Energy Consumption and Efficiency with Look-Up Table based Implementation of ALU

Deborah Campbell
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

Abstract— **The consumption of the overall energy can be executed using the Look-up tables (LUTs) and Arithmetic Logic Unit design (ALU), along with the implementation of its efficiency. The energy consumption may differ in accordance to the design and modification of the assembly code significantly. The program used for this paper, was constructed to prompt the user for the input of a word stored in a string function in order to find its equivalent in the hardcoded string. The Instruction Statistics & Counter tools are then used to measure the type of instructions and energy consumed after the output has been initialized.**

Keywords*— *Look-up Table, Energy consumption, MOSFET, FPGA, MRAM, Non-Volatile Memory, Arithmetic Logic Unit.

I. Project Design

The MIPS assembly language code constructed was used to convey/ prompt to the user to input a word in order to find the word stored in the hardcoded string. The number of occurrences computed in the program will then be counted by the output of the word, with the purpose being to input a word with less than 10 characters. To ensure the program will be able to find the word inputted by the user, restrictions were implemented (i.e. should not be case sensitive). The word, characters, variables, and strings used to generate the input and output are stored in the registers and memory respectively.

The purpose of the project was to enhance the understanding of data, address, memory contents, and strings. The Byte by Byte comparison is used to differentiate the word and string, while the performance of the program relies on loops and syscalls as significant points. To check for adequate match, add 32 for lowercase, branch 96 and 65 for the uppercase. If it's a match, jump to 'match', add one to both string and move to the next letter, then jump to 'loop', and check the new letters for accuracy. If it doesn't match, add one to the 'string', reload the term and jump back to loop to check the next character. Afterwards, jump to exit and print the number from the math counter at the end of the string.

To test the code five different inputs were used; lowercase, lowercase and uppercase; a word and a number separated by a space. The inputs tested were Knight. Knight, KNIGHT, KnIgHt, and KnIGhT. The first input tested if the program could handle an uppercase search word, the second demonstrated only lowercase, the third tested for all uppercase, and finally the forth and fifth tested for a combination of lowercase and uppercase

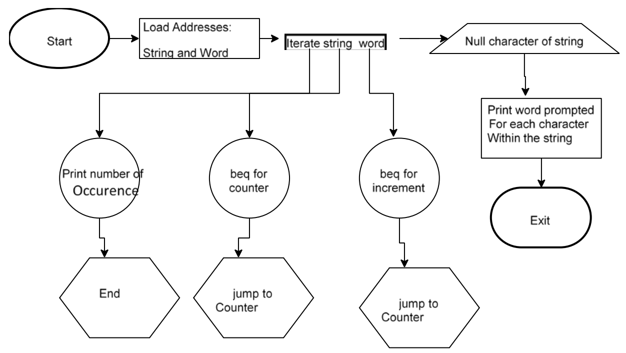
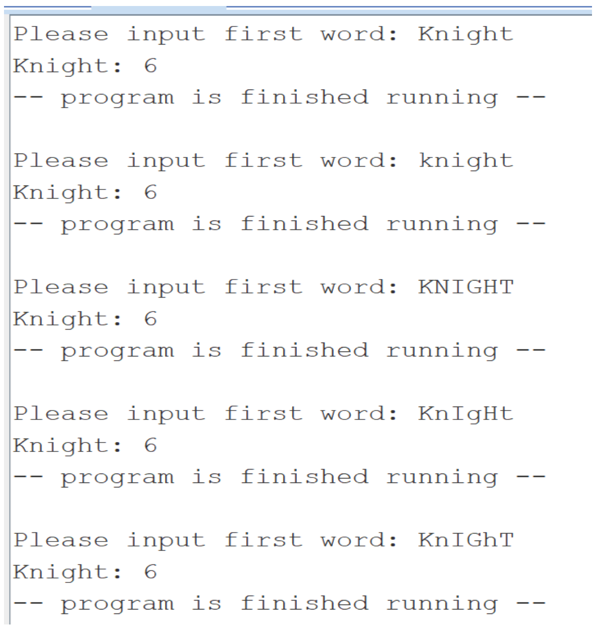


Fig.1: Flowchart of the assembly program.



```
Please input first word: Knight
Knight: 6
-- program is finished running --

Please input first word: knight
Knight: 6
-- program is finished running --

Please input first word: KNIGHT
Knight: 6
-- program is finished running --

Please input first word: KnIgHt
Knight: 6
-- program is finished running --

Please input first word: KnIGhT
Knight: 6
-- program is finished running --
```

Fig.2: Sample outputs of the assembly program.

letters. It can be concluded based on the hardcoded string, the word "knight" no matter its format occurs six times, as shown in Fig.2.

