Multi-Core Architecture Design for Ultra-Low-Power Wearable Health Monitoring Systems

Ahmed Yasir Dogan*, Jeremy Constantin†, Martino Ruggiero‡, Andreas Burg† and David Atienza*
*Embedded Systems Lab. (ESL) - EPFL, Lausanne, Switzerland. Email: {ahmed.dogan,david.atienza}@epfl.ch
†Telecommunications Circuits Lab. (TCL) - EPFL, Lausanne, Switzerland. Email: {jeremy.constantin,andreas.burg}.epfl.ch
‡Micrel Lab - University of Bologna, Viale Risorgimento 2, 40136, Bologna, Italy. Email:martino.ruggiero@unibo.it

***Abstract*—Personal health monitoring systems can offer a cost-effective solution for human healthcare. To extend the lifetime of health monitoring systems, we propose a near-threshold ultra-low-power multi-core architecture featuring low-power cores, yet capable of executing biomedical applications, with multiple instruction and data memories, tightly coupled through flexible crossbar interconnects. This architecture also includes broadcasting mechanisms for the data and instruction memories to optimize system energy consumption by tailoring memory sharing to the target application. Moreover, the architecture enables power gating of the unused memory banks to lower leakage power. Our experimental results show that compared to the state-of-the-art, the proposed architecture achieves 39.5% power savings at high workload requirements (637 MOps/s), and 38.8% savings at low workload requirements (5 kOps/s), whereby leakage power consumption dominates.**

I. Introduction and Related Work

Cardiovascular and modern human behavior-related diseases require accurate and continuous medical supervision, which is unsustainable for traditional healthcare delivery systems due to increasing healthcare costs and medical management needs [1]. Personal health monitoring systems are poised to offer large-scale and cost-effective solutions to this problem. The use of wearable, miniaturized and wireless sensors nodes, able to continuously measure and wirelessly report cardiac and other biomedical signals, can indeed provide the ubiquitous, long-term and real-time monitoring required by the patients, and enables faster coordination with medical personnel.

Recently, significant industrial and academic efforts have been dedicated to develop online automatic biomedical signal analysis on wearable personal health systems. Although several commercial products and research prototypes have been developed, especially for ambulatory heart-rate monitoring: Toumaz's Sensium Life Pebble [2], Corventis's PiiX [3] or IMEC's prototype of a single-lead bipolar electrocardiogram (ECG) patch [4], state-of-the-art unobtrusive health monitoring systems are either very simple regarding on-line signal processing (i.e., mostly signal filtering and simple signal analysis) or limited in overall autonomy due to their limited energy efficiency for advanced biopotentials processing [5].

An effective technique to achieve energy efficiency is supply voltage scaling. In the literature, voltage scaling has been extensively analyzed, including its limitations and disadvantages [6], [7], [8]. One of the main issues with low-voltage operation is performance degradation, which can limit the degree of use of voltage-scaling for a given processing requirement. Parallel computing using multiple cores can alleviate this issue, provided that the algorithms to be executed can be parallelized. To this end, the work presented in [9] explored the power/performance tradeoffs between sequential and parallel near-threshold computations for various biomedical signal processing requirements. The comparison shows that multi-core systems do not only solve the performance degradation problem, but also achieve good energy efficiency. Dreslinski et al. [10] proposed a near threshold computing (NTC), cluster-based multi-processor architecture with a shared cache that operates at a higher supply voltage to be able to serve multiple cores at the same time. Also, Yu et al. [11] introduced a sub/near threshold processor specialized for low-energy mobile image processing using architecture-level parallelism to compensate the performance loss. The multi-core architectures in [9] and [10] are built for a more general use, however they achieve limited energy efficiency, notably at low workloads.

Our ultimate goal in this paper is to synergistically exploit NTC in conjunction with multi-core architecture design to enable ultra-low-power (ULP) wearable health monitoring systems. The main contributions of this paper are the following:

1) We assess the feasibility of developing an ULP multi-core architecture for wearable personal health monitoring systems. This multi-core architecture is composed by up to eight cores, several shared multi-banked instruction and data memories, and flexible crossbar interconnects.
2) The core instruction set of our novel architecture has been customized to exploit the specific features of biosignal events, as well as the highly parallel computation opportunities of biosignal processing characteristics.
3) In addition to NTC, the proposed architecture also exploits other advanced low-power features which lead to further energy savings. In particular, the interconnects

This work was partially supported by the the Swiss Confederation under the Nano-Tera.ch NTF Project BioCS-Node. We also acknowledge the support of the Swiss NSF under the project number PP002-119052.

include broadcasting mechanism, enabling coordinated multiple accesses to the shared memories, thus energy savings in the memory hierarchy and in the interconnects. Moreover, the memory hierarchy enables power gating of the unused banks to lower leakage power.

4) Power/performance trade-offs of the proposed architecture are explored for different target workloads. The results show that the proposed multi-core solution achieves 39.5% and 38.8% power savings with respect to the state-of-the-art [9] at high workloads (637 MOps/s) and low workloads (5 kOps/s) respectively, due to the combination of multiple power management options.

The rest of the paper is organized as follows. Section II outlines the main features of biopotentials signal processing and the reference case study. Then, Section III discusses different multi-core architectural choices and details our proposed ULP multi-core processing architecture for biomedical signal analysis. Next, in Section IV we perform a comparative study of the energy consumption versus performance trade-offs entailed by the proposed ULP multi-core design for different realistic wearable biosignals monitoring scenarios. Finally, the conclusions of this work are summarized in Section V.

