Effects of Full-Adder Circuit Design on Assembly Program Total Energy Consumption

Megan Driggers
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

***Abstract*—Energy consumption is a very important factor that should be minimized as much as possible. The total energy consumption of an assembly program as a factor of full-adder circuit design will be examined. The assembly program takes a user input for a word and checks the word frequency within a hardcoded sentence. The program considers the uppercase/lowercase variation between the input word and word found in the sentence to provide a match even if the characters of the word are in different case. The word frequency value is then printed after checking for the word through the whole sentence. Several full-adder circuit designs were evaluated, and the Spin Transfer Torque-Magnetic Full Adder (SST-MFA) design [3] provided the least total energy consumption for this assembly program which was 140.146733 nJ.**

Keywords— Full-Adder, Full-Adder Approximation, Energy Consumption, ALU Design, Conventional Mirror Adder (CMA), Spin Transfer Torque-Magnetic Full Adder (SST-MFA), Assembly Programming, MIPS

I. Project Design

The assembly program begins by allocating 10 characters of space for the user input word using label ‘word’ and hard coding the sentence/paragraph using label ‘sentence’. The purpose of this program is to find the frequency of the ‘word’ within the ‘sentence’. Then, "Please input first word: Knight (or KnIGhT, knight, …):\n" is printed and the user input is stored in the allocated space labeled ‘word’. Next, the program prints "Input Word Frequency within Sentence:". The address of the sentence is then loaded into register $t0, and the address of the input word is loaded into registers $t1 and $t2.

The main loop begins using label ‘Loop’. The sentence character found at address $t0 is then loaded into register $s0, then the word character found at address $t1 is loaded into register $s1. A branch statement is used to determine if the input word character (stored in register $s1) equals 'enter'/ascii value 10 which means that the word has been incremented all the way through. If yes, then that means a word match has been detected within the sentence and jumps to label ‘wordmatch’. The word character is reset by loading the address of the input word into register $t1 and the word

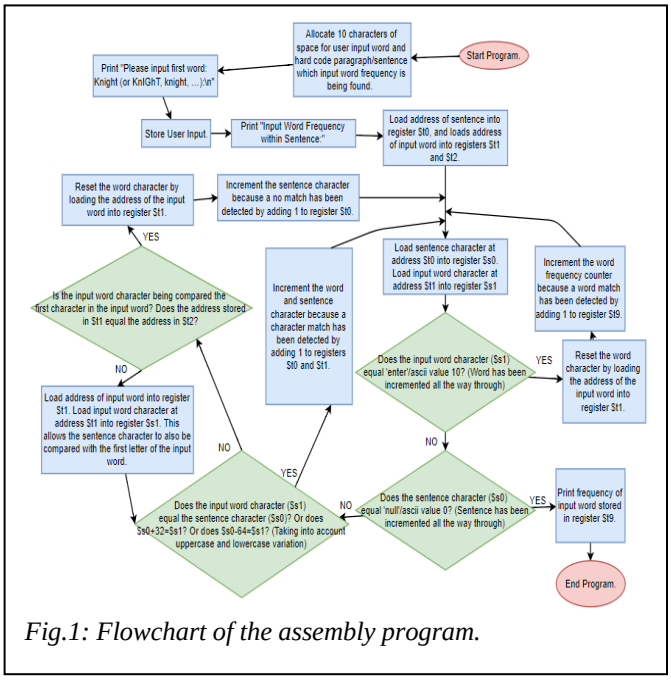

Fig.1: Flowchart of the assembly program.

frequency counter is incremented by adding 1 to register $t9. Then the program jumps back to the label ‘Loop’. If no, then the program moves on to another branch statement used to determine if the sentence character (stored in register $s0) equals 'null'/ascii value 0 which means that the sentence has been incremented all the way through. If yes, it jumps to label ‘end’ where the input word frequency stored in register $t9 is printed then the program ends. If the sentence character does not equal null, then the program moves on to another branch statement.

In order to determine if the input word character (stored in register $s1) is equal to the sentence character (stored in register $s0), the uppercase/lowercase variation needs to be takin into account. This can by done using three separate branch statements to determine if $s0=$s1 OR $$s0=$s1+32 OR $s0=$s1-64 keeping in mind that the ascii value for a uppercase and lowercase letter are separated by 32. If the answer is yes to any of those three statements, the program jumps to the label ‘charmatch’ because a character match has been detected. The word and sentence character are

then incremented by adding 1 to registers $t0 and $t1, then it jumps back to label 'Loop'. If the answer is no to all of the three branch statements, the program continues to a branch statement to determine if the input word character being compared (address stored in $t1) is equal to the to the first character of the input word (address stored in $t2). If yes, then it jumps to label 'nomatch' because the two characters are not a match. The word character is reset by loading the address of the input word into register $t1, the sentence character is incremented by adding 1 to register $t0, and the program jumps back to label 'Loop'.

