Minimizing Floating-Point Power Dissipation Via Bit-Width Reduction

Ying Fai Tong, Rob A. Rutenbar, and David F. Nagle
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213
{yftong,bassoon}@ece.cmu.edu

Abstract

For many low-power systems, the power cost of floating-point hardware has been prohibitively expensive. This paper explores ways of reducing floating-point power consumption by minimizing the bit-width representation of floating-point data. Analysis of several floating point programs that utilize low-resolution sensory data shows that the programs suffer almost no loss of accuracy even with a significant reduction in bit-width. This floating point bit-width reduction can deliver a significant power saving through the use of a variable bit-width floating point unit.

1 Introduction

Floating point numbers provide a wide, dynamic range of representable real numbers, freeing programmers from the manual scaling code necessary to support fixed-point operations. Floating-point (FP) hardware is also very power hungry. For example, FP multipliers are some of the most expensive components in a processor's power budget. This has limited the use of FP in embedded systems, with many low-power processors not including any floating point hardware.

For an increasing number of embedded applications such as voice recognition, vision/image processing, and other signal-processing applications, FP's simplified programming model (vs. fixed-point systems) and large dynamic range makes FP hardware a useful feature for many types of embedded systems. Further, many applications achieve a high-degree of accuracy with fairly low-resolution sensory data. Leveraging these characteristics by allowing software to use the minimal number of mantissa and exponent bits, standard floating-point hardware can be modified to significantly reduce its power consumption while maintaining a program's overall accuracy.

For example, Figure 1 graphs the accuracy of CMU's Sphinx Speech Recognition System [Sphinx98] vs. the number of mantissa bits used in floating point computation. The left most point, 23 bits of mantissa, is the standard for a 32-bit IEEE FP unit. With 23 bits, the recognition accuracy is over 90%; but even with just 5 mantissa bits (labeled A), Sphinx still maintains over 90% word recognition accuracy. For Sphinx, there is almost no difference between a 23-bit mantissa and a 5-bit mantissa. In terms of power, however, a FP multiplier that uses only 5 mantissa bits consumes significantly less power than a 23-bit mantissa FP multiplier. This property, that programs can maintain accuracy while utilizing only a few bits for FP number representation, creates a significant opportunity for enabling low-power FP units.

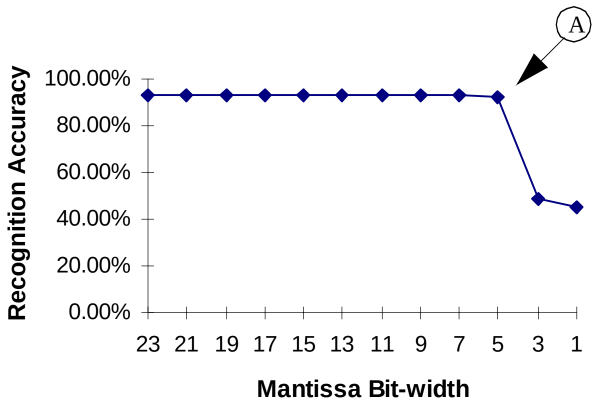

Figure 1: Accuracy of Sphinx Speech Recognition vs. Mantissa Bit-width

The goal of this work is to understand the floating-point bit-width requirements of several common floating-point applications and quantify the amount of power saved by using variable bit-width floating-point units. Section 2 begins our discussion by examining different aspects of the IEEE floating point standard that could lead to additional power savings. Section 3 presents the analysis of several floating point programs that require less bits than specified in the IEEE standard. In section 4, we describe the use of a digit-serial multiplier to design variable bit-width hardware and discuss the possible power savings. Finally, Section 6 outlines our conclusions and future work.

This work was supported by the Defense Advanced Research Projects Agency under Order No. A564 and the National Science Foundation under Grant No. MIP90408457.

Presented at the Power-Driven Microarchitecture Workshop in conjunction with the 25th International Symposium on Computer Architecture, June 27 - July 1, 1998, Barcelona, Spain.

2 Background

2.1 IEEE 754 Floating Point Standard

One of the main concerns of the IEEE 754 Floating Point Standard is the accuracy of arithmetic operations. IEEE-754 specifies that any single precision floating point number be represented using 1 sign bit, 8 bits of exponents and 23 bits of mantissa. With double precision, the bit-width requirements of exponents and mantissa go up to 11 bits and 53 bits respectively.

