Proceedings of the 5th International Conference on Information Technology and Application in Biomedicine, in conjunction with
The 2nd International Symposium & Summer School on Biomedical and Health Engineering
Shenzhen, China, May 30-31, 2008

Real-time VLSI architecture for bio-medical monitoring

V. A. Chouliaras, S. Hu, R. Summers, A. S. Echiadis, V. Azorin-Peris, I. King and J. Zheng

Abstract - **This paper discusses the architecture and implementation of SSS2, a high-performance real-time signal processing system developed with a hybrid ESL/RTL methodology and targeted to biomedical image processing. Traditional methodologies, as well as new tools, such as Cebatech's C2R untimed-C synthesizer have been employed in the design of the system. The SSS2 platform specifies a parametric number of scalar processing elements, based on multiple 32-bit Sparc-compliant engines, augmented with LE2, an ESL-designed 2-way LIW/SIMD accelerator. LE2, which is purely designed in C, exposes a consistent interface to its SIMD datapath directly which is directly derived from the C-source of open-source image processing codes. It is synthesized to Verilog RTL with C2R. Behaviorally-synthesized SIMD datapaths are then 'plugged-in' into the exposed LE2 datapath interface. The LE2 memory interface can be either a cache-based configurable vector load/store unit or a multi-banked, multi-channel streaming local memory system. Results drawn from this work strongly suggest a shift towards a hybrid approach in designing multi-core systems for high bandwidth streaming and for dealing with large scale medical image transfers and non-linear bio-signal processing algorithms.**

I. Introduction

In a clinically diagnostic environment, in-vitro and in-vivo assessment could be critical for the clinicians to make key medical decisions and perform medical interventions safely, accurately and quickly as these will be based on hard facts, derived in real-time from physiological data. Existing VLSI architectures are capable of delivering a practical solution to processing in real-time physiological data, particularly in biomedical image processing. They are thus invaluable in blood perfusion monitoring and oxygen consumption mapping. This unique capability afforded by high performance processing platforms, will drive surgical decisions such as the precise identification of a tumor boundary and it's subsequent removal (with guaranteed safety margins) thus prolonging human life and reducing post-operative risks and complications.

VLSI platforms based on embedded processor cores with a fixed Instruction-Set-Architecture (ISA) have been widely used in the past. Such architectures present a good compromise for the execution of general-purpose code, such as the user interface, protocol processing and an embedded operating system. However, they lacked considerably in the area of Digital Signal Processing (DSP), as needed by almost all of the core image processing algorithms needed for real-time biomedical monitoring. To increase the signal processing capability of such systems, system architects typically utilize a number of additional embedded DSP cores, in parallel to the main scalar processor core, to accelerate the performance and the critical parts of the application. This nevertheless comes at the expense of increased silicon area and utilization of a convoluted programming model due to the multiple address spaces, ISAs, and 'mailbox-type' communications. A possible solution to these issues is the hardwired implementation of the core DSP functionality; however this involves the development and validation of thousands of lines of parallel code at the register transfer level (RTL) and results in solutions that are of high performance yet, they are only tuned to the task at hand and offer little or no programmability. The latter is a serious deficiency in the targeted biomedical market as well as in more contemporary markets such telecoms and consumer; the latter are characterized by short time-to-market and associated market windows and ever-evolving standards.

Over the past few years, a promising processing paradigm has been increasingly utilized in such high performance SoCs. This paradigm has the form of configurable, extensible processors which allow the extension of their architecture (programmers model and ISA), and microarchitecture (execution units, streaming engines, custom coprocessors) by the system architect [1]. Configurable and extensible processors offer, on top of the very high performance, the added advantage of post-fabrication adaptability to evolving standards through the careful choice of custom ISA and execution/storage resources. Such diverse applications range from traditional consumer applications, such as video coding [2] [3] and audio processing [4], to less obvious domains, such as RTOS acceleration [5].