II. Look-Up Table Circuit

The Look-Up Table (LUT) is a necessary computation to stimulate and calculate performances without the requires of the Boolean gates. The structure of a LUT used in the design of ALU instructions assist in the performance of memory in the reduction of power storage. Functional failure is due to weak signals which increases the chances of loss.

In its growth, as well as other technical issues along with it, the MJT technology offer a solution to the power in their SRAM predecessor. The LUT has its advantages and disadvantages in its design depending on how its been programmed. With many of its major flaw being in how its usefulness is affected, a LUT knows the combination which it must perform and how the multiplication of numbers in the Boolean gates must return to correct values. The adequacy of the LUT designs lies in its continuous and extensive research in different aspect for more accurate functionality with MOSFET being the main component of majority of LUTs.

III. Results and Discussion

Using the MARS4.5 Instruction Statistic tool, the total energy consumption was acquired after running the assembly code created. The energy consumption is presumed to be the same, which was then applied to execute the calculations below. Here we have the ALU instruction given with the varying values provided in Table I. The other energy consumption per instruction values are shown below:

1) *ALU = Refer to Table I*
2) *Branch = 3 fJ*
3) *Jump = 2 fJ*
4) *Memory = 200 fJ*
5) *Other = 5 fJ*

Table I: Energy consumption for a single ALU Instruction in the designs provided in [1-3].

Design	Energy Consumption For Each ALU Instruction
STT C-LUT [1]	170.88 fJ
SHE C-LUT [1]	184.08 fJ
[2]	33.7 fJ
[3]	134 fJ

The total energy consumption is calculated using the number of instructions for the instruction type. Which is then multiplied by the corresponding energy as per the instruction value considering the energy acquired from the Instruction Statistics tools. The dynamic instruction counts (DIC) acquired from the assembly code are:

1) $ALU_{Instr.} = 4586$
2) $Branch_{Instr.} = 3306$
3) $Jump_{Instr.} = 1263$
4) $Memory_{Instr.}$ 1923
5) $Other_{Instr.} = 638$

The calculation for the total energy consumption can be derived from the formula below:
Total Energy Consumption = $(ALU*ALU_{Instr.})+(Jump*Jump_{Instr.})+(Branch*Branch_{Instr.})+(Memory*Memory_{Instr.})+(Other*Other_{Instr.})$

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

Design	Total Energy Consumption
STT C-LUT [1]	1183.889pJ
SHE C-LUT [1]	1244.424pJ
[2]	554.782pJ
[3]	1014.758pJ

IV. Conclusion

New technology has seen major improvements due to rapid process in recent years. This has been possible with its swift run and cost-efficient availability. However, though the efficiency is commendable there hasn't been much improvement in the power consumption.

References

[1] Soheil Salehi, Ramtin Zand, Ronald F. DeMara. 2019. Clockless Spin-based Look-Up Tables with Wide Read Margin. In Great Lakes Symposium on VLSI 2019 (GLSVLSI '19), May 9–11, 2019, Tysons Corner, VA, USA. ACM, Washington, DC, USA, 4 pages.

[2] B. Khaleghi and H. Asadi, "A Resistive RAM-Based FPGA Architecture Equipped With Efficient Programming Circuitry," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 7, pp. 2196-2209, July 2018.

[3] Zhao, Weisheng, Eric Belhaire, Claude Chappert, François Jacquet, and Pascale Mazoyer. "New non‐volatile logic based on spin‐MTJ." physica status solidi (a) 205, no. 6 (2008): 1373-1377.

[4] Ramtin Zand, Arman Roohi, Ronald F. DeMara, "Energy-Efficient and Process-Variation-Resilient Write Circuit Schemes for Spin Hall Effect MRAM Device", Very Large Scale Integration (VLSI) Systems IEEE Transactions on, vol. 25, pp. 2394-2401, 2017, ISSN 1063-8210.

[5] Sungsoo Ha, Klaus Mueller. "A Look-Up Table-Blased Ray Integration Framework for 2-D/3-D Forward and Back Projection in X-Ray CT," in IEEE Transactions on Medical Imaging, vol. 37, no. 2, pp. 361-371, August 2017, ISSN 0278-0062.

[6] A. Alzahrani and R. F. DeMara, "Process variation immunity of alternative 16nm HK/MG-based FPGA logic blocks," 2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS), Fort Collins, CO, 2015, pp. 1-4.