If the input word character being compared (address stored in $t1) is not equal to the to the first character of the input word (address stored in $t2) then the sentence character also needs to be compared to the first character of the input word. For example, if the input word is "ABC" and the sentence is "ABABC". The program would find a character match for the first 'A' then the first 'B' within the sentence, however since the third letter within the sentence is not a 'C' it is not a word or character match. But, it still needs to be compared with the first character of the input word to determine if it is the start of a word match. Therefore, if the input word character being compared is not equal to the to the first character of the input word then the program jumps to label 'possiblynomatch'. The address of the user input word is loaded into register $t1, and the first character of the input word is loaded into register $s1 using the address stored in register $t1. Then three branch statements are used to determine if the input word character (stored in register $s1) is equal to the sentence character (stored in register $s0) which are if $s0=$s1 OR $$s0=$s1+32 OR $s0=$s1-64. If yes to any of the three branch statements, then a character match has been detected and jumps to label 'charmatch'. If no to all three branch statements, then it is not a character match and jumps to label 'nomatch'.

Test Program #1 was chosen to test if the program output matches the sample output given the user input word "KNIGHT" and the hardcoded sentence "UCF, its athletic program, and the university's alumni and sports fans are sometimes jointly referred to as the UCF Nation, and are represented by the mascot Knightro.The Knight was chosen as the university mascot in 1970 by student election. The Knights of Pegasus was a submission put forth by students, staff, and faculty, who wished to replace UCF's original mascot, the Citronaut, which was a mix between an orange and anastronaut. The Knights were also chosen over Vincent the Vulture, which was a popular unofficial mascot among students at the time. In 1994, Knightro debuted as the Knights official athletic mascot.". The output for Test Program #1 can be found in Figure 2 which matches the sample output perfectly.

Test Program #2 was chosen to test if the sentence character is also being compared to the first character of the input word, if the original input word character being compared is not equal to the to the first character of the input word. The user input word "ABC" was compared to the sentence "ABABC" which follows the example written earlier in this section. The program handled this situation flawlessly and the output for Test Program #2 can be found in Figure 3.

Test Program #3 was chosen to test how the program reacts to nothing. The user input word was purely the enter key/ascii value 10 while the hardcoded sentence was "". The code did not react well as it never stopped running. The output for Test Program #3 can be found in Figure 4.

```
Please input first word: Knight (or KnIGhT, knight, &):
KNIGHT
Input Word Frequency within Sentence: 6
-- program is finished running --
```

Fig.2: Assembly output for Test Program #1

```
Please input first word: Knight (or KnIGhT, knight, &):
ABC
Input Word Frequency within Sentence: 1
-- program is finished running --
```

Fig.3: Assembly output for Test Program #2

```
Please input first word: Knight (or KnIGhT, knight, &):

Input Word Frequency within Sentence:
```

Fig.4: Assembly output for Test Program #3

II. Full-Adder Circuit

There are many different ways to minimize the energy consumption of the full-adder (FA) circuit. Reference [1] tackles this issue by starting with the Conventional Mirror Adder (CMA) which has a total of 24 transistors and approximates the full adder circuit by reducing the number of transistors in the CMA [1]. The inputs to the circuit include A, B, and Carry-in which are to be added together to produce the outputs Carry-out and Sum [1]. However, the trend appears that the greater the number of transistors removed from the full adder circuit then the worse the approximation will be [1]. The inputs, accurate outputs, and 3 separate cases of approximation outputs can be found in Figure 4 which show this trend [1]. Reference [2] follows a very similar method to reference [1] in reducing the energy consumption of the full-adder circuit by approximation and reduction of transistors. The less number of transistors used in the full-adder approximation, the greater the power savings were compared to the conventional mirror adder [2]. Approximately 15 years before reference [1] was published, power savings were being achieved in full adder circuit designs by minimizing the number of transistors [4]. The static energy recovery full adder

(SERF) design was published in 1999 which focused on having low power and low transistor count [4]. Although it was not the fastest full-adder design of those compared in reference [4], the SERF design ended up being the most energy efficient [4].

TRUTH TABLE FOR CONVENTIONAL FULL ADDER AND APPROXIMATIONS 1, 2 AND 3 [13]

Inputs			Accurate Outputs		Approximate Outputs					
A	B	C_{in}	Sum	C_{out}	Sum_1	C_{out1}	Sum_2	C_{out2}	Sum_3	C_{out3}
0	0	0	0	0	1	0	0	0	0	0
0	0	1	1	0	1	0	1	0	0	0
0	1	0	1	0	0	1	0	0	1	0
0	1	1	0	1	0	1	1	0	1	0
1	0	0	1	0	1	0	0	1	0	1
1	0	1	0	1	0	1	0	1	0	1
1	1	0	0	1	0	1	0	1	1	1
1	1	1	1	1	0	1	1	1	1	1

Figure 4: Truth Table from Reference [1] where FA approximation 3 has less transistors than FA approximation 2 which has less transistors than FA approximation 1

Reference [3] focuses on reducing energy consumption with a design of a spin transfer torque magnetic full adder (STT-MFA) based off the design of spin transfer torque magnetic random-access memory (SST-MRAM) [3]. The pre-charge sense amplifier (PCSA) circuit is used in the SST-MFA design to allow "the amplification from analog data to digital with ultra-low power" and allow "the read disturbance induced by sensing operations can be significantly decreased" [3]. This design also allows efficient area minimization and reaches output values based off the resistances of transistors within the STT-MFA circuit [3]. Although the inputs for this design [3] are the same as references [1], [2], and [4] the outputs are different due to design differences. The inputs and outputs of this design can be found in Figure 5.

