [1-3].

Design	Energy consumption of a Single Bit-Cell Memory
SEU-Latch [1]	0.88 fJ
DNU-Latch [1]	0.28 fJ
[2]	6.96 fJ
[3]	1.51 fJ

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

Design	Total Energy Consumption
SEU-Latch [1]	13288.12 fJ
DNU-Latch [1]	12898.72 fJ
[2]	17234.04 fJ
[3]	13.696.99 fJ

The results presented in the table show that it is more energy efficient to prevent error by designing with reliability in mind than it is to design to prevent and correct error to achieve reliability. The designs from reference one and reference three had lower energy consumption due to the lower energy consumption of the memory access tasks. These designs were engineered with energy efficiency in mind. Design two was also engineered with efficiency in mind, however, it has a different level of error correction built in as discussed above and as a result consumed more energy.

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

Task Type	Instruction Count	Energy Consumed
ALU	714	1 fJ
Branch	1739	3 fJ
Jump	638	2 fJ
Memory	649	Varies by design
Other	1102	5 fJ

IV. Conclusion

In general, as CMOS size continues to decrease soft error prevention and correction are becoming crucial to ensuring device reliability. A broader overview of the challenges facing memory value reliability and the steps needed to prevent and correct soft error were touched on in this document. This project provided the experience of reading upper level technical documents for comprehension while practicing the skill of reading and consolidating information from multiple sources to clearly convey an idea. Exposure to the layout and formatting for an IEEE style articles was also presented. The total energy consumed by the most efficient design was 12.9 pico Joules by design one's DNU-Latch.

References

[1] F. S. Alghareb, R. Zand and R. F. Demara, "Non-Volatile Spintronic Flip-Flop Design for Energy-Efficient SEU and DNU Resilience," in IEEE Transactions on Magnetics, vol. 55, no. 3, pp. 1-11, March 2019, Art no. 3400611.

[2] H. Pourmeidani and M. Habibi, "Hierarchical defect tolerance technique for NRAM repairing with range matching CAM," 2013 21st Iranian Conference on Electrical Engineering (ICEE), Mashhad, 2013, pp. 1-6.

[3] K. Katsarou and Y. Tsiatouhas, "Double node charge sharing SEU tolerant latch design," 2014 IEEE 20th International On-Line Testing Symposium (IOLTS), Platja d'Aro, Girona, 2014, pp. 122-127.

[4] V. Kiani and P. Reviriego, "Improving Instruction TLB Reliability with Efficient Multi-bit Soft Error Protection," *Microelectronics Reliability*, vol. 93, pp. 29–38, Feb. 2019.

[5] G. Gielen, P. D. Wit, E. Maricau, J. Loeckx, J. Martin-Martinez, B. Kaczer, G. Groeseneken, R. Rodriguez, and M. Nafria, "Emerging Yield and Reliability Challenges in Nanometer CMOS Technologies," *2008 Design, Automation and Test in Europe*, Mar. 2008.