Performance Evaluation and Improvements for Floating Point Computations

(Memory Based Computation)

Vivek Porush

Department of Electrical and Computer Engineering

University of Illinois Chicago

vporus2@uic.edu

I. Abstract

The purpose of this paper is to study and offer improvements in processor performance on complex floating-point functions such as square root, trigonometric and graphic functions. It is well known that such complex calculations require a lot of CPU effort (power consumption & CPI), which in turn results in an expensive application in terms of performance. Furthermore, performance of upcoming instructions is also deteriorated by such complex instructions, as forthcoming instructions have to wait until updated data is available. There are many techniques available to circumvent these penalties. For instance, using lookup tables (LTU) can result in reduction of these penalties and thus an improvement in performance. Applications that are data or computation intensive can be largely benefit from the use of LTUs. Moreover certain functions are more frequent than other functions thus favoring LTU implementation of these functions. Pre-computed functions over certain inputs could replace extensive runtime calculations resulting in a better performance.

However, in computer architecture space (memory) and time (computation) are of fundamental limitation. Thus a LTU implementation can become bottleneck for processor performance as the size and access time increases with complexity of LTUs. LTU performance can further be improved using an interpolation algorithm to generate reasonable approximations by averaging nearby samples. This will result in small sized LTU, which in turn will reduce accuracy of computation. However, recent advances in memory such as NRAM that has large size, high density and low power consumption can be beneficial in such scenarios.

In this paper we will evaluate these aspects and there effect on the performance (execution and power). Theoretically, the key idea is to and find a way to design future multi and many-core architectures that have better performance, spend less power and be easier to program all at a lower cost. Furthermore, we will try to specify architecture of certain specific applications such as 3-D integration technologies, and future graphics processors (GPUs) with performance more important than accuracy (non scientific applications such as real time game and video applications).

Keywords: CPU performance, Look up Tables, Power consumption, Multicore, Processor Architectures.

II. Motivation

The aim of computer architect is to design a system that has fast processor performance with minimum increase in cost. Traditionally we have basically utilized two methods to increase processor performance. These are:

- Hardware Optimization
- Software Optimization

In the hardware approach, the main aim of designer is to reduce chip geometry and increase clock frequency. On the other hand, software optimization results by exploiting various parallelisms such as ILP, DLP and TLP. State of art CMOS technology offers reduction in chip geometry along with increase in clock frequencies, however these advancements are accompanied with some serious disadvantages such as:

- Leakage Currents
- Excessive Power Consumption
- Excessive Heat Generation

Moreover, advantages of higher clock speeds are overshadowed by

- Memory Access Time
- "Von Neumann bottleneck" even with multithread/multiprocessing (memory operation)
- RLC delay
- Signal Transmission
- Possible New obstructions

Thus we can safely say that the hardware has become the ultimate bottleneck in present day processor architecture. The quest of a potential alternative to CMOS at the end of its roadmap has resulted in some new computational technologies. These technologies offer new approaches that can overcome limitations of conventional silicon based computers. Some examples include:

- Optical Computer
- Quantum Computers
- Molecular Computer
- DNA Computer

Although most of these emerging Nano devices are still in their early development phase, they hold tremendous potential in terms of integration density, low-power and low access times. However, investment by technologies giants such as Google, NASA and USRA in quantum computers (recently bought by these companies (oct 13) from D-wave) clearly indicate that CMOS is no longer able to keep up with our computational needs.

III. Why LUTs

An assuring solution to the challenges above is to use a reconfigurable memory-based computing (RMBC). A RMBC uses dense memory arrays (crossbar switches) for computation by organizing them as LUTs to hold the configuration for the mapped application. It is beneficial due to following reasons:

- Complex calculations require a lot of CPU effort (power consumption & CPI), which in turn results in an expensive application in terms of performance.
- Certain functions are more frequent than other functions thus favoring LTUs implementation.
- LUTs have already proven there success in image processing applications.
- Recent advances in memory such as MRAM, NRAM with low power consumption and high density
- Implementing LUTs results in computation stage unrestricted from underlying technology.
- Most applications have high temporal locality in data that can be exploited to reduce computation overhead.

In this paper we build on ideas proposed by Kamran Rahmani et al. [1] and Swarup Bhunia et al. [2] and propose an improvement in the memory based computation (MBC).

IV. Performance Evaluation

A detailed analysis between the performances of CMOS and RMBC based computers has been performed in [1], [2]. We will briefly discuss their design and then propose an improvement.

