Initial Results on Fuzzy Floating Point Computation for Multimedia Processors

Carlos Álvarez, Jesús Corbal, Esther Salamí, Mateo Valero

Abstract— **During the recent years, the market of mid/low-end portable systems such as PDAs or mobile digital phones have experimented a revolution in both selling volume and features as handheld devices incorporate *Multimedia* applications. This fact brings to an increase in the computational demands of the devices, while still having the limitation of power (and energy) consumption.**

Instruction memoization is a promising technique to help alleviate the problem of power consumption of expensive functional units such as the floating-point one. Unfortunately, this technique could be energy-inefficient for low-end systems due to the additional power consumption of the relatively big tables required.

In this paper we present a novel way of understanding multimedia floating point operations based on the fuzzy computation paradigm: losses in the computation precision may exchange performance for negligible errors in the output. Exploiting the implicit characteristics of media FP computation, we propose a new technique called fuzzy memoization. Fuzzy memoization expands the capabilities of classic memoization by attaching entries with similar inputs to the same output. We present a case of study for a SH4-like processor and report good performance and power-delay improvements with feasible hardware requirements.

Keywords— **Fuzzy Computation, Multimedia, Floating Point, Memoization.**

I. Introduction

ARCHITECTURAL techniques targeted at increasing performance for low-end systems are nowadays becoming an important field of research. However, the potential of the adoption of the same techniques introduced in general-purpose processors to increase performance in DSP systems is still a matter of controversy. Most authors identify speculation and dynamic execution as unsuitable for portable systems due to the power and energy inefficiencies of such techniques. Therefore, many designers of embedded processors [1] [2] [3] have opted for other techniques that do not jeopardize the power and energy consumption of the system, as for instance, the inclusion of sub-word level SIMD instructions.

This paper deals with the problem of floating-point computation for multimedia processors targeted at 2D imaging and 3D applications such as the Hitachi SH4 [4]. Trends in multimedia applications development indicate that media codes are migrating from fixed-point to floating-point computation to adapt to more complex protocol and media data types. Floating point units are expensive in terms of both area and power, and their high latencies are usually difficult to hide using in-order processors [5].

A very promising energy-efficient technique to improve performance for floating-point units is instruction memoization [6]. It allows to skip the execution of long latency instructions, such as multiplication and division, by trying to reuse previously invoked instances of the same operation with the same source operands via a look-up table (LUT). The major drawback of the technique is that large tables are usually required for leveraging significant reuse rates and so the energy consumption of such tables may offset potential savings.

In this paper we present a novel way of applying the fuzzy computation paradigm to floating-point instruction memoization. With fuzzy computation more effective reuse can be obtained by exploiting the fact that imprecision in calculation can exchange redundancy for negligible errors in the output of multimedia applications [7]. We present an initial evaluation of the performance and power benefits of this fuzzy memoization technique in a SH4-like processor. Initial results show a great potential for high energy and power savings. Given the properties of the fuzzy computation, we discuss the trade-offs between performance and quality of the media outputs.

II. Rationale for fuzzy floating-point memoization

A. Tolerance in media applications

Our focus is on improving the hit rate of memoization tables by exploiting an intrinsic characteristic of multimedia algorithms: *tolerance*. By tolerance we understand the robustness of media data types (such as image pixels or audio samples) to losses in the precision of computation. Precision of operations can be traded for a higher value locality in floating point instructions.

In sharp contrast with any other kind of applications, multimedia programs exhibit *tolerance* to losses in the accuracy of their outputs. Any given output of a 3D, video, image or audio application may present differences that are not visually (or audibly) perceptible. In other words, output "errors" may be filtered by the human senses in such a way that no difference in the subjective quality of the result is observed.

Fuzzy computation is a novel way of performing certain computations that takes advantage of this precise property. Precision in computation is exchanged for time and power savings. If we look closely at typical compression algorithms such as JPEG, MPEG-2 or MP3, we observe

C. Álvarez, J. Corbal, E. Salamí and M. Valero are with the Computer Architecture Department, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain. E-mail: {calvarez,jcorbal,esalami,mateo}@ac.upc.es

Manuscript submitted: 21 Dec. 2001. Manuscript accepted: 23 Jan. 2002. Final manuscript received: 30 Jan. 2002

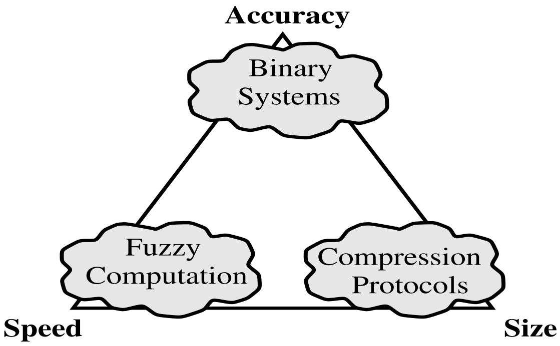

Fig. 1. Basis for the Fuzzy Computation Paradigm.