In addition to specifying the bit-width requirement for floating point numbers, IEEE-754 incorporates several additional features, including delicate rounding modes and support for gradual underflow to preserve the maximal accuracy of pro-

Dimension	Description
Reduction in mantissa bit-width	Reduce the number of mantissa bits at the expense of precision.
Reduction in exponent bit-width	Reduce the number of exponent bits at the expense of a smaller dynamic range.
Change of the implied radix	Increase the implied radix from 2 to 4 (or 16). This provides greater dynamic range but lower density of floating point numbers, potentially leading to power savings since fewer normalizing shifts are necessary.
Simplification of rounding modes	Full support of all the rounding modes is very expensive in terms of power. Some programs may achieve an acceptable accuracy with a modified low power rounding algorithm.

Table 1 : Design Dimensions for Floating Point Representation

grams. Nevertheless, the implementation of an IEEE compliant floating point unit is not always easy. In addition to the design complexity and the large area it occupies, a floating point unit is also a major consumer of power in microprocessors. Many embedded microprocessors such as the StrongARM [Dobberpuhl96] and MCore [MPR97] do not include a floating point unit due to its heavy implementation cost.

2.2 Accuracy Requirements and Workloads

For floating point applications that rely on sensory inputs, power savings can be obtained by modifying the floating point hardware's mantissa and exponent widths while maintaining sufficient accuracy for overall program execution. There are four conceivable dimensions that we can explore (see Table 1). Each of these dimensions allow us to make trade-off between program accuracy and the power consumption of the floating-point unit.

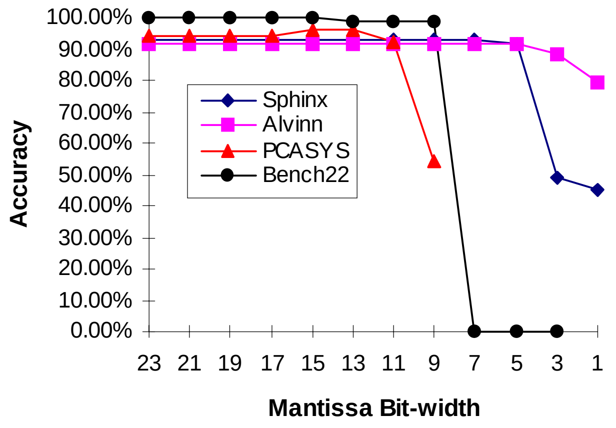

Figure 2: Program Accuracy across Various Mantissa Bit-widths

3 Experiments and Results

3.1 Methodology

To validate the usefulness and accuracy of reducing FP bit-widths, we analyzed four single-precision floating point programs (see Table 2). To determine the impact of different mantissa and exponent bit-widths, we emulated different bit-width FP units in software by replacing each floating-point operation with a corresponding function call to our floating-point software emulation package that initially implements the IEEE-754 standard. Careful modifications to the floating-point emulation package allowed us to simulate different mantissa and exponent bit-widths. For each bit-width, the emulation package was modified to use a smaller number of bits. Then, each program was run using the modified floating-point package and the results are compared to determine application accuracy.

3.2 Results

Figure 2 graphs the accuracy for each of the four programs across a range of mantissa bit-widths. None of the workloads displays a noticeable degradation in accuracy when the mantissa bit-width is reduced from 23 bits to 11 bits. For ALVINN and Sphinx III the results are even more promising; the accuracy does not change significantly with mantissa bit-width of 5 or more bits.

Workload	Description	Accuracy Measurement
Sphinx III	CMU's speech recognition program based on fully continuous hidden Markov models. The input set is taken from the DARPA evaluation test set which consists of spoken sentences from the Wall Street Journal. [Hwang94]	Accuracy is estimated by dividing the number of words recognized correctly over the total number of words in the input set.
ALVINN	Taken from SPECfp92. A neural network trainer using backpropagation. Designed to take as input sensory data from a video camera and a laser range finder and guide a vehicle on the road.	The input set consists of 50 road scenes and the accuracy is measured as the number of correct travel direction made by the network.
PCASYS	A pattern-level finger print classification program developed at NIST. The program classifies images of fingerprints into six pattern-level classes using a probabilistic neural network.	The input set consists of 50 different finger print images and the classification result is measured as percentage error in putting the image in the wrong class. The accuracy of the recognition is simply (1 - percentage error).
Bench22	An image processing benchmark which warps a random image, and then compares the warped image with the original one.	Percentage deviation from the original outputs are used as a measure of accuracy.