Figure1. Shows the architectural details of the RMBC. The computation is performed in following steps:

a) The Application is partitioned into smaller multi-input multi-output logic functions

b) These logical functions are threads as show in figure. These threads are then mapped to memory modules
c) A Scheduling table is formed which insures that these threads are issued such that the topological dependence among the partitions is satisfied
d) A partition is evaluated only when all its input values are available

The memory array shown in figure is an NRAM or Nano crossbar array with multiple inputs and outputs. Various threads are mapped onto these arrays and then evaluated.

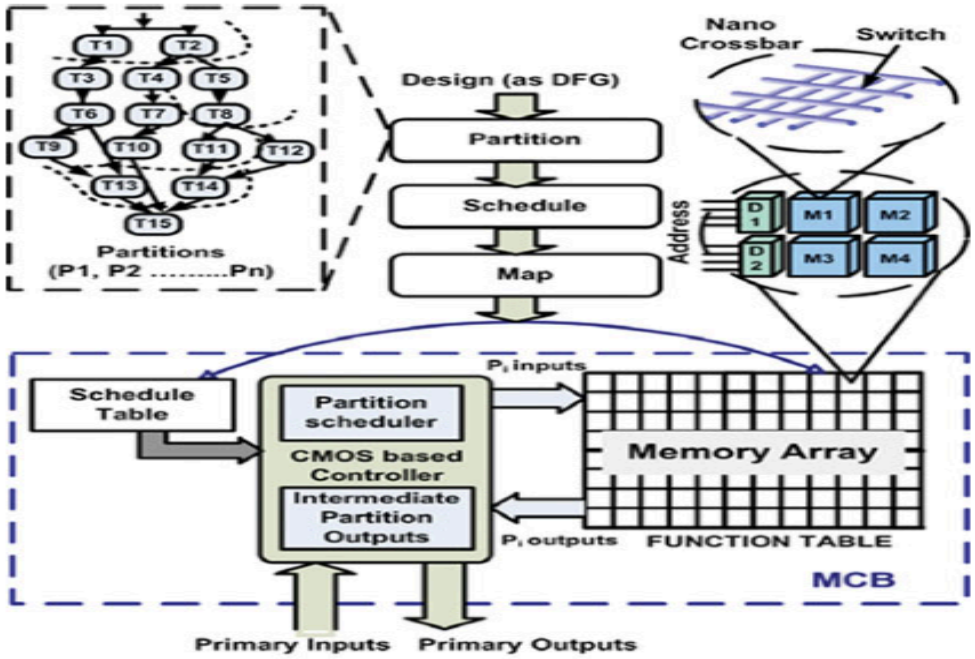


Figure.1. RMBC Architecture

A possible drawback of this arrangement is that for large applications the loading of function into these memory elements can result in high latency. These latency issues can be solved by utilizing techniques proposed in [2].

As proposed by Swarup Bhunia et al. [2] and shown in Figure.2. the memory latency can be hidden if not eliminated by utlizing a DMA. The DMA alocates the data bus to the memory element during intinsive memory transfers, thus hiding the latency.

Advantages of using NRAM:

LUTs have long been considered useless as an aid to computation because of the large size of memory they consume and high latency of memory operations. However, recent advances in memory such as MRAN and NRAM are capable of overcoming these disadvantages. Some of key features of NRAM are:

- Utilizes carbon Nano tubes and can achieve scaling up to atomic level.
- Scaling capability of NRAMs is only limited by present day lithography
- Much denser than DRAM (higher capacity)
- Nonvolatile nature.
- Activation energy of approximately 5eV thus very low power consumption.
- Less expensive eventually

Figure.3. show a typical NRAM cell.