A third proposition for the modeling and to a lesser extent, implementation of high performance SoCs, comes from a number of vendors in the form of co-design environments and RTL synthesis systems for electronic system-level (ESL) design languages such as SystemC [6]. This presents an interesting prospect for designing and modeling the consumer ASIC in a parallel language and in the process, creating an executable specification for high-speed validation, as well as for the final implementation. Extending the SystemC concept, the authors in [7] discuss an object-oriented system-level design specification and implementation flow based on the transformation of UML to SystemC. Similarly, SystemC based at transaction-level has

The authors are with the Department of Electronic and Electrical Engineering, Loughborough University, Loughborough, LE11 3TU, UK. (0044 (0) 1509 227 113); e-mail: v.a.chouliaras@lboro.ac.uk).

been used successfully in a co-design flow to model complex SoCs [8]. In [9], the authors propose NetC as a means of modeling and evolving Networks-on-Chip while producing cycle-accurate models in SystemC, whereas the authors in [10] utilize SystemC as both a modeling language and an implementation medium for a high-performance Network-on-Chip architecture.

It is only very recently that the introduction of powerful, system-level behavioral synthesis technology [11], has enabled a full, untimed C-based flow to be used directly to transform complex application sources into hardwired silicon, without the need for a single or multi-core programmable platform. In that respect, this work utilizes that flow, along with a more traditional (and laborious) RTL codebase to propose a configurable, 2-way Long Instruction Word (LIW) architecture that can easily adapt to the data and instruction parallelism, typically found in medical imaging processing codes.

The novelty of our work is three-fold: Firstly, this work fuses the configurable processor and ESL implementation domains in a novel way through using the later as the implementation medium of custom SIMD extensions. Secondly, it discusses the micro-architecture of a novel, high performance, configurable, extensible ASIC processor (a 2-way LIW/SIMD) which is capable of exploiting a moderate amount of instruction level parallelism (ILP) and substantial amounts of data level parallelism (DLP), typically found in image filtering applications. The later form of parallelism is enabled by a new, C-based VLSI synthesis flow. Finally, a configurable number of such LE2 augmented Sparc processors are brought together in a cache-coherent ecosystem which includes a multi-bank, multi-client local memory subsystem to satisfy the demands of the LE2 accelerators. The whole microelectronics system, known as the SSS2 platform, provides the core image processing capability of the Oximap tomographer [12].

II. Requirements for Biomedical Monitoring

The presented biomedical image processing system consists of three core subsystems: a) The optical subsystem, b) the electronic subsystem, and c) the micro-electronic subsystem. More specifically, the optical subsystem comprises of a high-speed image sensor coupled to an optical lens, and a multi-wavelength illumination unit which consists of large arrays of high-brightness light emitting diodes (LED).

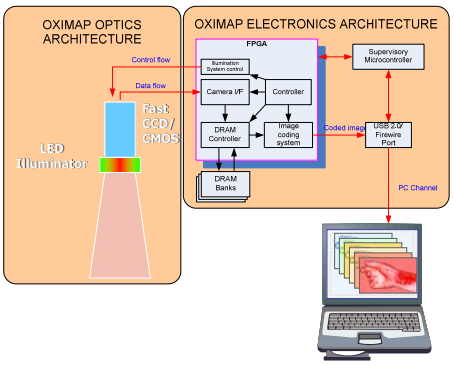


Fig. 1: Biomedical Imaging Platform Architecture.

The electronic subsystem provides accurate control of the constant current source and directly drives the LED arrays with high-current pulse trains, as well as it converts and distributes the supply voltage to the other subsystems.

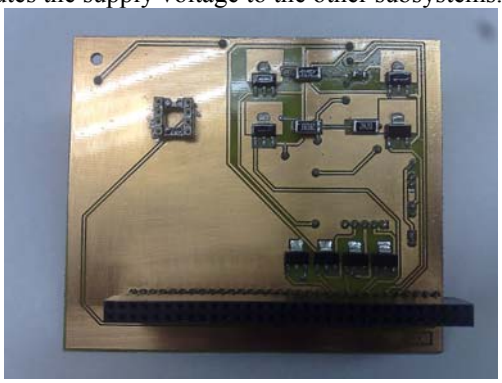
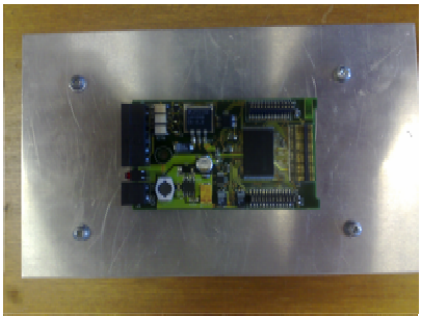
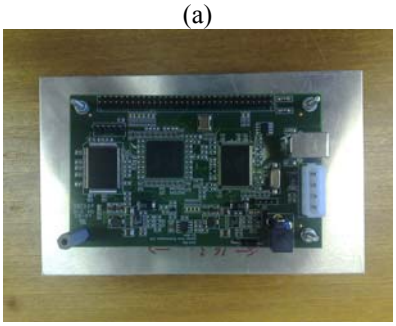


