Memory Write Power Conservation Trade-Offs in Bit-Cell Write Circuits Using MTJ, SHE, and STT Designs

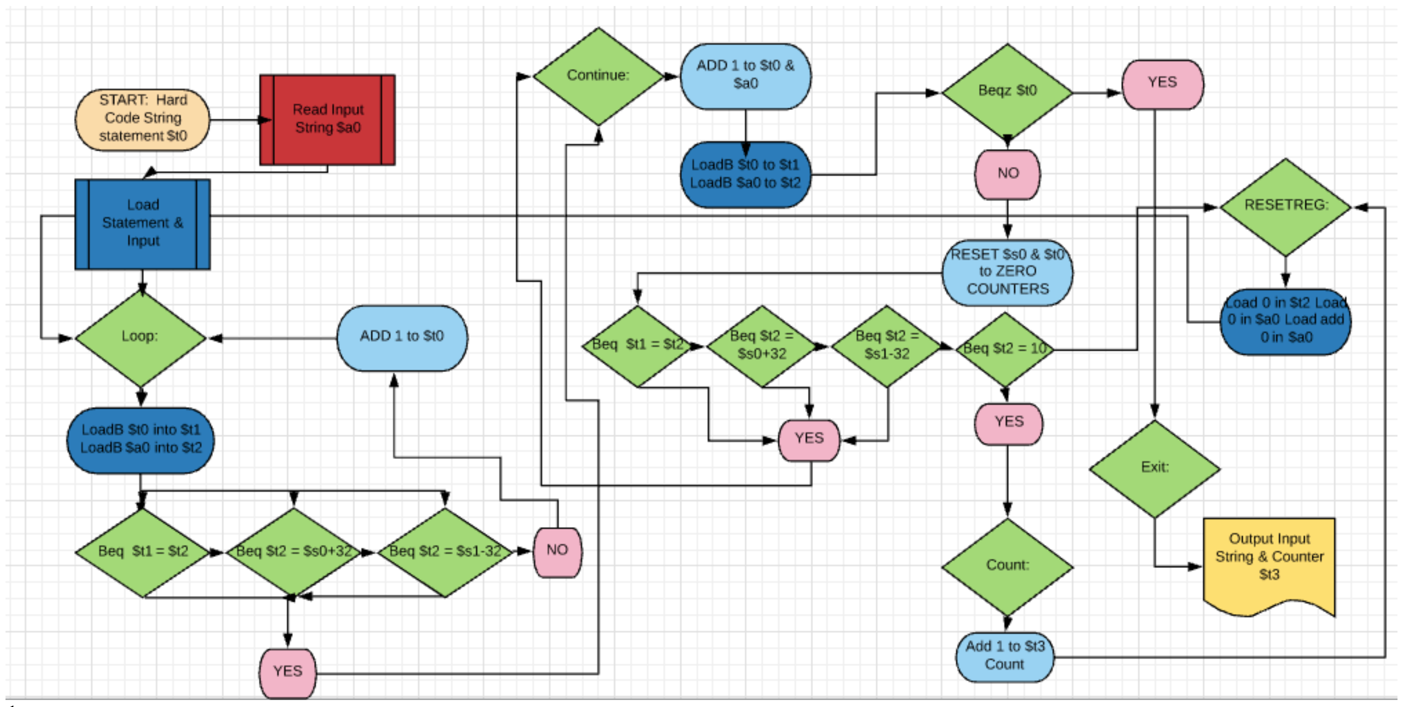
Daniel Patrick Warner
Department of Electrical and Computer Engineering
University of Central Florida

Orlando, FL 32816-2362

***Abstract*— In this paper, a MIPs assembly code written in MARS 4.4 has been analyzed by it's instruction count. The memory write operation has been chosen to evaluate which bit-cell write circuit design will a best fit to conserve power consumption. Different models of write circuits may conserve the most power but a trade-off must be made for an outstanding result. Some bit-cell designs analyzed include SHE-based magnetic random access memory (MRAM), (spinning Hall Effect) SHE-MTJ, an introduction to transmission gates and their uses help determine which circuit should best keep track of the written data to memory. It was concluded that the spin transfer torque (STT) mechanism made the cell more reliable but used the most power while the SHE-RMEM saved would conserve the most power with a trade-off of switching reliability**