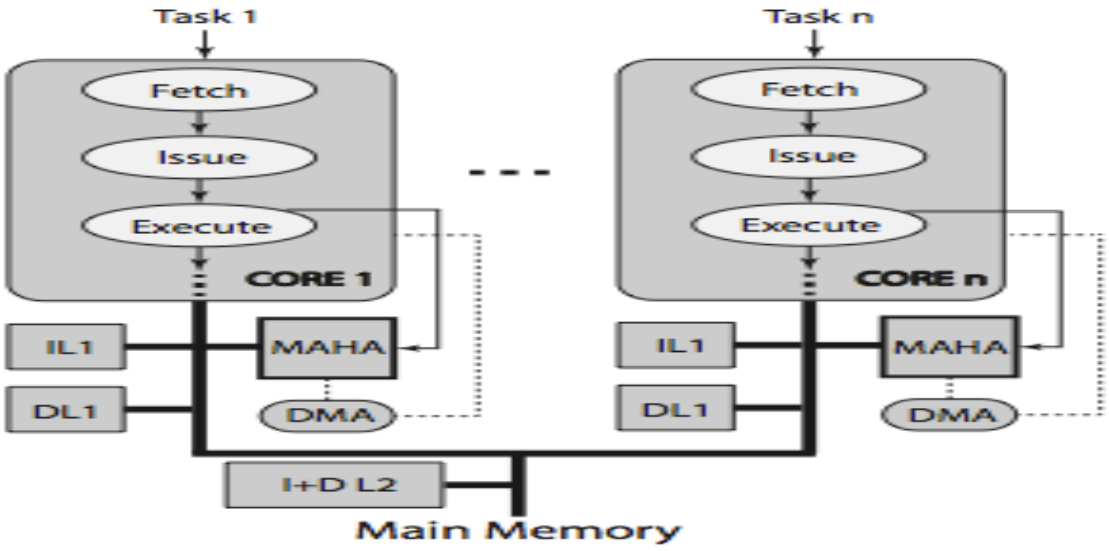


Figure.2 Hiding latency in RMBC

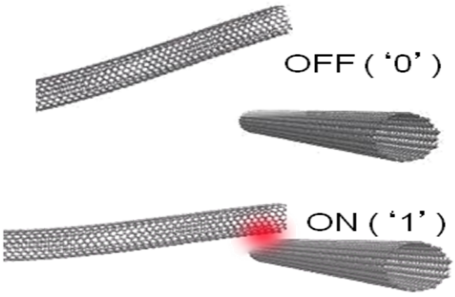
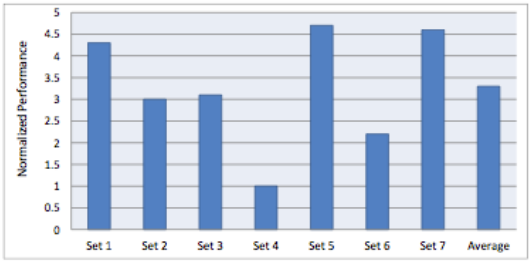


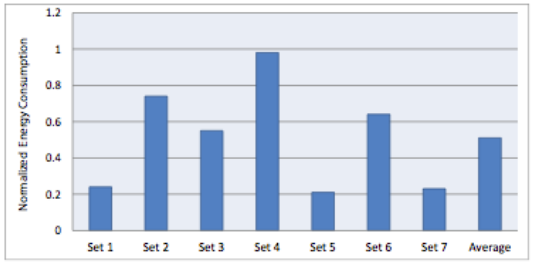
Figure.3. Typical NRAM structure

V. Results of Comparison between conventional and MBC

The following results are obtained from [1] & [2] which clearly show the comparison between conventional and MBC. Interested readers can refer to [1] & [2] for detailed explanation of these results. It is clear from the results that the results that the MBC performs very well and offers an improvement in performance (up to 4.7X, 3.3X on average) as well as reduce energy consumption (up to 4.7X, 2X on average) in multicore architectures.

Figure Performance and energy normalized to multicore without MBC





VI. Proposed Improvements

As explained in the class presentation we have used a simple example of real-time video call to illustrate our improvement. We have proposed that by making the LUT learnable in real-time we can switch the processor computation method between conventional and LUT based computations. This method will result in high performance processor architecture, which will be able to perform well for both the memory intensive calculations and conventional computations while maintaining initial design constrains (easy to program, low cost and high performance).

The method proposed by the authors of [1] & [2] does not take into account the variations in the LUTs. The method proposed in the aforementioned papers only issues a thread when it has all of its input variables available. Consider the following scenario (Figure.4) , in which a user is utilizing her computer for a real-time video call.

Performance of this video call depends on many factors such as network performance, number of applications running in parallel and processor performance. Let us consider that all of the above constrain are constant and do not degrade the performance of this call except for the processor performance. This assumption is fair as we are only concerned about the CPU performance in this paper.