Fig 2: Analog Electronics Subsystem.

Finally, the microelectronics processing subsystem consists of a Field Programmable Gate Array (FPGA) device in which the processing platform resides, a 1Gb DDR2 memory and a USB2.0 module which handles the communication with a computer. The control of the image sensor is also handled in this subsystem, where the sensor signals are fed into a 16-bit A/D converter.



(a)



(b)

Fig 3: a) Camera Control FPGA b) Main microelectronics platform

The performance of the whole system is greatly dependent upon the sensitivity of the image sensor. Typically, image sensors are more sensitive in the visible light spectrum (400-700nm) and their sensitivity, or quantum efficiency (Q.E.), gradually drops in the near-infrared region (>750nm). The selection of the image sensor is extremely important as the system employs wavelengths from 600nm to 1200nm, therefore the sensor must have a high Q.E. in the near-infrared region in order to be able to register the plurality of information contained in biological signals. Also, depending on the application of this platform, the sensor sensitivity can have a high or a low impact on the quality of information. For example, measurement of blood perfusion over a tissue area does not require extremely high sensitivity due to the strong light absorption characteristics of chromophores in blood. However, measurement of blood glucose will require a high sensitivity in order to be able to distinguish the weak light absorption of glucose between different wavelengths.

This real-time biomedical signal processing platform can be applied to numerous applications, with each particular application presenting its own requirements for sensitivity and algorithmic complexity. In order to provide flexibility in terms of the application, the system is designed such that to provide maximal sensitivity and processing power in order to meet the expectations of high-end applications, while providing backward compatibility to more conventional and less demanding biomedical signal processing algorithms.

III. LE2 Programmers Model and ISA

The programmers model specifies a parametric (VREGS, 16 max) number of vector registers, at 16-bit granularity. The maximum vector length is a compile-time constant and can be up to 4096 16-bit elements wide (VLMAX). There's a run-time (dynamic) vector length register (VLEN) which identifies how many scalar elements of the source and destination vector registers participate in the current computation. The predicate register functions as a further mask for up to VLEN elements. Finally, there are a parametric number of scalar registers (SREGS, 16 max), each 32-bit wide and two vector accumulators (VACC0/1), each VLMAX. There are two vector accumulators, each 32-bit wide and VLMAX/2 32-bit elements in length.

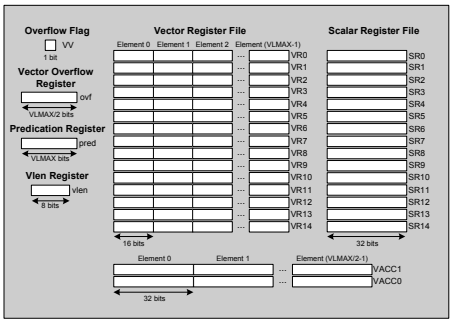


Fig. 4. LE2 coprocessor programmer's model.

IV. RTL Microarchitecture

The LE2 engine is depicted in Fig. 5. It is a parametric, 8-stage pipelined microarchitecture (Decode, address generation, memrd1, memrd2, exec1,exec2, pre-wb, wb). It is designed to be closely-coupled to either a Leon2 or Leon3 scalar engine, at stages 4 and 6 respectively, effectively being in series to the scalar engine. The benefits of this configuration include the zero-cycle latency when transferring state to the main scalar core and the correct operation under a multiple-exception-source regime. As such, the vector processor presents a precise exception model to the programmer. An interesting observation is the use of vector lanes 0, 1 (2x16-bit) for the execution of scalar operations thus dispensing with the use of a separate scalar datapath for the non-vectorizable sections of the code.