Keywords—Bit-Line, Word-Line, Ferromagnetic, Hall-Effect, Magnetic Tunnel Junction (MTJ), Bit-Cell, Mirror-Write Circuit, spin transfer torque (STT)

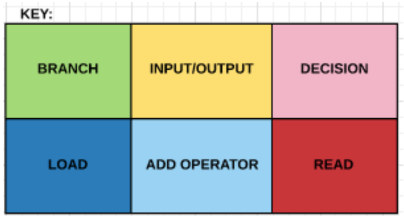
Figure 1: Flow Chart of assembly Code



I. PROGRAM DESIGN

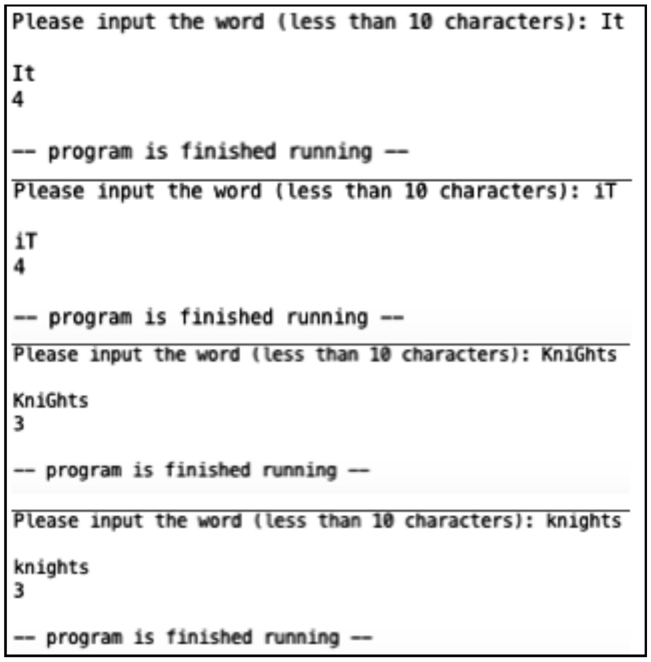
The project code design began with hard coding an input string. This string can be created with as many characters as the memory can withstand. In this particular case, the hardcoded string was a paragraph of about 15 sentences. Any prompt being used in the program was also hard coded. The MIPs .data file contains the strings that were hardcoded before the .text part of the inserted code. This was the program's instruction counter has the hardcoded information already inside the program. When asked to load the addresses of the .data strings, Mars 4.4 knows where to go to retrieve the array in the memory. Additionally, the string's first character acts as the location of the entire string. Assembly language focuses on few elements at a time; consequently, the address of the hard coded string in the .data file must be loaded one address byte at a time using a counter for each individual character of the paragraph. The objective of this code is to search the number of occurrences of a specific word entered by the program user. In order to accomplish this, the program must take in the string (word) inputted from the user and compare it to the hardcoded paragraph that has previously been entered into the .data section. Counters are used to keep track of what array element is being considered as a match to either the hardcoded string or the string entered by the user. In this program, the .data string will be compared to the entered string (word). This way, the program will conserve energy by filtering through a shorter string repetitively rather than a longer string. No matter which input character is being examined, if the corresponding character in the paragraph does not match the ascii value of the entered string, then the input string must be reset back to the beginning character and sent through again. The paragraph's characters could be either a lower case or upper case letter. If so, the program should be able to identify these characters and accept them regardless whether a lower or uppercase value has been found. This aspect shouldn't effect the output of occurrences. At the end of the user's input, the user will have entered the 'enter' key on the key board. The enter key and the spacebar key are ascii values that are to be considered in the code design of this project. If one of theses keys are found in the input string, the last character of the input word has been previously entered resulting in an occurrence of the inputted word. The same ending characters exist for the paragraph that has been hardcoded, minus the spacebar. If the NULL character is found in the hardcoded paragraph, the program should then terminate.

