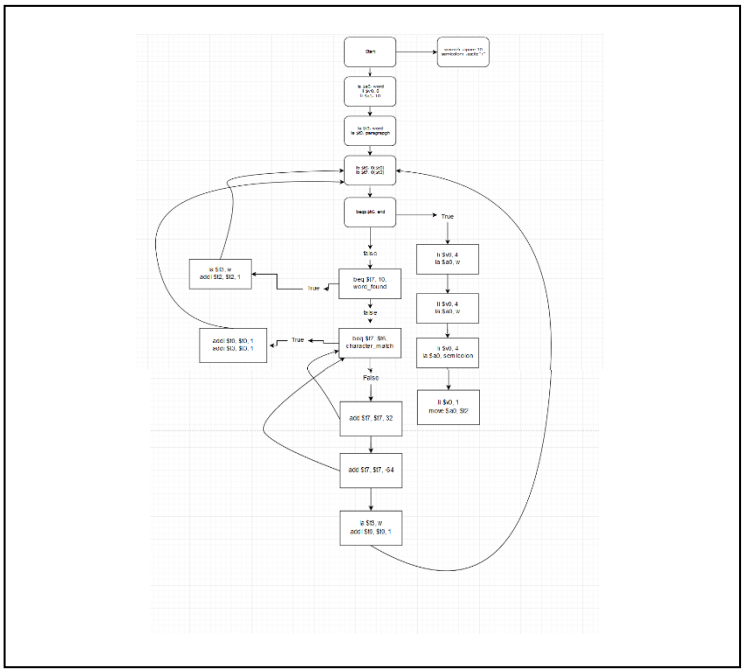
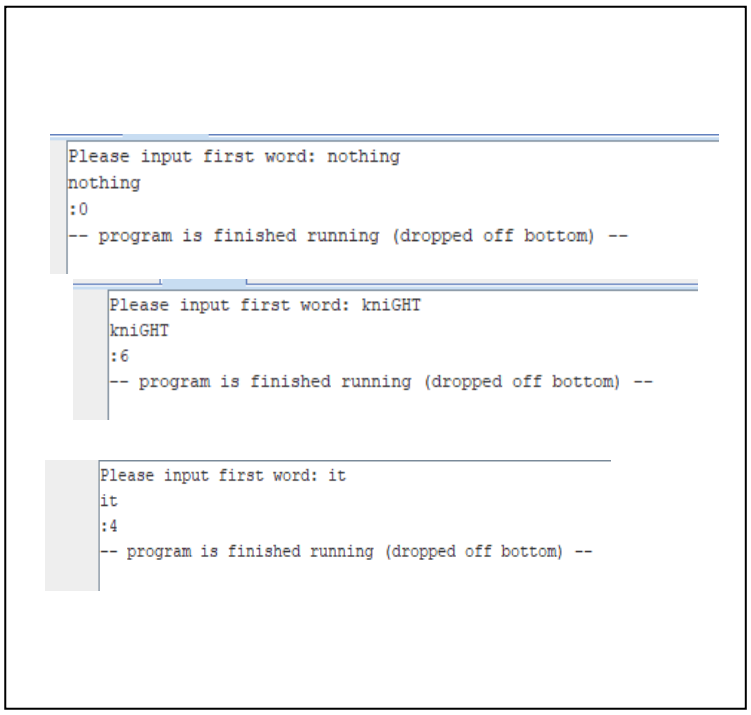
Analyzing the energy consumption of an algorithm using Conventional Mirror Adder approximations

Alonso Ninalaya
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

***Abstract*— Approximation and improving in the computing field determines that amount of energy that can be saved every time there is an algorithm to be processed. Although Algorithms do not vary significantly using different approximations, the energy consumption does, which is what we are looking for. This will bring different types of inaccuracy in the outputs but there will always be a better approximation to follow. Logic complexity reduction comes to be an alternative approach to get a more accurate output. This is demonstrated by proposing various approximations Full Adders cells with reduced complex city or less circuits and utilize them to design multi-but adders. For our project, we are going to use Conventional Mirror Adder(CMA) which consist of a total of 24 transistors. Then, these three approximations are being used to find the total energy consumption of our code; they are, AMA, AMA2 and AMA3. This algorithm is a program using MIPS in which the user can find how many times any word can be found inside a determined paragraph. Our output will include the word followed by how many times it was mentioned in the paragraph. The lowest total energy consumption is AMA 3 with 136.11 nJ.**

***Keywords*— genetic algorithms, approximate computing, adder, variable accuracy, error distance, power consumption, power reduction, Mirror adder.**





I. Project Design

The objective of this program is to find how many times a word is found within a pre-selected paragraph. This word is given by the user and it can be any random one. It can also be written using uppercase or lowercase in any order. Because of this, we will have to ignore whether the character is in lowercase or uppercase. Since the word is given by the user, our output is going to be based on what word is taken.