TRUTH TABLE AND RESISTANCE CONFIGURATION OF "CO"

A	B	C_i	Resistance Comparison	Co	Sub-branch AC_i	Sub-branch $\overline{AC_i}$
0	0	0	$R_L > R_R$	0	$2R_{OFF}$	$2R_{ON}$
0	0	1	$R_L > R_R$	0	$R_{OFF}+R_{ON}$	$R_{ON}+R_{OFF}$
0	1	0	$R_L > R_R$	0	$2R_{OFF}$	$2R_{ON}$
0	1	1	$R_L < R_R$	1	$R_{OFF}+R_{ON}$	$R_{ON}+R_{OFF}$
1	0	0	$R_L > R_R$	0	$R_{ON}+R_{OFF}$	$R_{OFF}+R_{ON}$
1	0	1	$R_L < R_R$	1	$2R_{ON}$	$2R_{OFF}$
1	1	0	$R_L < R_R$	1	$R_{ON}+R_{OFF}$	$R_{OFF}+R_{ON}$
1	1	1	$R_L < R_R$	1	$2R_{ON}$	$2R_{OFF}$

Figure 5: Truth Table from Reference [3] of SST-MFA

III. RESULTS AND DISCUSSION

Energy consumption is a very important factor that should be minimized as much as possible. When it comes to the assembly program above, we will be examining its total energy consumption as a factor of full-adder circuit design which is assumed to be present in every dynamic ALU instruction. The energy consumption for each ALU instruction for the designs found in references [1-3] can be found in Table I. The dynamic instruction counts for each instruction type contained in the assembly program can be found in Table II. Assuming the energy consumption for branch instructions is 3 pJ, jump instructions is 2 pJ, memory instructions is 100 pJ, and other instructions is 5 pJ, the total energy consumption of the assembly program based off of which full-adder design is used (from references [1-3]) can be found in Table III.

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

Design	Energy Consumption For Each ALU Instruction
[1]	5 fJ
CMA [2]	39 fJ
AMA [2]	12 fJ
[3]	1 fJ

Table II: Dynamic Instruction Count for the assembly program specified in Program Design by Instruction Type.

Instruction Type	Dynamic Instruction Count
ALU	3733
Jump	629
Branch	4285
Memory	1260
Other	6
Total	9913

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

Design	Total Energy Consumption
[1]	140.161665 nJ
CMA [2]	140.288587 nJ
AMA [2]	140.187796 nJ
[3]	140.146733 nJ

The full-adder circuit design with the least amount of total energy consumption is the STT-MFA circuit design from reference [3]. This design uses approximately 0.142 nJ less energy than the least energy efficient design which is the conventional mirror adder from reference [2].

IV. Conclusion

The full-adder circuit design used within an assembly program can have a significant impact on the total energy consumption. Here are the main points to take away from this paper:

- When trying to minimize the energy consumption for a specific assembly code, try to minimize the dynamic instruction count as much as possible. Make sure to prioritize minimizing the types of instruction count with the highest energy consumption per instruction.
- Accurate CMAs require more energy consumption than an approximation based off this design
- Even if a FA design is the most energy efficient, it does not mean that the design is the fastest or provides efficient area minimization
- Energy consumption, speed, area, and approximations are all factors that need to be considered when picking the best FA design for specific applications
- There are many ways to design a low energy consumption FA which has a big impact on the total energy consumption of an assembly code
- Between the designs mentioned in references [1-3], the Spin Transfer Torque-Magnetic Full Adder (SST-MFA) design from reference [3] provided the least total energy consumption for this assembly program which was 140.146733 nJ.

References

[1] A. A. Naseer, R. A. Ashraf, D. Dechev, and R. F. DeMara, "Designing energy-efficient approximate adders using parallel genetic algorithms," *SoutheastCon 2015*, Fort Lauderdale, FL, 2015, pp. 1-7.

[2] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: imprecise adders for low-power approximate computing," *In Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design (ISLPED '11)*, Piscataway, NJ, USA, 409-414.

[3] E. Deng, Y. Zhang, J. O. Klein, D. Ravelsona, C. Chappert and W. Zhao, "Low Power Magnetic Full-Adder Based on Spin Transfer Torque MRAM," in *IEEE Transactions on Magnetics*, vol. 49, no. 9, pp. 4982-4987, Sept. 2013.

[4] R. Shalem, E. John, and L. K. John, "A Novel Low Power Energy Recovery Full Adder Cell" , *Proc. Great Lakes Symp. VLSI*, pp. 380-383, Feb. 1999