Reliability Methods of Modern Computer Software and Bit-Cell Hardware

Isaiah Williams
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

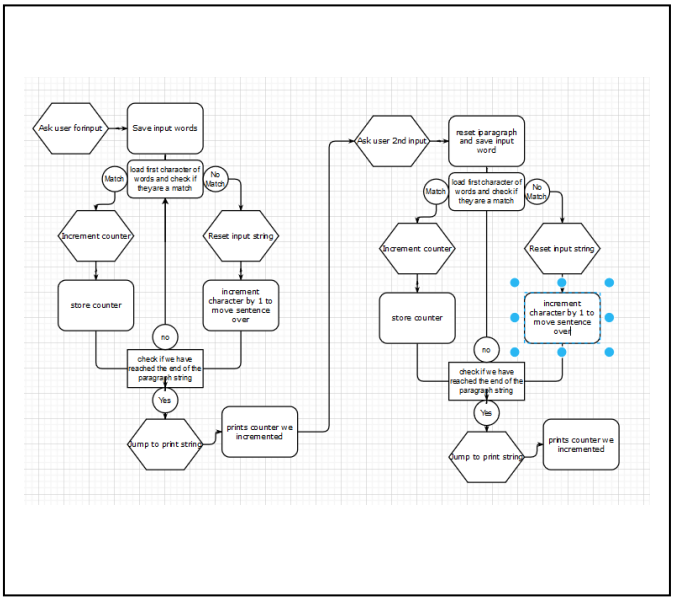
***Abstract*— The level of reliability of bit-cells within a digital system is crucial to its ability to properly carry out tasks when it is met with errors or upsets, otherwise such upsets can cause catastrophic failure within an entire system. Types of errors include SEU's AND DNU's, and within this paper we will investigate the methods that are being explored to be able to ensure higher reliability of bit-cell and digital systems, such as radiation hardening, and in particular triple modular redundancy (TMR). A program (Project 3 code) has also been created that will count the appearance of two user input words within a paragraph string and output the number of occurrences of them. This program will be analyzed, and the methods behind it explained. The program will also be examined and used in determining the energy consumption of different types of error-resistant latches. And we have found that the DNU-latch is the most energy efficient option at 117,834.04fj when calculating energy for this program.**

Keywords—Single-event upset (SEU), Double-Node upset, reliability, Triple Modular Redundancy (TMR), bit-cell, Content addressable memory (CAM),radiation hardening, DNU-latch, SEU-latch, Spin-transfer torque

I. Introduction.

A. Project Design

The goal of the MIPS program in this paper was to take two user input words, with a maximum length of 10 characters and check to see how many times that word occurred within a string (in our test cases it was a paragraph. The design of this program was based mainly on the use of arrays (accessing memory addresses in MIPS). The design of this program was based on matching the individual characters between the paragraph string with the characters of the user input words. Firstly, this program will request and then store up to two user input words that are to be used for searching in the paragraph. Once these inputs are stored the searching algorithm can begin. Next the first byte(array index) will be loaded into two separate variables after this there will be contingency checks to check to see if the end of either string is reached, if the end of the user input is reached then it will increment a counter that counts the number of occurrences, and at the same time if the end of the paragraph is reached the program will print the results. After the checks occur, to make the checking process simpler the inputs and searching string are converted to lowercase ascii character for conversion using a 'jal' instruction that jumps to



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 difficul
# Please type in two words(up to 10 characters) that you are looking for (e.g. Knight or KnIGht, knight,...):
Please type your first word: TUITION
Please type your second word: grant
# Number of times the word TUITION
 was found in the input statement: 1

# Indexes of the matches found:

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

# Indexes of the matches found:

-- program is finished running --
```

2.