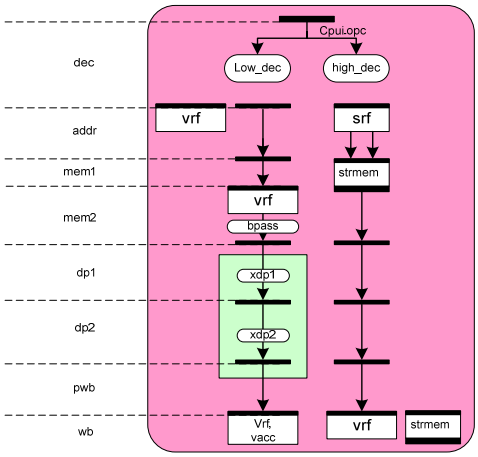


Figure 5: LE2 microarchitecture (Green section identifies C2R-designed blocks)

V. SoPC EcoSystem

Fig. 6 depicts the larger system context which makes use of multiple Leon2's + the hybrid accelerator discussed above in the context of the Oximap system. It consists of a parametric number of such modified Leon2 CPUs, each with an attached LE2 engine, in a cache-coherent configuration (single AHB, transaction snooping and a write-through cache).

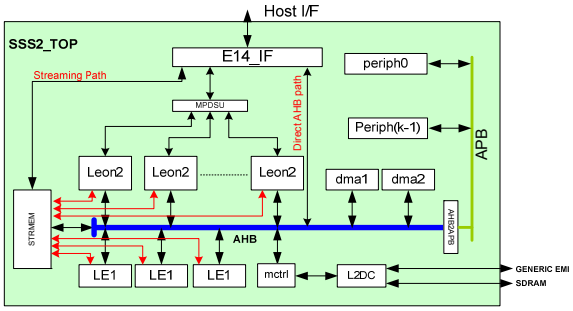


Fig. 6. The SSS2 SoPC.

The high-bandwidth data traffic of the data engines (including both LE1 and LE2) is serviced by multi-bank, multi-client, DMA-driven distributed memory blocks (STRMEM). This is depicted in Fig. 7

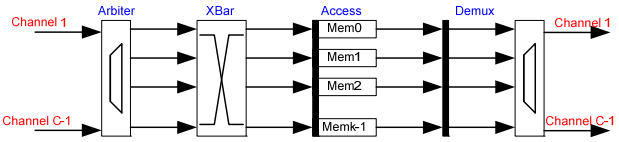


Fig. 7. Distributed, multi-bank, multi-client local memory architecture

The system includes a custom, multi-core debug support unit (MP-DSU), to provide debug access to the individual processors, multiple LE1 VLIW processors (an 8-stage implementation of the HPL-PD architecture [13], to be discussed in a future journal), two DMA engines, serving the AHB and the STRMEM. The latter is (global) memory-mapped at address 0x30000000, thus being accessible by all the processing elements and streaming engines. Finally, there's a configurable level-2 data cache (cascade configuration), serving the base-band traffic (OS, control code, LE1 IRAM streaming) and decoupling the external DDR2 from the continuous write traffic of the multiple write-through L1DC's of the Leon2 engines.

The system has been prototyped in a PicoComputing E14 cardbus board (Fig. 8). The board is controlled from a host x86 Windows laptop and full streaming capability (at 132 MB/s) into and out of the STRMEM is allowed.

Fig. 8: Pico E14 prototyping board

A visual environment has been built to allow access to the state of the multiple engines, download application code and extract processed data streams from the SSS2 platform. The platform is programmed in C and assembly, under Linux, and via a make-automated process, the scripting-assisted compilation phase produces a final file which is the image of the global memory of the system. This is transferred by the application programmer onto the STRMEM and the DDR2 of the SSS2 platform, as shown in Fig. 9; execution of the individual processors is handled in a very intuitive and user-friendly way.