Table 2 : Description of Workloads

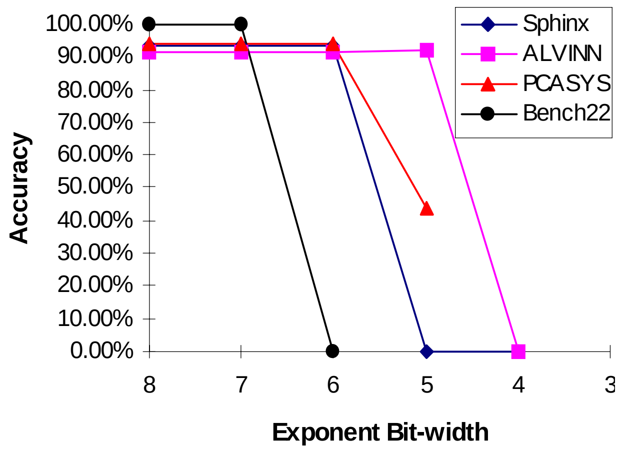

Figure 3: Program Accuracy across Various Exponent Bit-widths

Figure 2 and Figure 3 show that we can reduce both the mantissa and exponent bit-width without affecting the accuracy of the programs. This effect is especially prominent in the mantissa. This reduction of bit-width can be turned into a reduction in power dissipation with the use of appropriate arithmetic circuits

Figure 3 shows that each program's accuracy has a similar trend when the exponent bit-width is varied. With 7 or more exponent bits, the error rates remain quite stable. Once the exponent bit-width drops below 6, the error rates increase dramatically and in some cases the programs could not finish properly.

Many programs dealing with human interfaces process sensory data with intrinsically low resolutions. The arithmetic operations on these data may generate intermediate results that require more dynamic range, but not vastly more precision. This is different from many scientific programs such as wind tunnel simulation or weather prediction, which not only require a huge amount of precision and dynamic range but also delicate rounding modes to preserve the accuracy of the results.

For programs that do not need the dynamic range nor the precision of floating point arithmetic, the use of fixed-point arithmetic might well be a better choice in terms of chip space, operation latency, and power consumption. But for the programs we have analyzed, three of them require 6 bits or more of the exponents to preserve a reasonable degree of accuracy, which means they need more than the 32 bits of precision that fixed point arithmetic can offer. Simply using fixed point representation without additional scaling will not resolve the problem.

It should be noted that these complex applications were aggressively tuned by various software designers to achieve good performance using full IEEE representation. However, Figure 2 and Figure 3 suggest that significantly smaller bit-width FP units could be used by these applications without compromising the necessary accuracy. For instance, certain floating point constants in the Sphinx III code require more than 10 bits of mantissa to represent, but we modified those numbers so they can be represented using fewer bits during our experiment and yet this have little impact on the overall recognition accuracy. We believe that if the numerical behavior of these applications are adjusted to a smaller bit-width unit, we could get even better performance.

Multiplier	Area	Cycle Time	Latency/op
Wallace (24x24)	.777square mm	40ns	40ns
Digit-Serial (24x8)	.804 square mm	15ns	15ns

Table 3 : Timing and Area of the Two Multipliers

To perform 16 bits multiplication using the digit-serial multiplier, 2 cycles are needed which increases the total delay/op to 30ns. Similarly, 24 bits multiplication takes 3 cycles(45ns).

4 Power Savings by Exploiting Variable Bit-width Requirement

4.1 Multiplication with a Digit-Serial Multiplier

Since different floating point programs have different requirements on both the mantissa and exponent bit-width, we propose the use of a variable bit-width floating point unit[1] to reduce power consumption. To create hardware capable of variable bit-width multiplications (up to 24x24 bit), we used a 24x8 bit digit-serial architecture similar to the one described in Hartley and Parhi [Hartley95]. The 24x8 bit architecture allows us to perform 8, 16, and 24-bit multiplication by passing the data once, twice, or three times though the serial multiplier. A finite state machine is used to control the number of iterations through the CSA array.

