Bioinformatics Specific Cell BE ISA Extensions

Sebastian Isaza and Georgi Gaydadjiev
Computer Engineering Laboratory
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Makelweg 4, 2628 CD, Delft, The Netherlands
{sisaza, georgi}@ce.et.tudelft.nl

***Abstract*— Among other traditionally important application domains, bioinformatics has been recently recognized as a challenging field. The need to perform lots of processing of huge amounts of data within reasonable times demands the use of high performance computing systems. Although this kind of high performance architectures already exists and can indeed provide some speedup, efficiency levels keep being low, both in terms of performance and power consumption. This is understandable since the traditional design of microprocessors has focused on more commercially attractive application domains like multimedia and networking. With the arrival of the multicore era, researchers have already started mapping bioinformatics applications onto platforms like Cell BE. This type of heterogeneous multicore chips have been shown to be capable of achieving significant speedups by exploiting thread-level and data-level parallelism at the same time. However, it has also been demonstrated that architectures like Cell BE have shortcomings that limit the performance efficiency when targeting bioinformatics workloads. Based on a previous work, we have augmented the instruction-set of the Cell BE SPU, with the aim of accelerating bioinformatics applications. We started looking at the possible improvements of the computational parts of a representative application of the field, i.e. ClustalW. Four arithmetic instructions have been tested with the CellSim simulator, achieving an accumulated acceleration of almost 14%.**

***Index Terms*—Bioinformatics, Instruction-Set Extensions, Cell BE, Application's acceleration**

I. Introduction

Bioinformatics has been recently recognized as a challenging field not only for computer scientists but also for hardware designers [9]. The rapid growth of biological databases and the computational complexity of the algorithms require high performance computing systems able to provide reasonable execution times.

Recent studies have shown that heterogeneous multicore architectures are able to provide significant performance improvements by means of multi-dimensional parallelism and specialization [6], [20], [12]. In a recent work [12], we presented the architectural limitations encountered when porting a representative bioinformatics application to Cell BE. This processor, which was designed for the game box market, has been demonstrated to have potential in other domains as well [18]. Cell Be architecture [13], [10] places on the same chip a PowerPC Processing Unit (PPU) and eight 128-bit SIMD cores named Synergistic Processing Units (SPUs). Each SPU has a 256KB Local Store (LS) memory and the Element Interconnect Bus (EIB) makes the connection between the nine processing cores. One particular feature of Cell BE is that there two different ISAs, that is, the PPU's ISA (PowerPC ISA) and the SPU's ISA. The latter is a SIMD-only instruction-set similar to the PowerPC's Altivec, aimed at improving the processing performance (and power efficiency) of the computationally intensive parts of applications.

Extending to our work presented in [12], but limiting our analysis to the SPU's ISA, we have added a few arithmetic instructions in order to accelerate the most intensive computational kernel of a representative bioinformatics application, ClustalW [11]. Although being compound-instructions, these are rather simple operations and could still be considered general purpose instructions useful for other applications' domains. Experiments have been performed using CellSim [1], an open source Cell BE simulator developed under the UNISIM framework [5]. The GCC compiler has been modified so that the new instructions can be called through intrinsics from the application's C source.

ClustalW is a widely used application to perform multiple sequence alignment. These kind of applications to align biological data are among the most important in bioinformatics [7]. They are used in different scenarios ranging from evolution studies to cancer research and drug design.

The main contributions of this paper are:

- the creation of new SPU instructions for the acceleration of a bioinformatics application;
- the implementation of those instructions within the CellSim simulator;
- the incorporation of intrinsics into the GCC compiler for the easy use of the new instructions;
- an almost 14% of acceleration obtained for the main kernel in ClustalW, adding only simple arithmetic instructions.

This paper is organized as follows: Section 2 describes the SPU architecture and presents a brief overview of recent works related to the mapping of bioinformatics applications and hardware support for its acceleration. Section 3 describes the experimental methodology used. Section 4 describes our example application and the newly introduced instructions. Section 5 presents the obtained results and Section 6 the conclusions and future work.

