Energy Consumption Analysis of Different ALU Look-up Table Based Circuit Designs

Ivan Bernal Garcia
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

Abstract—The purpose of this paper is to evaluate how different look-up table designs have an effect on energy consumption. For this paper an assembly language code was made that would ask the user for an input word, store it in the form of a string and use it to find matches on a hardcoded string. Once the code went through the string the program returns the input string capitalized with the number of times that input appeared in the hardcoded string. Once the code was completed the Instruction Statistics tool was used to measure the types of instructions used and how many times they were used. These numbers were used to calculate the energy consumption of each design, and design [1] had the best energy consumption out of all the other designs only consuming 267.9 pico-joules.

Keywords— fj (femto joules), LUT (look up table), ALU (algorithmic logic unit), FPGA (field programmable gate array), CMOS, energy consumption, NEMS, SRAM

I. PROJECT DESIGN

The assembly code used takes in a word input for the user and stores it in the form of a string. We store the word in the label ‘term’ that has 10 bytes reserved for cases of different word length, including the enter character that has a value of 10. The term is then printed in all caps by utilizing a loop that turns lower case characters to upper case by subtracting 32 from the characters ASCII value. The string we are searching is loaded to the label ‘string’ and the term is load again so we start off from the first characters. A label called loadterm is used to reload the term every time we either miss or get a match. We jump to label ‘loop’ where we do multiple things, load a byte from both strings, check if term equals 10 (ASCII value for enter) means there’s a word match we jump to ‘loadtermmatch’ to rest the term string and add on to the match counter. If ‘string’ equals zero we have reached the end of the string and proceed to print the results. Plus to check if we have a perfect match, add 32 to check if we have a lower case match, subtract 64 in case its an upper case match, if we get a match from those three options we jump to ‘match’, add one to both string to move to the next letter, then jump to ‘loop’ and check the new letters. If there is no match we add one to the ‘string’ and reload the term and jump back to loop to check the next character. Once we reach the end of the string we jump to exit and print the number from the word match counter. To test the code three different inputs were used, lower case, lower case and upper case, and a word and a number separated by a space. This was done to see if the code work for

Fig.1: Flowchart of the assembly program.

Fig.2: Sample outputs of the assembly program.
different character types, which it did providing the right number of times the input word appeared on the string and printing the outputs in the desired format correctly.

II. LOOK-UP TABLE CIRCUIT

Look-up table (LUT) based ALU are interesting pieces of technology that have a wide range of applications [7]. A look up table has a pre-defined number of inputs and outputs, all possible calculation you want can be simulated without the need for Boolean gates, which require time to perform the calculations and take up a lot of space. A LUT is essentially a large truth table written in memory containing all possible inputs and their outputs. Basically, a LUT knows that 5x5 is just by looking it up that combination, when using Boolean gates, they have to work out what 5*5 is first then return an answer. LUTs are used in FPGAs logic blocks to perform logic functions, they are stored in multiple logic tiles and can be wired together to make larger LUTs [1][5].

Multiple technologies are used to realize LUTs, PT-base MUXes are used to make LUTs that use low voltage. For faster LUTs a spin-MTJ based LUT using CMOS and sense amplifier switches are used however this increases speed, but waste more energy making it the least efficient design [2]. LUTs that use SRAM and CMOS circuits tend to be cheaper but have power leaking due to lots of transistors. Reducing the number of sense amplifiers and sharing transistors to reduce the size of the LUTs compensates for the loss [3][6]. Additionality a combination of CMOS-NEMS switch LUTs have the ability to have near zero power leakage while still being fast [4].

III. RESULTS AND DISCUSSION

Using the instruction statistics tool in MARS we run the code that was created to see how much of each instruction type was used. Here we have the number of each type of instruction used and the corresponding amount of energy each instruction uses:

1) ALU: 3751 intructions
2) Branch: 3109 intructions
3) Jump: 1813 intructions
4) Memory: 1279 intructions
5) Other: 18 intructions

\[ \text{ALU} = \text{Refer to Table I} \]
\[ \text{Branch} = 3 \text{ fJ} \]
\[ \text{Jump} = 2 \text{ fJ} \]
\[ \text{Memory} = 200 \text{ fJ} \]
\[ \text{Other} = 5 \text{ fJ} \]

Table I: Energy consumption for a single ALU Instruction in the designs provided in [1-4].

<table>
<thead>
<tr>
<th>Design</th>
<th>Energy Consumption For Each ALU Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>0.1 fJ</td>
</tr>
<tr>
<td>[2]</td>
<td>1.2 fJ</td>
</tr>
<tr>
<td>[3]</td>
<td>0.6 fJ</td>
</tr>
<tr>
<td>[4]</td>
<td>0.25 fJ</td>
</tr>
</tbody>
</table>

Table 1 contains the amount of energy require per instructions for the different LUT designs design shown in texts [1-4].

For the total energy calculations, we take the number of instructions for an instruction type and multiply by the corresponding energy per instruction and the corresponding design option.

Table II: Total Energy consumption for the assembly program using designs provided in [1-4].

<table>
<thead>
<tr>
<th>Design</th>
<th>Total Energy Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>267.9 pj</td>
</tr>
<tr>
<td>[2]</td>
<td>272.0 pj</td>
</tr>
<tr>
<td>[3]</td>
<td>269.8 pj</td>
</tr>
<tr>
<td>[4]</td>
<td>268.5 pj</td>
</tr>
</tbody>
</table>

IV. CONCLUSION

As fast as technology is improving by becoming faster and cheaper, the area that has remained mostly the same is the power consumption of these devices. There’s a lot of promising technology being developed to improve the efficiency of LUT based ALUs. These improvements made by these different technologies made a small difference in the total energy consumption. I learned how to use the UCF online library resources to find scholar papers on my topic, how spintronic devices function, the modularity of FPGAs, how LUTs work to simulate Boolean functions, realizing assembly code to find words in a giving string. The overall best design was the one proposed in reference [1] only consuming 267.9 pico-joules.

REFERENCES