To perform accurate power and timing measurements, the multiplier was described in Verilog and then taken to layout using our ASIC design flow (for a standard 0.5u process). Synopsys' Design Compiler was used to synthesize the multiplier's control logic. Next, the entire structural model was fed into Cascade Design Automation's physical design tool Epoch. A description of digit-serial arithmetic and the block diagram of the digit-serial multiplier can be found in the appendix.

We compare our variable bit-width multiplier with a baseline fixed-width 24x24 bit Wallace Tree multiplier. The layout of this Wallace Tree multiplier was generated by Epoch's cell generator in the same 0.5u process as used in the design of the digit-serial multiplier. The two multipliers are described in Table 3. Cycle time is estimated using Epoch's static timing analysis tool, Tactic, and is rounded to the nearest 5ns interval for the convenience of power simulation.

4.2 Power Analysis

For each design, a SPICE netlist was generated from layout and used to estimate power consumption with Avanti's Star-Sim. Determining the complete power dissipated in a multiplier requires the sensitization of all possible combinations of inputs, which means we need to have 2^{2N} input combinations where N is the number of inputs. Fortunately, it is possible to obtain a mean estimation of the power consumption using statistical techniques [Burch92]. In our approach, we ran 50 batches of vectors with each batch containing 30 vectors to insure a 95% confidence intervals. The energy dissipation is computed using the cycle time in Table 3. The test vectors are taken from some of the actual multiplication operands in Sphinx III.

Figure 4 graphs the energy/operation and latency/operation for the digit-serial multiplier. Both the energy/operation and latency/operation decrease linearly with the operand bit-width. This is different from [Callaway97], where a multiplier's power

1. Our current research focuses particularly on the floating point multiplier, since multipliers are usually the major consumer of power [Tiwari94].

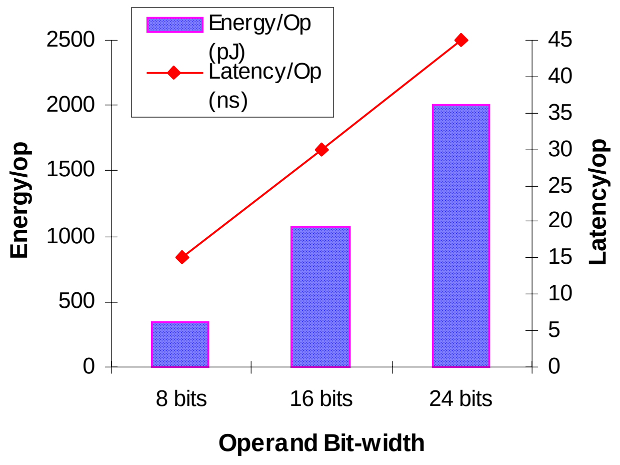

Figure 4: Performance of the Digit-Serial Multiplier

Both energy/op and latency/op of the digit-serial multiplier increase linearly with the operand bit-width. Digit-serial architecture allows us to perform variable bit-width arithmetic and save power when the bit-width requirement is less than that specified in the IEEE standard.

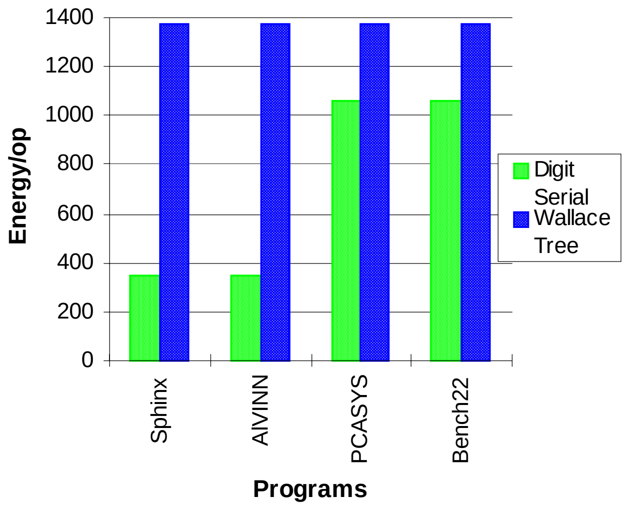

Figure 5: Power Reduction using Digit-Serial Multiplier

A reduction of 70% in energy/op is attainable for Sphinx and ALVINN with the use of digit-serial multiplier. Both Sphinx and ALVINN need only 5 bits of mantissa to be 90% accurate and thus an 8 bit operand bit-width is used for the digit-serial multiplier. PCASYS requires 11 bits of mantissa while Bench22 requires 9 bits and thus a 16 bit operand bit-width is used which results in a 20% energy/op reduction

consumption decreases exponentially with the operand bit-width. The difference between the two results is due to the fixed structure (the 24x8 bit CSA array) of the digit-serial multiplier and the control circuitry needed to do iterative carry and sum reduction. This additional power dissipation is the penalty we pay for the flexibility of doing variable bit-width multiplication.