II. Related Work

The most innovative component of the Cell BE is the SPU. It was designed with the aim of achieving high data processing throughput while keeping a low power consumption. The SPU has a 256KB non-cacheble local store memory that can be accesses in 6 cycles. With the help of a DMA engine, data can be moved to and from the main memory, and this happening in parallel to the SPU program execution. The processing core

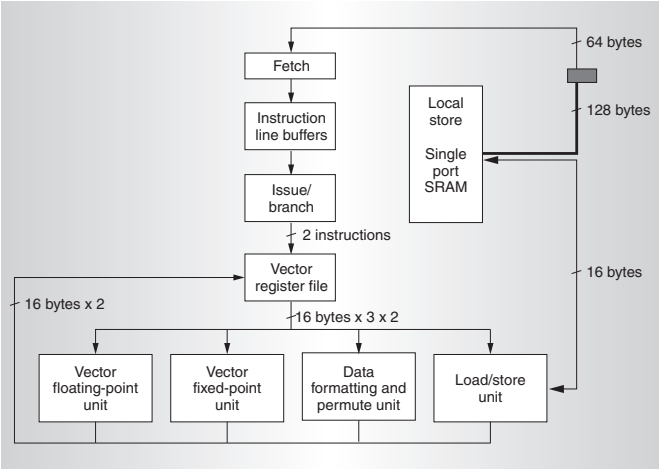

Fig. 1: SPU block diagram

of the SPU along with its local store are shown in figure 1. There are 128 registers of 128 bits width and two pipelines of 20 stages. Both scalar and SIMD execution take place in the same pipiline so that control complexity associated to handling separte scalar units is avoided. This however makes scalar processing less efficient. Schedulling of instructions and branch prediction are left to the compiler so that control logic can be simplified, achieving high frequencies at low power. Control-dominated codes with many conditionals should be left to the PPU since the SPU will poorly process them. Further circuitry simplification is pursued by only allowing 16-byte aligned memory accesses (both to the local store and through the DMA engine). The compiler should make sure accesses are aligned by introducing additional loads/stores and data reorganization instructions. This in turns may affect performance depending on the data-access patterns.

Since the release of the Cell BE processor, there have been a number of articles reporting experiences of mapping bioinformatics applications to this new architecture. Some of them have used sequence alignment applications and explored parallelization strategies and manual code optimizations [6], [20], [18], [17], but none of them have evaluated the instruction set efficiency. Those works have shown that both parallelization and the use of specialized cores can provide significant speedups, yet staying within a general-purpose platform. However, the traditional focus on multimedia and networking applications have not allowed bioinformatics to efficiently benefit from current architectures. In our previous article [12], we investigated the limitations of the Cell BE architecture when targeting bioinformatics. Some of the limitations mentioned there have been considered here by adding new instructions.

Hardware support for bioinformatics has been explored in the form of fully custom accelerators, most of them using FPGAs [9], [15], [16], [21], [8] and some ASICs [19]. Although capable of achieving very significant speedups, FPGA solutions suffer from low power efficiency and programability issues. On the other hand there is also the case of Anton [19]. D.E. Shaw Research is about to finish the implementation of this new special-purpose machine to target molecular dynamics simulations. Although containing a programmable processor core, Anton's architecture is fully application-specific aiming at very high speedups for a single application regardless of the high cost of such a solution.

To the best of our knowledge, there has not been previous works on instruction-set extensions targeting bioinformatics applications. It should be noticed however, that in a timid attempt Intel has recently announced that SSE4.2 (the newest version of their SIMD instruction-set extensions) will feature one instruction aimed at genome mining (an application closely related with sequence alignment). Although this fact shows a new interest to offer support for bioinformatics applications from the general-purpose perspective, still much is to be investigated so that efficiency gains are significant.

III. Experimental Methodology

The evaluation of our instruction-set extensions has been done using CellSim. Based on UNISIM framework, CellSim is a modular cycle-level simulator that allows one to test modifications to Cell BE architecture. The behavior of instructions can be described in C language. Additionally, in order to make easy use of the new instructions, we have added them as intrinsics to the CellSim GCC compiler. The code running on the SPU has been instrumented in order to call the CellSim profiler to measure the cycles taken by the function of interest.

Since our proposed instructions are independent of the use of multiple cores and due to the long simulation times, we have only simulated the kernel function of ClustalW running on one SPU and processing two 50-symbols long input sequences. The previous conditions do not affect the analysis since the number of loop iterations and computations only depend on the inputs size but not on the data content. That is, the speedup will remain the same for any other input data.

IV. Adding instructions to the SPU ISA

New instructions have been chosen by first profiling and then by manual inspection of the kernel code. This section describes ClustalW application, in particular its most time consuming function and then present the instructions proposed.