Considering the objective of the designed assembly code above, the program must contain a number of loops/ labels. In design, there must be a minimum of four labels. The program shouldn't end until the paragraph's characters have all been examined. The first loop should check for an array element match between the first element of the input array and the first element of the paragraph. If a match, a 'continue' loop should be executed. This continue loop seen in Figure 1 shows how both array elements are adjusted to the following element. If the ascii values do not match, only the paragraph element should move to the next byte and check back to see if a



character matches to the word inputted. There exists the first

Assembly Code Output:



```
Please input the word (less than 10 characters): It

It
4

-- program is finished running --
Please input the word (less than 10 characters): iT

iT
4

-- program is finished running --
Please input the word (less than 10 characters): KniGhts

KniGhts
3

-- program is finished running --
Please input the word (less than 10 characters): knights

knights
3

-- program is finished running --
```

two loops in the flow chart, 'Loop' and 'Continue'. If the word is found and the input string comes to an end, the address register values must be reset and sent back to the starting loop

Figure 2: Outputs of code

after adding 1 to the word occurrence counter, this action demonstrates the last two loops inside of the assembly code. Label 'Resetreg' resets the registers back to zero making it possible to check for the next array element in the input. Simultaneously, the 'counter' label waits inside of the 'Resetreg' block to add an occurrence the input. Finally, when the .data section string comes to a terminating NULL character, the program's instruction counter is branched to the 'Exit' label. Once the 'Exit' label is executed, the program prints out the entered string followed by the amount of occurrences of that particle string. To test the program I have selected the following user input strings: KniGhts, knights, It, and iT. These inputs are chosen to make certain that each word entered, regardless of the upper/lower case, has the same output occurrence values.

II. Memory Bit-Cells

Memory write circuits are created using different models to manipulate data pulled from the memory. Computers can only compare two bits at a time and therefore a bitwise voltage analysis between the memory machine code and the state of a circuit made of semiconducting materials must determine one of three conditions: either to hold, read, or write the value [5]. When pulled data, regardless whether it's received from the user or the memory, enters this cell, one of the options is determined. Advancement of technologies, cost analysis, durability, precision, speed, and energy consumption factor into circuit selection. Some energy reduction write circuit schemes are used with the spin Hall Effect (SPE) - based magnetic tunnel junctions (MTJ) and a transmission gate (TG) [1]. The circuits using this model reduce the amount of energy typically used in correspondence of the amount of current being distributed throughout the circuit. The MTJ consists of two ferromagnetic layers separated by a thin oxide barrier. This barrier allows very little current to flow through saving the circuit power [1].

Examples of circuits using magnetic tunnel junctions include a Conventional 2T-1R bit cell, this bit cell uses the MTJ between the read and write bit lines operating the two separately with by not sharing the same current through the MTJ [1]. Develop current mirror write circuits are also introduced. These current mirror circuits change the circuit outputs/flip back and forth at different current levels. The output may change at one given input current but cannot change back at the same current level isolating the output operations to alternative inputs. With the clock rate changing the currents may also change and the difference could result in faulty behavior if the transistors do not turn on due to low current supply. For reliability purposes of these current mirror write circuits, the transistors may have to be wider for more current to flow. More referenced current must flow through a device to keep it speedy and reliable, there is a trade-off of power consumption.

To solve the above problem of power consumption another design, Energy-aware write circuit that was inspired by the designs proposed by Ben-Romdhane, were introduced [3]. This circuit was built with a clock switching mechanism that creating an, almost, perfectly symmetric behavior outputs. Fewer transistors were used in the making of the bit cell write circuit. A problem found inside of the circuit was that the circuit through the transistors are smaller than the critical current at times. This creates faulty switching when the current is so low.

These circuits have different trade-offs in relation to speed, energy, and reliability. Every circuit's input values come from word lines (WL) and bit-lines (BL). If the word length is 8 bits wide, then there are 8 word-lines. If there exists 8 bit-lines and 8 word-lines, then, where these signs cross one another exist a bit-cell at the junction. Read or write operation is determined at these junctions. The data is collected from each crossing from these circuit hardware lines and if 'read' is determined through the bit-cell semiconducting circuit, the signal is sent to data lines to be stored. Adversely, if 'write' is determined, the data at the cell junction is overwritten or kept, then stored to be read later or outputted later on.

