Analysis of Various Memory-Read Sense Amplifiers Based Upon Total Energy Consumption

Anish Umashankar
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

***Abstract*—**The purpose of this paper is to test our understanding data, address, memory contents, strings and to evaluate some of the fundamental metrics for the 'Memory Read Operation'. The objective of the program used in the analysis is to ask the user to input a word with less than 10 characters to which the program will then output how many times that word occurred in the sample string given. To implement this in MARS, I used a basic looping structure in which the program goes through a checklist: Checking to see whether the hard-coded string character is at the null character, if the user input is at its 10th character, and whether the characters of the input word and string match (regardless of case). Analyzing the energy consumption of the four given design types (EASA, VISA, PWSA, and BVSC), it is shown that the EASA design has the lowest total energy consumption at 77671.26 fJ when compared to VISA, PWSA, and BVSC.

Keywords—MTJ (Magnetic Tunnel Junction), Energy Aware Sense Amplifier (EASA), Variation Immune Sense Amplifier (VISA), Preread and Write Sense Amplifier (PWSA), Body-Voltage Sensing Circuit (BVSC), Sense Amplifier (SA), Sense Margin (SM), Transmission Gates (TGs) .

I. Project Design

The basic objective of this program is to ask the user to input a word with less than 10 characters to which the program should output throw many times that word occurred in the sample string given. To achieve this goal, you must first prompt the user to input their word and then use a syscall with the system code 8 format to read a user string. Next load registers $t1 and $t2 with the Input and String respectively and set the counter for the input occurrences ($t0) to 0. Next the looping structure of the program goes as follows: load the current character of the Input and String into $t3 and $t4 respectively, check to see if character in $t4 is at '\0'(if yes then print the word that the user inputted along with the final occurrence counter value of that word), if no then check to see if the user input character is at the 10th character (if yes, meaning that the entire word that the user inputted has been found, then increment occurrence counter, reset the address of the input string, and restart loop), if no then check to see if the current character of user input and string match, also change case of character in input by adding and subtracting 32 to register $t3 (if yes then increment string and input by 1 and restart loop), if no then reset input character to beginning, increment the string character by 1, and then restart the loop.

To see if the program is running as intended, 3 main input variants need to be tested. The first variant that needs to be tested is a basic user input to see if the program counts the word correctly and outputs the user input word. This is achieved by using the basic input 'It', which gets outputted correctly with the correct occurrence counter value. The second variant is a mix of both upper and lowercase ('kNiGhT') and serves to test whether the program can constantly convert between upper and lowercase for each character in the user input and count the occurrences properly. The program outputs the user input correctly while also outputting the correct occurrence counter value. The last variant to test is a word that doesn't occur in the user hard-coded string to see if the program prints the correct

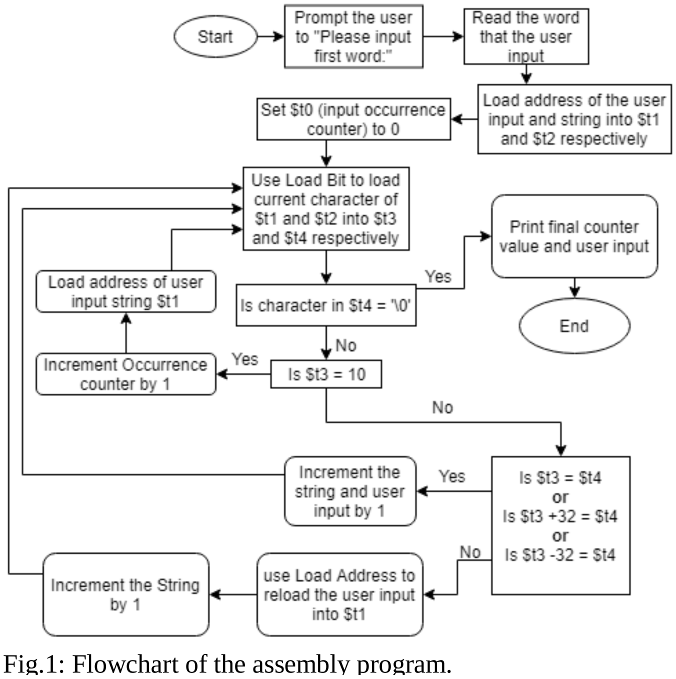
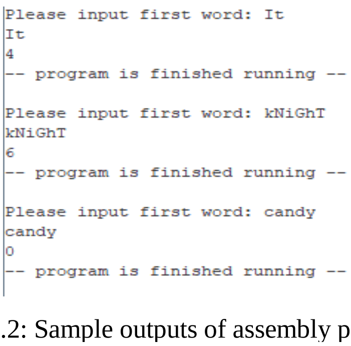