A. Application's description

ClustalW [11] is an application used to perform multiple sequence alignment. Profiling of the code revealed that the first phase of the application consumes most of the execution time. A function called *forward_pass* takes between 60% and 80% of the total runtime depending on the inputs sizes. This function implements a slightly modified version of the Needleman-Wunsch algorithm [14], computing a similarity

measure of two sequences. In our experiments we have used the vectorized and optimized version of *forward_pass* that we presented in [12]. *forward_pass* is composed of two nested loops that iterate over the entire length of the input sequences. The body of the inner loop is then the target of our analysis for new instructions.

B. Instructions added

As mentioned before, the SPU ISA is based on SIMD processing, that is, instructions operate on 128-bit vector operands. In the case of *forward_pass*, vectors contain eight 16-bit elements to be processed in parallel. The instructions proposed in this work are also SIMD.

Inspection of the loop body code shows that it is based on additions, subtractions and conditionals that find maximum values. According to this observation we have tested four new arithmetic instructions whose functionality is shown in table I.

INSTRUCTION	FUNCTION
MAX Rt,Ra,Rb	if(Ra>Rb) Rt=Ra else Rt=Rb

MAX3Z Rt,Ra,Rb,Rc	if(Ra>Rb&&Ra>Rc) temp=Ra
	else if(Rb>Rc) temp=Rb
	else temp=Rc
	if(temp>0) Rt=temp
	else Rt=0

SUBMX Rt,Ra,Rb,Rc	temp=Ra-Rb if(Rc<temp) Rt=temp else Rt=Rc
ADDS Rt,Ra,Rb	S=Ra+Rb if(Ra>0&&Rb>0&&S<0) Rt=2^{16}-1 else if(Ra<0&&Rb<0&&S>0) Rt=-2^{16} else Rt=S

TABLE I: Functionality of the new SPU instructions

In short, MAX and MAX3Z find maximums, SUBMX finds a maximum after a subtraction and ADDS perform saturated (signed) addition.

V. Results

Using the CellSim profiler we have instrumented the source code running on the SPU. Numbers reported correspond to the cycles consumed by ClustalW kernel, *forward_pass*. Table II shows the percentage of acceleration achieved with respect to the vectorized and optimized version using the original SPU ISA. The third column indicates how many old SPU instructions are needed to emulate the proposed ones. It should

INSTRUCTION	ACCELERATION	EQ. IN.
MAX	5%	2
MAX3Z	3%	6
SUBMX	4%	3
ADDS	3.5%	9
TOTAL	13.8%	-

TABLE II: Kernel acceleration results after adding the new instructions and equivalent ”old” SPU instructions

be noticed that the total acceleration achieved does not equals the sum of individual contributions. This is because the use of MAX3Z overlaps some instances of MAX.

Although in principle these instructions should be able to provide more speedup, many cycles are being consumed by data reorganization instructions (masks creation, rotates, shifts, shuffles). These instructions are inserted by the compiler in order to make possible scalar manipulations and unaligned memory accesses, not supported by the SPU hardware.

VI. Conclusions and future work

We have presented the performance results of adding 4 instructions to the Cell BE SPU, targeting bioinformatics applications. Acceleration of almost 14% has been achieved for the hot-spot function in ClustalW. Irregular memory access patterns in *forward_pass* are responsible for a lot of time consumed in the body loop. These need scalar manipulations and unaligned memory accesses that are costly in the SPU. Our next step is to focus on this issue in order to add instructions that facilitate these operations.

One important observation is that our proposed instructions are fundamental not only in *forward_pass* kernel but in various other critical functions of ClustalW, and in other applications of the same field like Ssearch (FASTA package [2]) and Hmmer [3]. Incorporating our instructions into those other functions and applications is part of our future work.

Our mid term intention is to explore hardware acceleration of different granularities. We are starting by investigating the potential of rather simple general-purpose-like instruction-set extensions. Afterwards we will evaluate more coarse-grained and complex instructions and lastly we will explore the design and integration of large and more customized accelerators into the heterogeneous multicore architectural template being developed in the SARC project [4].

Acknowledgments

This work is being supported by the European Commission in the context of the SARC integrated project #27648 (FP6).

