Application of Lookup Tables and their implementation FPGAs

Nicholas Castillo
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

***Abstract*— Compare the implementation of a Lookup-Table (LUT) to a traditional two-input Boolean gate when designing/performing ALU instructions. The purpose of this paper is to evaluate the implementation of LUT based ALU in various in various cases. The paper focuses on the application of Lookup Tables and their implementation toward Field Programmable Gate Arrays (FPGAs). FPGA uses a LUT to implement any logical function. The type of implementation is dependent on certain factors depending on the application. The program designed in MIPS uses an algorithm that sifts through a sample string that has been hardcoded, and takes the word being searched (input) and counts the instances/occurrences of the word (output). The program is not case sensitive and a detailed flow chart of the design (Fig.1) as well as calculation for total energy consumption (Table II) are provided below.**

Keywords—FPGA, ALU, LUT, ROM, PLD, Logic Cells

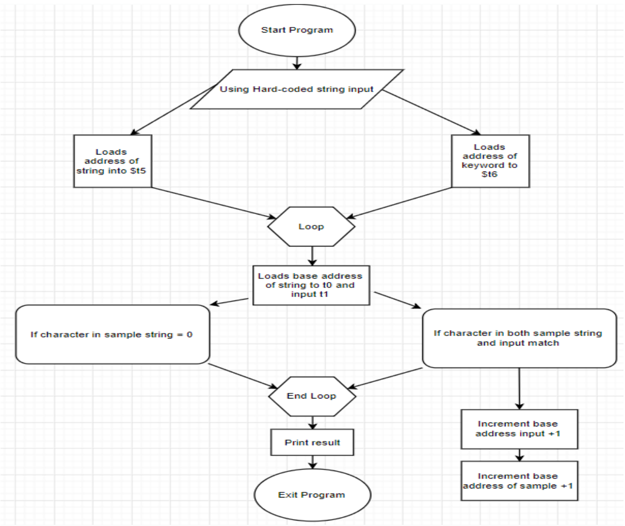
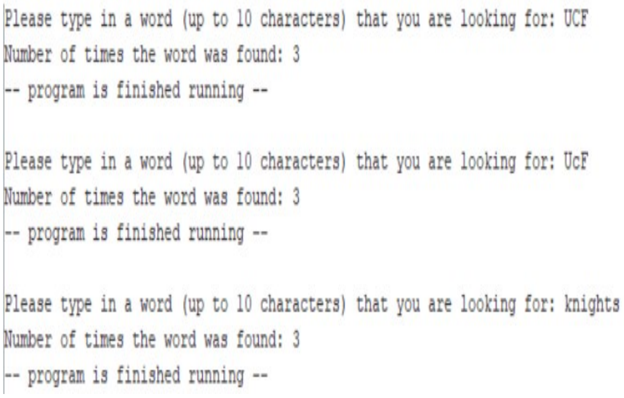

Start Program
Using Hard-coded string input
Loads address of string into $t5
Loads address of keyword to $t6
Loop
Loads base address of string to t0 and input t1
If character in sample string = 0
If character in both sample string and input match
End Loop
Print result
Exit Program
Increment base address input +1
Increment base address of sample +1

Fig.1: Flowchart of the assembly program.



```
Please type in a word (up to 10 characters) that you are looking for: UCF
Number of times the word was found: 3
-- program is finished running --

Please type in a word (up to 10 characters) that you are looking for: UcF
Number of times the word was found: 3
-- program is finished running --

Please type in a word (up to 10 characters) that you are looking for: knights
Number of times the word was found: 3
-- program is finished running --
```

Fig.2: Sample outputs of the assembly program.

I. Introduction

A. Project Design

The program was designed to take a sample string which was hardcoded in the beginning, then to take an input as a word (10 characters max from user), and search/match how many occurrences the word appears in the sample string and display the result. To do this, first we store the sample string, and then prompt the user for the input to count occurrences. We then load the address and bytes of the string and input into two different registers. This enables the program to focus on individual characters when matching/comparing the sample and input. The conditionals of the loop help narrow down the search. If the character byte of the string is 0 ends loop, next loop searches if characters in both sample and input match. If it is a match then increment the base address input by one and then increment the base address of the string by 1 to check then next character and the following words. The ascii values +32 and -32 are added to the input character bytes to account for both lower case and uppercase incidences (not case sensitive). The address of the input is loaded back into the register so it resets, and the address of the string will print the result (+1) for each instance.