Start
Prompt the user to "Please input first word:"
Read the word that the user input
Load address of the user input and string into $t1 and $t2 respectively
Set $t0 (input occurrence counter) to 0
Use Load Bit to load current character of $t1 and $t2 into $t3 and $t4 respectively
Is character in $t4 = '\0'
Yes
Print final counter value and user input
End
No
Is $t3 = 10
Yes
Increment Occurrence counter by 1
Load address of user input string $t1
No
Is $t3 = $t4 or Is $t3 +32 = $t4 or Is $t3 -32 = $t4
Yes
Increment the string and user input by 1
No
use Load Address to reload the user input into $t1
Increment the String by 1

Fig.1: Flowchart of the assembly program.



```
Please input first word: It
It
4
-- program is finished running --

Please input first word: kNiGhT
kNiGhT
6
-- program is finished running --

Please input first word: candy
candy
0
-- program is finished running --
```

Fig.2: Sample outputs of assembly program.

occurrence counter 0. This is done by using the word ‘candy’, to which the program outputs the correct occurrence counter value.

II. Memory Bit-Cells

From the program design description above, there are four main bit-cell read operation designs to consider: Energy Aware Sense Amplifier (EASA), Variation Immune Sense Amplifier (VISA), Preread and Write Sense Amplifier (PWSA), and the Body-Voltage Sensing Circuit (BVSC). The underlying fundamental concept of these designs comes from the Sense Amplifier (SA) which is a part of the read circuitry whose goal is to sense the low power signals and amplify them into something that the logic of the circuit can understand. In addition, the Magnetic Tunnel Junction (MTJ) is a major element of this process, as it is crucial to the read process of hard disk drives [6].

The first design that is being analyzed is the Energy Aware Sense Amplifier (EASA). This design uses devices called Transmission Gates (TGs), to provide energy efficient switching to reduce the leakage energy or problems occurring with Process Variance (PV) [1]. PV has an inversely proportional relation to the reliability of a program, as PV increases the reliability of said program will decrease due to the variation in a program’s output. The disadvantage to this design come in play when you add TG’s on the path of MTJs, because placing these components in a line as such will only serve to increase the resistance of this path (series), which could result in read errors and an increase to PV [1].

The second design being analyzed is the Variation Immune Sense Amplifier (VISA). This design is very similar to the EASA design, but it utilizes inverters and uses a parallel configuration for the MTJs. Due to the configuration of the inverters in unison with thee TGs, when the inverter outputs are low, the TGs will be off [1]. This will lower the leakage energy. The parallel configuration will lower the resistance of the MTJs as they will not longer be additive, but an inverse of the addition of the MTJ’s inverse (parallel resistance) [1]. Just like how the EASA series configuration increased resistance which increased read errors and PV, lowering the resistance will lower the amount of read errors and PV [1].

The third design being analyzed is the Preread and Write Sense Amplifier (PWSA). The fundamental idea of this design is to combine both read and write functions into a circuit to improve power efficiency [2]. This specific PWSA is for High-Speed MRAM. MRAM in the recent years has been praised by its proponents as the soon-to-be dominant memory type, even over flash Ram and DRAM [5]. This design uses the voltage-controlled magnetic anisotropy effect for switching, which lowers the write power and transistor size of the design by more than 10x [2]. In addition to this, the circuit receives a 2x larger sensing margin (SM).