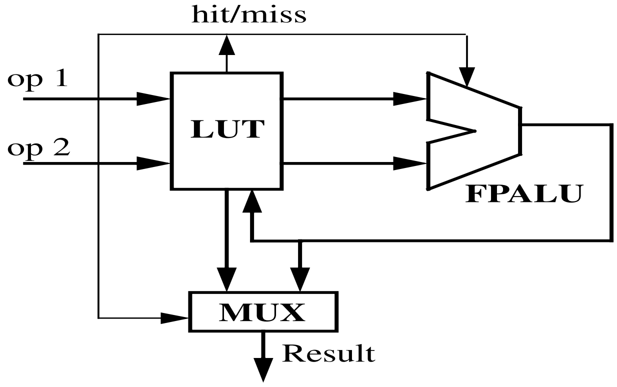

Fig. 2. Hardware configuration of a sequential LUT.

that these applications, in fact, exchange quality for output compression ratios. Fuzzy computation extends this trade-off with a third dimension: speed (as figure 1 illustrates).

Two reasons explain why fuzzy computation works:

(1) We have high redundancy in the data representation. Floating point data represents a much broader range of numbers than what is really required for the capacity of human senses. Fuzzy computation takes advantage of the fact that most of the error introduced is overridden by the final conversion of data to bounded integer representations.

(2) In media processing, the final destination are the human senses, and human senses have not unlimited capabilities. For example, the human eye can distinguish millions of different colors, but only some thousands at the same time. Moreover, the information received by human senses is processed by the brain, which has a natural tendency to adjust the perception. For instance, we detect very well edges in the images but detect poorly different color graduations in a flat surface, which is a property typically exploited by fuzzy computation.

B. Why floating point fuzzy memoization?

The floating-point units are not a very critical factor of performance and power consumption for aggressive processors. Floating-point latency is hidden thanks to out-of-order execution and the energy consumption is not a very significant part of the whole processor. Nevertheless, in low-end systems, execution is typically in-order, and therefore, the inclusion of floating-point units has a time and power consumption impact that is worth being addressed with special techniques such as memoization.

In this paper we want to apply the concept of fuzzy computation to floating-point instruction memoization. The Hitachi SH4 [4] is a 2-way RISC processor specially targeted at home-videogame systems and hand-held devices. The floating-point unit is the most critical part of the design in terms of power and area. Our objective is to evaluate the effect of tolerance over execution time, power and output quality when applied to a basic memoization unit targeted at floating-point multiplication and division.

III. Fuzzy memoization implementation

In classical memoization, two instances of the same operation are only reused when the two pairs of input operands are identical. Applying the concept of tolerance we can consider that two instances of the same operation can be reused not only when their pairs of input operands are equal, but also when they are close enough. As an example, if two multiplications have as input values 5.00×5.01 and 5.00×5.00 the result of the two will be 25.05, that is the result of the first multiplication.

This implementation of fuzzy computation has the main advantage of being very simple because tolerance is performed by just cutting off the N less significant bits of every operand before being used to access the table. In addition, the table is smaller than in classical memoization because it does not need to keep the $2 * N$ bits dropped. We use this number of bits dropped (N) to define the level of tolerance. Look up table works as in classical memoization, so if there is a hit in the table, the result (64 bits) is forwarded. If there is a miss, the pair of input operands is passed to the floating-point unit to do the complete operation and update the reuse table.

As we are focusing on the low power domain, the table has been implemented sequentially with the floating-point unit. Therefore, a miss in the table will increase the operation latency by one, but a hit will make the FPALU to not consume any power (figure 2). As table size is critical, only the two most demanding operations (multiplication and division) were memoized in a single shared table. In addition trivial operations [8] were treated apart so they do not pollute the reuse table.