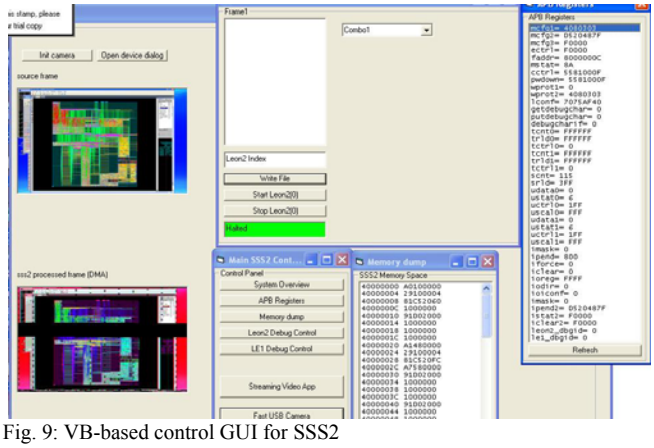


Fig. 9: VB-based control GUI for SSS2

VI. Conclusion and Future Work

This work presented the design, performance modeling and implementation of a 2-way LIW/SIMD engine designed with a hybrid RTL/ESL methodology and targeting high-speed, real-time biomedical imaging applications. Our group has also undertaken the task of a full RTL design of a second vector engine targetting voice-over-IP (VoIP) telephony applications. Through our experience with both flows, we have built significant confidence in the use of advanced tools such as C2R for the design of high-performance programmable engines. In that respect, the LE2 engine is slowly being modified, with major RTL blocks removed and replaced by C2R-logic; the behavioral synthesizer is very mature and allows for precise control of clocks and interfaces thus enabling the design of large-scale programmable engines in a fraction of the time required by existing methodologies. We will be reporting on our findings in the near future.

Acknowledgment

The authors gratefully acknowledge *Cebatech Inc.* for the donation of research licenses of their flagship C2R product as well as unrestricted access to in-house expertise.

References

[1] Leibson, S.; Kim, J, *"Configurable processors: a new era in chip design"*, IEEE Computer, July 2005, vol.38, no.7, pp. 51-59

[2] Mbaye, M.; Belanger, N.; Savaria, Y.; Pierre, S, *"Application specific instruction-set processor generation for video processing based on loop optimization"*, IEEE International Symposium on Circuits and Systems (ISCAS) (IEEE Cat. No. 05CH37618), 2005, Vol. 4, pp. 3515-3518, vol. 6

[3] V. A. Chouliaras, J. L. Nunez, K. Koutsomyti, S. R. Parr, D. J. Mulvaney, S. Datta, 'On the development of a custom vector accelerator for high-performance speech coding', IEE Electronic Letters, Vol. 40, Issue 24, 25 Nov. 2004, pg 1559-1561

[4] Bower, J, "A system-on-a-chip for audio encoding", Proceedings of the 2004 International Symposium on System-on-Chip, Nov. 2004 Page(s):149 - 155

[5] Zhenyu; Sindhwani, M.; Srikanthan, T. Edited by: Diessel, O., Williams, J., "RTOS acceleration on soft-core processors using instruction set customization", Proceedings of the 2004 IEEE International Conference on Field- Programmable Technology, pp. 371-374

[6] http://www.celoxica.com/techlib/files/CEL-W050520101L-335.pdf

[7] Luo Juan; Cao Yang; Jiang Jian-Lin, 2005 International Conference on Communications, Circuits and Systems. Volume II. Signal Processing, Computational Intelligence, Circuits and Systems (IEEE Cat. No. 05EX1034), 2005, pp. 1343-7, vol. 2

[8] Moussa, I.; Grellier, T.; Nguyen, G. Edited by: Wehn, N., Verkest, D "Exploring SW performance using SoC transaction-level modeling", Proceedings Design, Automation and Test in Europe Conference and Exhibition, 2003, pg. 120-125

[9] Liwei Ma; Yihe Sun, "On-chip network evolution using NetC", Proceedings of the 2005 IEEE VLSI-TSA International Symposium on VLSI Design, Automation & Test (VLSI-TSA-DAT), pp. 249-252

[10] Bertozzi, D.; Benini, L., "Xpipes: a network-on-chip architecture for gigascale systems-on-chip", IEEE Circuits and Systems Magazine, 2004, vol.4, no.2, pp. 18-31

[11] http://cebatech.com/c2r.php

[12] www.oximap.co.uk

[13] www.trimaran.org