The final design being analyzed is the Body-Voltage Sensing Circuit (BVSC). This design aims to mitigate issues when it comes to scaling CMOS and MTJ devices [3]. It does this by using a body-voltage sensing scheme to achieve high Sense Margin (SM) with a short sensing time [3]. Based upon results of [4] using a 2-bit in a 1-Mb memory can increase read margin by 3x. This can be very useful as scaling of CMOS and MTJ devices continues and makes reading of STT-RAM difficult [4].

III. Results and Discussion

Given that the dynamic instruction count can be found using **MARS4.5➔Tools➔Instruction Statistics** and the following energy consumption per instruction values:

1) *ALU = 1 fJ*
2) *Branch = 3 fJ*
3) *Jump = 2 fJ*
4) *Memory = Read Energy (Refer to Table I) + Write Energy (50fJ)*
5) *Other = 5 fJ*

Using the information given above and Table I, one can calculate the total energy consumption of the assembly program using the designs provided. Using the first test input *‘It’* the dynamic instruction counts listed by Instruction Statistics is 8699. The 3706 ALU instructions require only 1fJ per instruction, therefore their energy consumption is 3706 fJ. The 630 Jump instructions require 2 fJ per instruction, increasing the net energy consumption to 4966fJ. The 3095 branch instructions need 3fJ per instruction, again increasing the net total to 14251 fJ. The 6 other instructions require 5 fJ per instruction increasing the net total to 14281fJ. The memory energy is equal to the Read Energy for the specific design (Values from Table I) added to the Write Energy (50 fJ) and finally this energy consumption per instruction value is multiplied to the amount of memory instructions which is 1262. The calculated memory energy consumption values are 63390.26 fJ, 65447.32 fJ, 108532 fJ, and 309821 fJ for the EASA, VISA, PWSA, and BVSC respectively. When each individual energy consumption is added to the net total value 14281 fJ the total energy consumption found. The total energy consumption for the assembly program using the designs provided are listed in Table II.

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

Design	Energy Consumption For Each Bit-cell’s Read Operation
EASA [1]	0.23 fJ
VISA [1]	1.86 fJ
PWSA [2]	36.0 fJ
BVSC [3]	195.5 fJ

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

Design	Total Energy Consumption
EASA [1]	77671.26 fJ
VISA [1]	79728.32 fJ
PWSA [2]	122813.00 fJ
BVSC [3]	324102.00 fJ

IV. Conclusion

The purpose of this paper was to use a simple assembly code to test whether the total energy consumption of 4 different designs EASA, VISA, PWSA, and BVSC. Through this process many topics had to be analyzed closely, such as reliability, energy efficiency, Sense Margins (SMs), Process Variation (PV), fundamentals of Sense Amplifiers (SA), and Instruction Statistics of the assembly code given a specific input. Using all these topics and Table II, it is concluded that EASA is the most energy efficient program due to it having the lowest total energy consumption (77671.26 fJ) compared to the other 3 designs.

References

[1] S. Salehi and R. F. DeMara, "Process variation immune and energy aware sense amplifiers for resistive non-volatile memories," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, 2017, pp. 1-4.

[2] H. Lee *et al*., "Design of a Fast and Low-Power Sense Amplifier and Writing Circuit for High-Speed MRAM," in *IEEE Transactions on Magnetics*, vol. 51, no. 5, pp. 1-7, May 2015.

[3] F. Ren, H. Park, R. Dorrance, Y. Toriyama, C. K. K. Yang and D. Marković, "A body-voltage-sensing-based short pulse reading circuit for spin-torque transfer RAMs (STT-RAMs)," Thirteenth International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, 2012, pp. 275-282.

[4] F. Ren, H. Park, C. K. K. Yang and D. Marković, "Reference Calibration of Body-Voltage Sensing Circuit for High-Speed STT-RAMs," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 60, no. 11, pp. 2932-2939, Nov. 2013.

[5] Huai, Yiming. "Spin-transfer torque MRAM (STT-MRAM): Challenges and prospects." AAPPS bulletin 18.6 (2008): 33-40.

[6] Zhu, Jian-Gang Jimmy, and Chando Park. "Magnetic tunnel junctions." Materials today 9.11 (2006): 36-45.