The program design is based on using different conditions. First, we are going to start by saving our paragraph and word in a string with enough space. After that, we proceed to load the address of each string inside each register so that we can compare them. After that, we need to compare each character of the paragraph and the word, so we need to load each byte to a different register for each of them. We are to load the first byte and create another loop where we increment the address of each register. Once it is done, we need proceed to compare them, however, since our word can be in in uppercase or lowercase, we have to convert each of them and compare. This can be done by adding 32 and compare, then subtract 64 and compare. This

method will help you to compare the uppercase or lowercase of the given word.

After comparing each character, we are going to need the 3 different cases, where a character is match, character does not match, and the word is found. In the first case, we need to increment the register where the addresses of our paragraph and our word are. Then go back the redo the loop. The second case is where we do have a character match, where we just increment the address of the paragraph and we reload the address of our word in the same register. Also, we need to go back to the loop. The third case is where we found the word, where we need to reload the address of the word, increment a counter, which will be printed at the end of the program, and redo the loop.

Finally, the output is going to depend on the word given by the user. Our program is going to print the word and the amount of times it was found in the program. We are going to test 3 cases. The first will be a word that is going to be in the paragraph, all lowercase. The second will be a word that is also in the paragraph, but with randomly lowercase and uppercase characters. The last one will consider on the word that is not inside the paragraph.

II. Full-Adder Circuit

A full-adder function can be described as given the three 1-bit inputs A, B, Cin. It is desired to calculate the two 1-but outputs. Sum and Cout, where:

Sum = (A ^ B) ^ Cin
Cout = A.B + Cin(A^B)
Here, we are introduced to the conventional mirror adder which Consists of 24 transistors having 3 different approximations.
These ones are meant to reduce the energy operation for applications which can tolerate some impression. These addresses utilize lesser number of transistors than original accurate design. The first one is obtained by reducing the number of transistors individually until the defined error constraints are breached. Then, the last modification is reverted to obtain the approximation. Any input combination of A, B, and Cin does not result in short or open circuits in the simplified schematic. Here, the Sum circuit is discarded completely, and a buffer stage is introduced after Cout to reduce the delay.
The second approximation contains 2 errors in Cout and 3 in Sum, we consider the inputs A and B are interchangeable, where Cout = A. Thus, a second approximation is proposed where we invert the input A to calculate Cout and Sum. They are similarly calculated to the simplified MA. In the first 2 approximations, Cout is calculated by using an inverter with C out as input.

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

The third approximation are accurate for 4 out of 8 outputs. 4 errors for Sum. The dependency of Sum on Cin is reduced. Leaving with two choices. Sum = A, and Sum =B. Another one, were Cout = A. Thus, we have one where we find that both Sum and Cout match with accurate outputs in only 2 out of the 8 outputs. If we want t minimize the error, then we for the second choice where we ensure that Sum makes Cout correct.

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

Design	Total Energy Consumption
[1]	137.39nJ
CMA [2]	137.51nJ
AMA [2]	137.41nJ
[3]	136.11nJ

III. Results and Discussion

In this section, I am using these energy values to calculate the total energy consumption of my code.

1) *ALU = Refer to Table I*
2) *Branch = 3 pJ*
3) *Jump = 2 pJ*
4) *Memory = 100 pJ*
5) *Other = 5 pJ*

IV. Conclusion

This paper will indicate how to calculate the power energy consumption utilizing the three different approximations of a conventional mirror adder. We have proposed several approximations FA cells that can be effectively utilized to build multi-bit arithmetic units and trade off power, are and quality for error. Our approach aims to simply the complexity of a conventional mirror adder cell by reducing the number of transistors and to give a margin of error to each output of the adder. This contributes to a reduction in power dissipation.

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.