Figure 5 shows the potential power reduction for our three programs if we use the digit-serial multiplier as the mantissa multiplier. For 8-bit multiplication, the digit-serial multiplier consumes less than 1/3 of the power than the Wallace Tree multiplier (in the case of Sphinx and ALVINN). When 9 to 16 bits of the mantissa are required (in the case of PCASYS and Bench22), the digit-serial multiplier still consumes 20% less power than the Wallace Tree multiplier. The digit-serial multiplier does consume 40% more power when performing 24-bit multiplication due to the power consumption of the overhead circuitry.

Table 3 shows another benefit which is improved speed. When performing 8-bit or 16-bit multiplications, the operation's delay can be greatly reduced if the critical path of the circuit lies in the multiplier. This increases the throughput in addition to the energy saving.

4.3 Summary of Results

These power comparison results show the potential power savings achievable by using variable bit-width arithmetic units. It should be noted that the digit-serial multiplier is designed using an ASIC approach and is not as heavily optimized physically; the Wallace Tree multiplier was optimized for the process. This explains why the 24x8 bit digit-serial multiplier is actually slightly larger than the Wallace Tree multiplier. We believe that with a more careful implementation, both the power and area of the digit-serial multiplier can be reduced. In addition, [Chang97] presents several low power digit-serial architecture that can further reduce the power consumption of the digit-serial multiplier.

For software designers who know a program's precision requirements, code annotation could be used to allow the underlying arithmetic circuits re-configure themselves for variable bit-width operations. As Figure 4 and Figure 5 show, as long as the bit-width requirement is less than 16 bits, the 24x8 bit digit-serial multiplier consumes less energy than the Wallace Tree multiplier. Even for programs which require a large bit-width to maintain precision, there may be sections within the program that require a smaller bit-width requirement and thus can benefit from a variable bit-width unit.

5 Previous Work

The idea of reducing bit-width to save power has been employed in other areas of low power research. In [He97], it is shown that an average of more than 4 bits of pixel resolution can be dropped during motion estimation to obtain a power reduction of 70%. To reduce energy consumption of the I/O pins, [Musoll97] proposes sending only those address bits that have changed. For integer applications that do not need 32 bits of precision, the Intel MMX instructions allow arithmetic operations on multiple data simultaneously. Overall, the basic idea of bit-width reduction is to avoid waste.

6 Conclusions and Future Work

It is clear that floating point programs that use human sensory inputs may not need the entire 23 bits of mantissa and 8 bits of exponents as specified in the IEEE standard. Our software analysis shows no loss of accuracy across each of our programs when the mantissa bit-width is reduced to less than 50% of the original value. This large reduction in mantissa bit-width enables significant power reduction without sacrificing program accuracy. Further, the digit-serial multiplier results show that it is possible to obtain a substantial power saving by using a variable bit-width arithmetic unit.

Currently, we are looking at various digit-serial architectures and other means of exploiting the variable bit-width requirements of programs. Our future work will be directed towards investigating other characteristics of floating point programs