III. Results and Discussion

In calculation of the energy consumption of the MIPS assembly code the following energies per instruction were used:

Table 1

Energy Consumption for a Single bit-cell write operation in the designs provided through reference	
Design	**Energy Consumption For Each ALU write Instruction (fJ)**
SHE-MTJ	300
SHE-MRAM	280
TG SHE-MTJ (TG based write circuit)	360
STT-MRAM (highest)	420

- *ALU = 1 fj*
- *Branch = 3fj*
- *Jump = 2 fj*
- *Memory = Read Energy (1fj) + Write Energy(refer to Table1)*
- *Other = 5 fj*

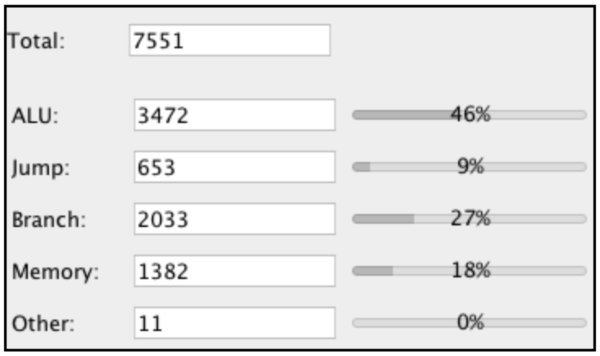
The input string sent through the MIPS assembly program to gain the instruction statistics was "Knights."

IV. Conclusion

The project designs observed have trade-offs between reliability and energy consumption. A desired switching behavior was introduced to save power using the spinning Hall Effect through electro magnetic physics. MOSFET transistors inside of the bit-cell circuits are great power conservators by eliminating gate currents. Nonetheless, with the current values being lower, the transistor may not activate and the switching mechanism might fail. This same faulty behavior as to whether the switching occurs may result from the use of the MTJ due to the oxide barrier eliminating power consumption yet not switching the outputs. Transmission gates added into the MTJ circuit creates the full switching effect solving the reliability issues, but with these gates factoring into the cell the power consumption increasing again due to a lack in leakage currents. The most conservative approach is to use the Spinning Hall Effect MTJ to reduce power consumption. Spinning torque effect was examined to accurately switch the circuit and a is a more reliable approach of control. This technique of torque switching uses an abundant amount of power and is not a reasonable trade-off.

References

1. R. Zand, A. Roohi and R. F. DeMara, "Energy-Efficient and Process-Variation-Resilient Write Circuit Schemes for Spin Hall Effect MRAM Device," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 9, pp. 2394-2401, Sept. 2017.
2. A. Aziz, W. Cane-Wissing, M. S. Kim, S. Datta, V. Narayanan and S. K. Gupta, "Single-Ended and Differential MRAMs Based on Spin Hall Effect: A Layout-Aware Design Perspective," *2015 IEEE Computer Society Annual Symposium on VLSI*, Montpellier, 2015, pp. 333-338.
3. N. Ben-Romdhane, W.S. Zhao, Y. Zhang, J.O. Klein, Z.H. Wang, D. Ravelosona, "Design and analysis of racetrack memory based on magnetic domain wall motion in nanowires," In *Proceedings of the 2014 IEEE/ACM International Symposium on Nanoscale Architectures*, pp. 71-76, 2014.
4. S. K. Gupta, S. P. Park, N. N. Mojumder, and K. Roy. "Layout-aware optimization of STT MRAMs," In *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 1455-1458. EDA Consortium, 2012.
5. Singh, Sangeeta and Lakhmani , Vikky. "Read and Write Stability of 6T SRAM," International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE), Volume 3, Issue 5, May 2014



Total:	7551	
ALU:	3472	46%
Jump:	653	9%
Branch:	2033	27%
Memory:	1382	18%
Other:	11	0%

Figure 2: Dynamic Instruction Statistics

Table 2

Total Energy Consumption for the Assembly Program Using designs provided by Reference	
SHE-MTJ	217543 fj
SHE-MRAM	203723 fj
TG SHE-MTJ (TG based write circuit)	259003 fj
STT-MRAM	300463 fj

The memory instruction set determines the total number of read and write instructions. Read instructions are only taking 1 femto-joule, in MIPS assembly language there are only so many registers that can be read from the memory. Here, it is assumed the amount of write operations is equivalent to the amount of memory read operations for simplicity of determining which design will reduce the most power consumption. Additionally, the mirror circuits discussed have a clock cycle that is mostly symmetrical on both the bit-line and the word-line. Once a memory location has been read with limited space in MARS 4.4, the location is usually over written or written using the operational write instruction. Especially in our case of going through a long string.