A. Downsides and limits of fuzzy computing for FP

The main drawback of fuzzy computation is, as it has been already said, the introduction of an error in the output data. Although there are a lot of computation environments that cannot afford such kind of losses, the special characteristics of media applications make this error not only affordable but negligible. Media outputs are targeted to human senses, so their quality is subjective and cannot be easily measured. Great errors that spread out over all the signal (like an increment in the luminosity in entire

Original Image ($N = 0$ bits) Good Quality ($N = 46$ bits) Regular Quality ($N = 47$ bits) Bad Quality ($N = 48$ bits)

Fig. 3. Different outputs of the algorithm when different tolerance levels are used.

TABLE I

Power Breakdowns of the model.

Hardware Structure	Percentage of total Power
Caches	28 %
Registers	1 %
Instruction Queues	1 %
Integer Execution Unit	9 %
Floating Point Unit	30 %
Total Clock Power	31%

TABLE II

Configuration of model processor.

Processor Core	
Issue	In order
Physical Registers	32
Fetch width per cycle	2
Decode width per cycle	2
Issue width per cycle	2
Commit width per cycle	2
FP units	1
Integer units	1
Branch prediction	Not Taken
Floating Point Latencies (issue latencies)	
Addition	4 (3)
Multiplication	8 (6)
Division	24 (24)
Memory Hierarchy	
L1 Dcache Size	16 K
L1 Dcache Assoc.	2-way
L1 Icache Size	16 K
L1 Icache Assoc.	2-way
DTLB Size (full assoc)	32
ITLB Size (full assoc)	32
L2 Cache	none
Process Specifications	
Feature Size	.25um
Vdd	1.8 V
MHz	200

image) are hardly perceived, however, concentrated little errors (like a red dot in a white wall) are easily perceived.

This behavior makes it difficult to set an objective error measure. The most common measure used in the signal processing environment is the Signal Noise Ratio. The SNR is defined as:

$$SNR = 20\frac{\log P_S}{\log P_N} \qquad (1)$$

where

$$P_S = \sum_{1}^{row}\sum_{1}^{col} x^2 \qquad (2)$$

$$P_N = \sum_{1}^{row}\sum_{1}^{col} (x-y)^2 \qquad (3)$$

and x is a pixel of the original image and y is the same pixel of the image with tolerance. The SNR is used to make an objective measurement of the quality of the output data.

B. Implementation Details

To measure the effectiveness of our method we have hacked the SimpleScalar [9] and Wattch [10] simulators to incorporate the Fuzzy LUT at the IEEE double precision multiplication unit. Our model was parameterized so that different tolerance levels (different values of N) can be used. The measures has been applied to the Epic compression algorithm [11].

The Simplescalar and Wattch tool sets have been configured to model a SH4 like processor. The Wattch power results were scaled down to emulate the ones of a SH4 .25 micron technology at 200 MHz and 1.8 Voltage supply. The power consumed by the LUTs has been measured using Cacti models and taking into account LUT delayed update in a miss due to operation latencies. Table I shows the power breakdowns of the SH4 like processor modeled and table II shows its hardware characteristics. The total power obtained for the model is a little bit more than 1.3 W at full power which is something more than the power declared for an SH4. Nevertheless we are only interested in percentage of reduction, so the total power is not significant.

C. Final Results

Figure 4 presents the total power consumption and overall execution time savings obtained with a 10 KB LUT table for different tolerance levels. For those cases in which some error is introduced, the SNR of the output image is also included. Two different areas can be distinguished. In the first one (up to 38 bits of tolerance), the tolerance introduced has nearly no effect in the output image. This area is where fuzzy computing takes benefit of the redun-

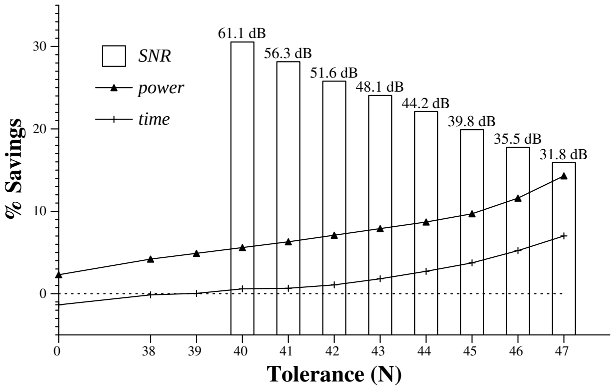

Fig. 4. Total power & overall execution time savings with 10 KB LUT.