References

[1] *Cellsim*, http://pcsostres.ac.upc.edu/cellsim/doku.php.
[2] *Fasta web site*, http://fasta.bioch.virginia.edu/fasta_www2/fasta_list2.shtml.
[3] *Hmmer web site*, http://hmmer.janelia.org/.
[4] *Sarc project*, http://www.sarc-ip.org/.
[5] *Unisim*, http://unisim.org/site/.
[6] F. Blagojevic, A. Stamatakis, C. Antonopoulos, and D. Nikolopoulos, *Raxml-cell: Parallel phylogenetic tree inference on the cell broadband engine*, Proceedings of the 21st IEEE/ACM International Parallel and Distributed Processing Symposium, March 2007.
[7] J. Cohen, *Bioinformatics-an introduction for computer scientists*, ACM Computing Surveys (2004), 122–158.
[8] P. Faes, B. Minnaert, M. Christiaens, E. Bonnet, Y. Saeys, D. Stroobandt, and Y. Van de Peer, *Scalable hardware accelerator for comparing dna and protein sequences*, Proceedings of the First International Conference on Scalable Information Systems, May 2006.
[9] M. Gokhale and P. Graham, *Reconfigurable computing*, Springer, Dordrecht, The Netherlands, 2005.
[10] M. Gschwind, H.P. Hofstee, B. Flachs, M. Hopkins, Y. Watanabe, and T. Yamazaki, *Synergistic processing in cell's multicore architecture*, IEEE Micro (2006), 10–24.
[11] D. Higgins, J. Thompson, T. Gibson, and J.D. Thompson, *Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice*, Nucleic Acids Research **22** (1994), 4673–4680.

[12] S. Isaza, F. Sanchez, G. N. Gaydadjiev, A. Ramirez, and M. Valero, *Preliminary analysis of the cell be processor limitations for sequence alignment applications*, Proceedings of the 8th International Workshop SAMOS 2008, July 2008.

[13] J.A. Kahle, M.N. Day, H.P. Hofstee, C.R. Johns, and D. Shippy, *Introduction to the cell multiprocessor*, IBM Systems Journal **49** (2005), no. 4/5, 589–604.

[14] S. Needleman and C. Wunsch, *A general method applicable to the search for similarities in the amino acid sequence of two proteins*, Journal of Molecular Biology **48** (1970), 443–453.

[15] T. Oliver, B. Schmidt, and D. Maskell, *Hyper customized processors for bio-sequence database scanning on fpgas*, Proceedings of 17th International Symposium on Field Programmable Gate Arrays, February 2005.

[16] T. Oliver, L.Y. Yeow, and Schmidt, *High performance database searching with hmmer on fpgas*, Proceedings of the 6th Workshop on High Performance Computational Biology, February 2007.

[17] F. Petrini, G. Fossum, J. Fernandez, A.L. Varbanescu, M. Kistler, and M. Perrone, *Multicore surprises: Lessons learned from optimizing sweep3d on the cellbe*, IEEE International Parallel and Distributed Processing Symposium, IPDPS (2007), 1–10.

[18] V. Sachdeva, M. Kistler, E. Speight, Tzeng, and T.H.K., *Exploring the viability of the cell broadband engine for bioinformatics applications*, Proceedings of the 6th Workshop on High Performance Computational Biology, 2007, pp. 1–8.

[19] D. E. Shaw, M. Deneroff, R. Dror, J. Kuskin, R. Larson, J. Salmon, C. Young, B. Batson, K. Bowers, J. Chao, M. Eastwood, J. Gagliardo, J. Grossman, R. Ho, D. Ierardi, I. Kolossvry, J. Klepeis, T. Layman, C. McLeavey, M. Moraes, R. Mueller, Y. Shan E. Priest, J. Spengler, M. Theobald, B. Towles, and S. Wang, *Anton: A special-purpose machine for molecular dynamics simulation*, Proceedings of the 34th Annual International Symposium on Computer Architecture, June 2007.

[20] H. Vandierendonck, S. Rul, M. Questier, and K. De Bosschere, *Experiences with parallelizing a bio-informatics program on the cell be*, Proceedings of the 3th International Conference on High Performance and Embedded Architectures and Compilers, 2008, pp. 161–175.

[21] J.P. Walters, X. Meng, V. Chaudhary, T. Oliver, L.Y. Yeow, D. Nathan, B. Schmidt, and J. Landman, *Mpi-hmmer-boost: Distributed fpga acceleration*, Journal of VLSI Signal Processing (2007), 223–238.