```
# 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
# Please type in two words(up to 10 characters) that you are looking for (e.g. Knight or KnIGht, knight,...):
Please type your first word: KNights
Please type your second word:
# Number of times the word KNights
 was found in the input statement: 6

# Indexes of the matches found:
```

3.

```
# 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 dif
# Please type in two words(up to 10 characters) that you are looking for (e.g. Knight or KnIGht, knight,...):
Please type your first word: KnightS
Please type your second word: KnightS
# Number of times the word KnightS
 was found in the input statement: 6

# Indexes of the matches found:

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

# Indexes of the matches found:

-- program is finished running --
```

an ascii conversion function below main function and will return the lowercase value of the register that held the upper case prior. Then it will branch if there is no match between the two characters and go to a "move on" label which will increment the paragraph address to move one character over to continue the search, but will reset the user input array index to the beginning of the word by storing the original user input back into it before iterating the next loop.

If matches are found up to the null terminating character in the user input then we have a successful match and can increment a counter $t7, used to count occurrences of words within the string .This will continue until the end of the searchable string is reached, then it will branch to label *useroutp1* which will print out our required output and for the second word the process functions exactly the same then it will exit the program. To find the indices of where these matches were found, one way to go about it would be to have a label that holds the addresses of the matches in another label to print after matches are confirmed. This process is also shown in the above Flowchart.

B. Test Cases

Three test cases were chosen, in the first test case the words "TUITION" and "grant" were chosen. These specific cases were chosen to not only check if the whole code was working but to check that the ascii conversion function completes its job in making comparison easier, which it did by changing all of "TUITION" to lowercase. The Second test case input "Knights, was to test to make sure that if only one word was entered that the code would still function. The third and final test case inputs "Knights" and "Knights" were chosen to see how identical inputs would affect the output as well as to make sure that the 1st and second loops are independent of each other.

These test cases all test different outcomes and possibilities and ensure the program is functional across the parameters we have chosen.

II. Reliability Bit-Cells

When referring to Computer Architecture reliability is a crucial part to the performance and function of components (hardware and software),because if a component can continue to work correctly for a longer time, despite SEU's, or DNU's it is seen as more reliable, and in the business world is more valuable to a consumer. Within the scope of computer architecture however reliability is really shown by a machine, or circuit's ability to withstand random error failures and continue to keep working. Increasing and ensuring higher fault tolerance rates can help increase the reliability of new technologies, reliability is increased by acts of radiation hardening to avoid radiation based failures, bug-fixing in software, and for cluster defects using range-matching CAM to fix issues, and Triple Modular Redundancy (TMR).More aspects of increasing reliability is using spin-transfer Torque in RAM, developing higher density memories, and scalability of said methods.

TMR will be the focus of this discussion, and TMR is the implementation of logic gates in a fashion that allows for the failure of 1 out of 3 components, that is corrected by the other 2 components in the logic circuit sending their signals to a voter gate. The reliability of this voter gate is the most important aspect, so to further increase reliability the TMR can be made even more secure by adding another level of triple redundancy. This method increases reliability by accounting for more and more possible single-point failures and makes up for it. While it is a very good method for increasing reliability, it does have the drawbacks of consuming more energy due to the increased number of logic circuits, as well as increased usage of area in the component are some drawbacks.