Figure.4 (a) Anticipated call

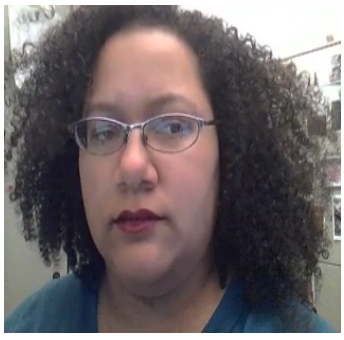


Figure.4 (b) Performance degraded call

Using the approach as suggested by the aforementioned papers can alleviate this degradation.

However, to make the LUTs learnable we can divide this application threads into a number of threads and monitor their inputs and outputs over the time in the same manner as in a branch prediction. This scenario is shown in following figure.

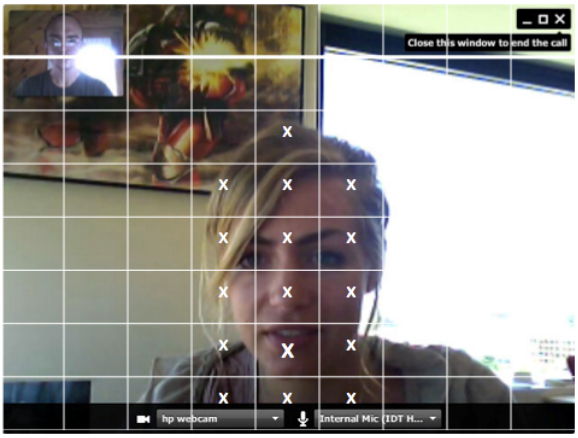


Figure.5 Breaking into multiple threads (Boxes) and monitoring their Next State.

From the above figure it is clear that only a small number of threads will require real time processing (face of the user marked with "x") and rest of the threads are relatively stationary.

Thus we can implement all the stationary threads via LUT and only the varying threads will require real-time processing.

If there is a change in stationary threads we can handle that change as an interrupt to the CPU and change the pixel value just before it appears on the monitor. One-dimensional

LUT are widely used in the monitor *gamma table*. These tables enable efficient modification of pixel intensity just before an image reaches the monitor. Thus there will be no performance degradation.

This method will result in a high quality of real-time video processing with little or no extra cost.

VII. Advantages of Implementing LUTs

There are a number of advantages offered by the implementation of LUTs some of these are:

- Simple design as we have FPGA
- Reduced design cost via batch processing
- Early time-to-market
- Rapid prototyping and
- Easily customizable hardware systems
- High-throughput
- Reduced latency
- Less dynamic power consumption due to minimization of switching activities in NRAM

VIII. Applications of MBC:

- Evaluating functions that are expensive to compute and inexpensive to cache.
- Digital filtering and DSP
- Real time games and video applications
- Future graphics processors (GPUs) with performance more important than accuracy
- General Purpose computing (accuracy decreases with precision)

The only concern with the implementation of this technique is interfacing NRAM with CMOS. As NRAMs have very less (5eV) as compared to the traditional CMOS logic. However, we have already developed sense amplifiers (Schmitt trigger) that can be used to interface NRAMs with CMOS logic.

IX. Conclusion

With the development of NRAM and similar technologies memory is no longer a bottleneck in the CPU performance. To fully utilize the advantages offered by these technologies we must use MBC models. I have proposed a possible improvement in the MBC by utilizing learnable LUTs. The theoretical analysis strongly suggests that proposed improvement will indeed result in increased performance in real time applications while maintaining comparable performance for conventional computations.

X. References

1. Kamran Rahmani et al. “Memory-based Computing for Performance and Energy Improvement in Multicore Architectures”. In University of Florida ESL publications 2012
2. Swarup Bhunia et al.“A Scalable Memory-Based Reconfigurable Computing Framework for Nano scale Crossbar”. In IEEE TRANSACTIONS ON NANOTECHNOLOGY, VOL. 11, NO. 3, MAY 2012
3. S. Majzoub et al. Reconfigurable platform evaluation through application mapping and performance analysis. In IEEE SPIT, 2006.
4. Pramod Meher ”Memory-Based Computing for DSP Applications”. Institute for Infocomm Research, Singapore, Dec 2010
5. B.Mouttet “Logic less Computational Architectures with Nanoscale Crossbar Arrays”. In NSTI-Nanotech, www.nsti.org, ISBN 978-1-4200-8505-1 Vol. 3, 2008

- Web references
    1. https://developer.nvidia.com
    2. http://www.nantero.com