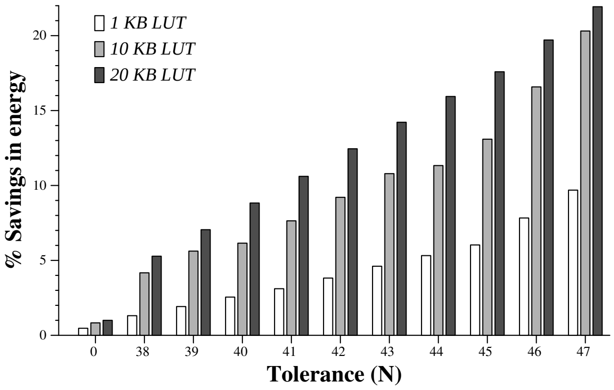

Fig. 5. Energy savings of different size tables.

dancy in the representation of the image. If we take out the redundancy we can make the LUT table thinner because we do not need to store the redundant bits; however the hit rate does not increase because bits taken off are not significant, and so different input values remain different. In the second area is where we are in fact tolerating, as the same result is assigned to different but close enough input values. In this case, both power and time are saved at the cost of introducing some error, but as figure 3 shows, errors are negligible until tolerance levels of 46 bits. Furthermore, when using low quality output devices (such as handheld screens), 47 or even 48 tolerated bits could be acceptable.

On the other hand, it can be seen that classical memoization ($N = 0$) only gets little savings in power but spends more time than the original configuration. This behavior is due to the low hit rate achieved and the sequential configuration (so a miss in the LUT means one more cycle per operation). In order to avoid this additional cycle, the LUT could be placed in parallel with the FPALU. However, in that case power savings would decrease.

In figure 5 we show the energy savings for different table sizes. This table shows that with very cheap tables (less than 1 KB) 10% energy savings can be achieved, while with more aggressive but still feasible tables (10 KB) savings can raise up to a 20%.

IV. Summary

In this paper we have evaluated the impact of the memoization technique for floating point operations in multimedia for the low-end domain. We have shown that although memoization could save some energy, the size of the tables required to achieve these results makes this technique not worth in a low power environment.

We have presented and developed the novel idea of fuzzy computation that could be used in multimedia algorithms to increase the hit rate of memoization tables by doing fuzzy memoization. This new technique has been implemented and evaluated in a low-end model processor and it has been shown that reuse is significantly improved. With realistic table sizes, considerable power and time savings are achieved, up to a 20% in energy and a 25% in the energy-delay product.

We believe that fuzzy computation opens a new paradigm for multimedia applications that will deserve future work. Currently we are evaluating other configurations, such as memoizing additions or memoizing complete FP functions (which we call Fuzzy Block Reuse). Also more benchmarks should be included.

Acknowledgments

This work has been supported by Direccio General de Recerca de la Generalitat under grant 1998FI-00260, by the Ministry of Education of Spain under contract CICYT TIC2001-0995-C02-01 and by the CEPBA.

References

[1] TI, "TMS320C62XX family," Tech. Rep. http://www.ti.com/sc/docs/products/dsp/tms320c6201.html, Texas Instruments, 1999.

[2] Philips Semiconductors, "Trimedia tm-1300," vol. http://www-us3.semiconductors.com/trimedia/, 1999.

[3] Analog Devices, "Introducing tigersharc," *Whitepaper*, vol. http://www.analog.com/new/ads/html/SHARC2.

[4] F. Arakawa, O. Nishii, K. Uchiyama, and N. Nakagawa, "Sh4 risc multimedia processor," March-April 1998.

[5] Stuart Franklin Oberman, "Design issues in high performance floating point arithmetic units," *PhD thesis, Stanford University*, November 1996.

[6] D. Citron, D. Feitelson, and L. Rudolph, "Accelerating multimedia processing by implementing memoing in multiplication and division units," *ASPLOS VIII*, 1998.

[7] Carlos Alvarez, Jesus Corbal, Esther Salami, and Mateo Valero, "On the potential of tolerance reuse for multimedia applications," *International Conference on Supercomputing, ICS-01, Sorrento*, Italy, 2001.

[8] Stephen E. Richardson, "Exploiting trivial and redundant computation," *11th IEEE Symposium on Computer Arithmetic*, 1993.

[9] Doug Burger and Todd M. Austin, "The simplescalar tool set, version 2.0," *Wisconsin-Madison CS Dep. technical Report #1342*, 1997.

[10] David Brooks, Vivek Tiwari, and Margaret Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," *International Symposium on Computer Architecture, ISCA27, Vancouver*, 2000.

[11] C. Lee, M. Potkonjak, and W.H. Magione-Smith, "Mediabench: A tool for evaluating and synthesizing multimedia and communication systems," *MICRO 30*, 1997.