As discussed before the voter gate is the most crucial gate because it dictates the final output and is meant to be kept in check by at least 2/3 of the inputs being correct, if for some reason this voter fails it could result in catastrophic failure of a component, which is why the method of doubling up the TMR mentioned before can really reduce the failure rate by having an increased number of sub-voters all feeding to the voter variables. Even within the software written for this topic, methods of increasing reliability of the program were introduced such as making sure that the two outputs are not dependent on each other and making them modular was important so that no "cross contamination" of code happens and allows the code to run for many software iterations and is resistant to single event upset errors. And of course, testing many cases allows the programmer to identify where some holes in the program might be. Furthermore an important aspect of reliability is its ability to be used in many applications. This is where scalability comes into play. It has been observed that the use of a magnetic hard disk is a way of making higher density memories. This leads into the use of techniques such as, Spin-transfer torque as an efficient upset tolerant mode of reliability. This implementation can be used in systems of varying sizes, and this leaves it as an important technique of reliability.

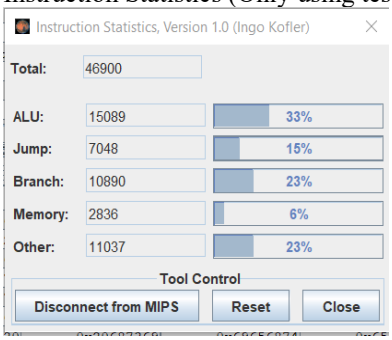
III. Results and Discussion

To determine the total energy consumed by the Word searching program, it was necessary to ensure some form of uniformity among calculations the first test case we used in section I (First input: "Tuition", Second input: "grant") will be our sample for energy consumption. Using the instruction value figures mentioned below, the total energy consumption of the program was calculated as such for each form of single-bit memory:

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

Instruction Statistics (Only using test case #1):



For the instruction types we use the following for each value:

1) ALU = 1 fJ
2) Branch = 3 fJ
3) Jump = 2 fJ
4) Memory = Refer to Table I for memory values
5) Other = 5 fJ

SEU-Latch:

$$(1 * 15{,}089) + (3 * 10{,}890) + (2 * 7048) + (0.88 * 2836) + (5 * 11037) = 119{,}535.68fJ$$

DNU-Latch:

$$(1 * 15{,}089) + (3 * 10{,}890) + (2 * 7048) + (0.28 * 2836) + (5 * 11037) = 117{,}834.04fJ$$

Figure [2]:

$$(1 * 15{,}089) + (3 * 10{,}890) + (2 * 7048) + (6.96 * 2836) + (5 * 11037) = 136{,}778.56fJ$$

Figure [3]:

$$(1 * 15{,}089) + (3 * 10{,}890) + (2 * 7048) + (1.51 * 2836) + (5 * 11037) = 121{,}322.36fJ$$

These final figures are listed in Table II below:

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

Design	Total Energy Consumption of program
SEU-Latch [1]	119,535.68 fJ
DNU-Latch [1]	117,834.04 fJ
[2]	136,778.56 fJ
[3]	121,322.36 fJ

From the above data it is also suffice to say that the use of a DNU-Latch will result in the lowest energy consumption at a value of 117,834.04 fJ and therefore makes for the most efficient mode of memory for executing this code, as well as it provides a better reliability than an SEU-latch which will only protect against single-event-upsets in circuit.

IV. Conclusion

In this paper, a program has been written that will identify and display the number of times a user-chosen word is found to occur in a hardcoded string. We have analyzed this program in depth and viewed several test cases to test the range of this program, and in all of our test cases it has successfully run. We have also discussed how reliability within machines can be improved, by delving into the matter of TMR(Triple Modular Redundancy), and furthermore we have further looked at the energy consumption of several reliability increasing circuits/latches, to observe how a tradeoff for reliability and energy consumption impacts real-world usage of a computer system. In our findings in Section III we see that the DNU-latch was found to be the most energy efficient Single bit-cell memory options

5 technical topics:

1-Learned about how Triple modular Redundancy's implementation can improve digital system's reliability

2-Learned about different types of logic errors (SEU's, DNU's, radiation-based errors, nano-cluster defects)

3-Learning how to use and implement MIPS coding methods (such as jal, loop design, memory address manipulation, etc.)

4-I've learned how scientific research papers are methodically carried out, by completing this project and reading through the reference papers.

5-I've learned about how to find energy calculations of a program in Mars, which can be useful when making programs in real-world applications, and I need to conserve energy for a project.

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] Kim, Y., Gupta, S. K., Park, S. P., Panagopoulos, G., & Roy, K. (2012). Write-optimized reliable design of STT MRAM. In *ISLPED'12 - Proceedings of the International Symposium on Low Power Electronics and Design* (pp. 3-8). (Proceedings of the International Symposium on Low Power Electronics and Design). https://doi.org/10.1145/2333660.2333664

[5] Yixin Luo, Sriram Govindan, Bikash Sharma, Mark Santaniello, Justin Meza, Aman Kansal, Jie Liu, Badriddine Khessib, Kushagra Vaid, and Onur Mutlu. 2014. Characterizing Application Memory Error Vulnerability to Optimize Datacenter Cost via Heterogeneous-Reliability Memory. In Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '14). IEEE Computer Society, USA, 467–478. DOI: https://doi.org/10.1109/DSN.2014.50

[6] S. Aritome, R. Shirota, G. Hemink, T. Endoh and F. Masuoka, "Reliability issues of flash memory cells," in *Proceedings of the IEEE*, vol. 81, no. 5, pp. 776-788, May 1993.