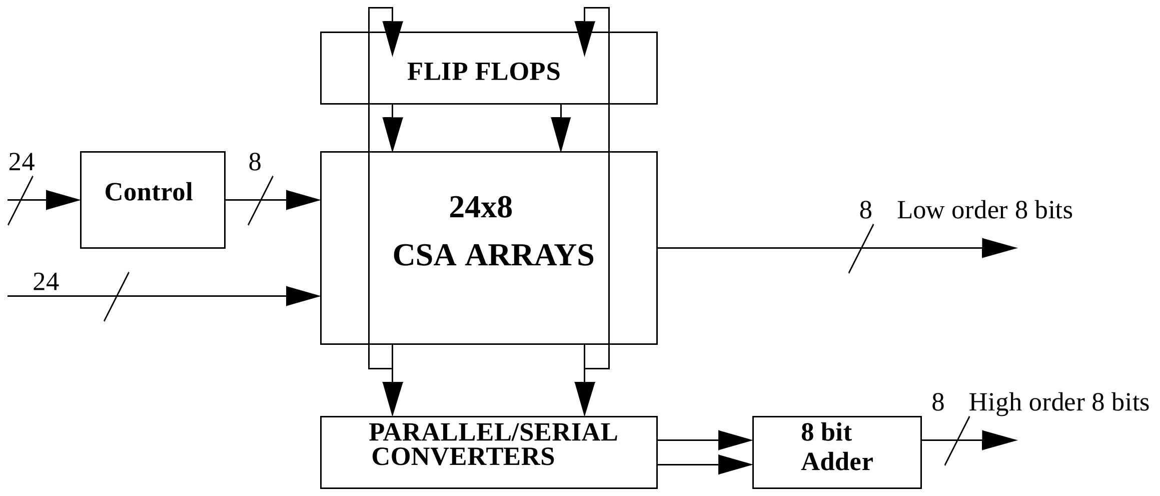

Figure 6: Block Diagram of a 24x8 Digit-Serial Multiplier

that may provide additional power savings. This includes using a simplified rounding algorithm and changing the implied radix.

7 Appendix

Arithmetic operations can be performed in different styles such as bit-serial, bit-parallel, and digit-serial. Bit-serial architecture processes data one bit at a time, saving hardware at the expense of processing speed. Bit-parallel circuits process all bits of the data operands at the same time with more hardware. Digit-serial arithmetic falls in between these two extremes by processing a fixed number of bits at one time. Figure 6 shows the block diagram of the digit-serial multiplier used in our experiment. For applications that require a moderate amount of throughput while having serious constraints on design space, digit-serial systems have become a viable alternative for many designers. Most of the digit-serial systems are used because of space limitation. Even though there is research on low-power digit-serial multipliers [Chang97], it has focused on comparing power consumption among different digit-serial architecture.

8 References

[Burch92] Burch R., Najm, F., Yang, P., Trick, T. *McPOWER: A Monte Carlo Approach to Power Estimation*, IEEE/ACM International Conference. on CAD, 1992.

[Callaway97] Callaway, Thomas K., Swartzlander, Earl E. *Power-Delay Characteristics of CMOS Multipliers*, IEEE 13th Symposium on Computer Arithmetic, 1997.

[Chang97] Chang, Y.N., Satyanarayana, Janardhan H., Parhi, Keshab K. *Design and Implementation of Low-Power Digit-Serial Multipliers*, IEEE International Conference on Computer Design, 1997.

[Dobberpuhl96] Dobberpuhl Dan. *The Design of a High Performance Low Power Microprocessor.* International Symposium on Low Power Electronics and Design, 1996.

[Hartley95] Hartley, Richard I., Parhi, Keshab K. *Digit-Serial Computation.* Norwell, Kluwer Academic Publishers, 1995.

[He97] He, Z.L., Chan, K.K., Tsui, C.Y., Liou, M. L., *Low Power Motion Estimation Design Using Adaptive Pixel Truncation*, International Symposium on Low Power Electronics and Design, 1997.

[Hwang94] Hwang, M., Rosenfeld, R., Theyer, E., Mosur, R., Chase, L., Weide, R., Huang, X., Alleva, F. *Improving Speech recognition performance via phone-dependent VQ codebooks and adaptive language models in SPHINX-II.* International Conference on Acoustics, Speech and Signal Processing, 1994.

[MPR97] *MCore Shrinks Code, Power Budgets*, Microprocessor Report11 (14), 27 October 1997.

[Musoll97] Musoll, E., Lang, T., Cortadella, J. *Exploiting the locality of memory references to reduce the address bus energy.* International Symposium on Low Power Electronics 1997.

[Sphinx98] SPHINX Group, Carnegie Mellon University, Pittsburgh, PA. *http://www.speech.cs.cmu.edu.speech/sphinx.html*

[Tiwari94] Tiwari, V., Malik, S. Wolfe, A. *Power Analysis of Embedded Software: a first step towards software power minimization*, IEEE Transactions on VLSI systems, vol.2, pp. 437-445, December 1994.