B. Test Cases

The inputs I chose can be found in Fig.2. The inputs I chose were varying in word lengths which let e know that the program worked for multiple cases. They were also varying in case sensitivity, some were uppercase, some lowercase, and some mixed, to see if results held true. The final reason I chose these test cases were because I was able to count by hand to compare my results.

II. Look-Up Table Circuit

A lookup table can be defined as a table that is filled with values that can represent multiple inputs or outputs, which can be referenced/addressed (looked up). Lookup tables are widely used to define data for nonlinear expressions. A lookup table allows for approximations of a function/equation which can reduce number of complex calculations (simplifying operations). Programmable Logic Devices (PLDs) are a type of integrated circuit designed to perform logic functions. [8] A field programmable gate array (FPGA) is a type of PLD which generally implements a lookup table (LUT) to realize functions. A Look-Up table that has been programmed into an FPGA can look up what the output should be for a given inputs. As oppose to hard-wiring gates of an actual circuit LUTs can search for data in its memory to the possible outputs for given inputs which can save a lot of time and money. An FPGA can perform Boolean algebra by using LUTs. LUTs use a similar concept to truth tables to relate the input and output. [6] The number of inputs for each lookup table depends on the complexity of the FPGA. [5]

Arithmetic Logic Unit Function (ALU) supports several different functions such as adder, subtractor, increment, as well as logical functions such as AND, OR, and XOR. A LUT can be designed to perform functions that could be otherwise difficult with discrete logic gates. A LUT becomes more useful with increasing complexity of functions. [7] For example designing a circuit using a ROM would require a lot of addressable memory location, which would require a lot of programming and leaves a lot of room for error. Each individual computation of binary inputs could output an incorrect value or other complications. However, by using a LUT there are far more possibilities which aren't limited by the constraints of logic gate functions. [7] Programmable Logic Devices (PLDs) are a type of integrated circuit designed to perform logic functions. [8]

Different approaches depend on specifications of the complexity and length of the operation at hand. LUT provide a low-volume cost, meaning multiple test cases and instances can be run at a reasonable price. [1] LUT enable FPGAs to execute calculations at a fast rate which is beneficiary in lengthy and complex functions. LUT however require a large memory base to cover functions, which means in a scenario with simple or short functions the LUT will have a lot of variables and instances not being used or addressed.

III. Results and Discussion

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

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

Design	Total Energy Consumption
STT C-LUT [1]	1,008,139.6 fJ
SHE C-LUT [1]	1,066,937.08 fJ
[2]	408,471 fJ
[3]	836,360 fJ

IV. Conclusion

With technology always moving forward boundaries need to be pushed and complications need to be minimalized. There is more than one solution to a problem. LUTs and Boolean gates are both good choices when working with ALUs. When deciding on aproach it is important to focus on efficiency and account for total costs, energy consumption, and time. During this report I attained a much better understanding of how LUTs work. Reading an article on ROM helped put in perspective how much addressable memory is required for certain circuits. I learned step by step how a PLD logic cell implements an ALU function. I've also learned the logic component of an FPGA, such as look-up tables and flip-flops and their role.

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] Francis, "A tutorial on logic synthesis for lookup-table based FPGAs," *1992 IEEE/ACM International Conference on Computer-Aided Design*, Santa Clara, CA, USA, 1992, pp. 40-47.

[5] A. DeHon, "The density advantage of configurable computing," in Computer, vol. 33, no. 4, pp. 41-49, Apr 2000. doi: 10.1109/2.839320

[6] "Tutorial - Performing Boolean Algebra inside an FPGA using Look-Up Tables (LUTs)." Tutorial - Boolean Algebra using Look-Up Tables (LUT). 08 Apr. 2019 <https://www.nandland.com/articles/boolean-algebra-using-look-up- tables-lut.html>.

[7] "LATEST".All About Circuits

www.allaboutcircuits.com/textbook/digital/chpt-16/look-up-tables/

[8] "US6476634B1 - ALU Implementation in Single PLD Logic Cell." Google Patents, Google, patents.google.com/patent/US6476634B1/en.