II. Biopotentials Processing Features

Signal processing on wearable personal health monitoring systems consists mostly of arithmetic computations with relative complexity on single- or multi-input biological signals. Hence, it has been recently shown that they can be optimized to run in real-time on typical embedded low-power microcontrollers. For instance, Rincon et al. [5] showed how delineation of multi-lead ECG signals, using a complex multi-scale wavelet transform algorithm, can be realized on a commercially available personal health monitoring system node with limited computation capability. In fact, multi-lead biological signals analysis are often needed to obtain an accurate view of biological events. However, the analysis of these multi-lead signals entails considerably parallel computation opportunities which can be exploited on multi-core processing platforms [9].

The reference benchmark in this work is a real-time multi-lead ECG processing application which comprises two components: compressed sensing (CS) and Huffmann coding. CS [13] performs a 50% compression on a block of 512 samples of ECG data (sampled at 250 Hz) per lead whereas the Huffmann coding part encodes the compressed data further for wireless transmission. The benchmark operates on 8 leads in parallel (one core per lead) to make the system more accurate and resilient to noise artifacts. The CS part follows always the same program flow independent of the input data, however the Huffmann coding adds a short section of data-dependent program flow.

For a single lead, the benchmark uses a total of 552 bytes for instructions and 16922 bytes for data. This data consist of two parts, namely working data (2586 bytes) and read-only data (14336 bytes). More specifically, the read-only data is comprised of 3 lookup tables (LUTs), i.e, a random vector for CS (12288 bytes) with a linear access pattern and two data dependent LUTs (1024 bytes each) for the Huffmann coding.

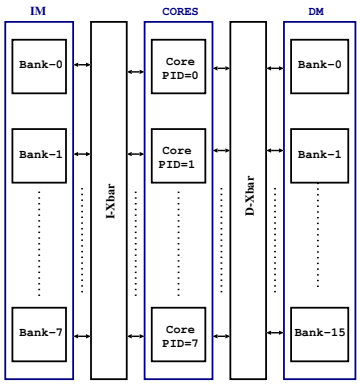

Fig. 1. The Proposed Multi-Core Architecture

III. ULP Multi-Core Processing Architecture

The multi-core architecture proposed in [9], herein referred to as *mc-ref* architecture, is a starting point of our investigation to explore an energy-efficient multi-core architecture for biomedical applications. Similar to the *mc-ref* architecture, our proposed architecture, shown in Fig. 1, involves 8 cores sharing a data memory (DM), divided into 16 memory banks (64 kBytes total) via a central data crossbar (D-Xbar) interconnect. However, as opposed to the *mc-ref* architecture, our proposed architecture offers instruction memory (IM) sharing via a central instruction crossbar interconnect (I-Xbar) similar to the D-Xbar, yet supporting 8 memory banks instead of 16. To this end, the proposed architecture includes memory management units (MMUs) (cf. Fig. 2), enabling each core to access its individual working data with a single instance of a compiled application executed by all the cores. A unique processor identity (PID) for each core is used to place the working data in the DM. In case of memory access conflicts, the requests are served alternately while the waiting cores are stalled using clock gating to avoid unnecessary active power consumption. The following subsections explain in detail the features of the proposed architecture.

A. Low-Power Core (TamaRISC)

The core we have developed for the presented system architecture is a custom-designed reduced instruction set computing (RISC) architecture for biosignal analysis. It is shown in Fig. 2. The core architecture focuses on minimizing the instruction set complexity, while still providing enough hardware support, especially regarding addressing modes, for efficient execution of the target biomedical applications. The processor has a 3-stage pipeline (fetch, decode and execute stages). The core operates on a data word length of 16-bit, comprises 16 working registers and 3 external memory ports (one for instruction read, one for data read, and one for data write, all accessible in the same cycle. The instruction word length is 24-bit, and every instruction has a single-word size. All instructions are executed in one cycle, guaranteed by the complete data bypassing inside the core for registers as well as memory write-back data.

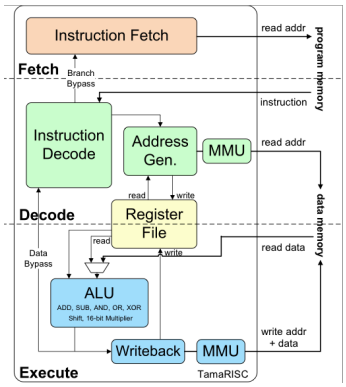

Fig. 2. Custom-Designed Core (TamaRISC) Architecture

The instruction set architecture (ISA) comprises a total of 11 unique instructions, with 8 arithmetic logic unit (ALU), 2 program flow and 1 general data-move instructions. The ALU supports addition, subtraction, shift, logical AND, OR and XOR, as well as full 16-bit by 16-bit multiplications. All ALU instructions work on 3 operands, using the exact same addressing mode options for each instruction, which reduces the complexity of the architecture, since the operand fetch logic and the arithmetic operation are completely decoupled. Additionally, the instruction word encoding is designed as regular (fixed bit positions) and as simple as possible to allow for very efficient decoding of the operands and the different instruction words in general. The supported addressing modes are register direct, register indirect (with pre- or post-increment and decrement), as well as register indirect with offset. Branching is possible in direct and register indirect mode, as well as by an offset with 15 different condition modes (dependent on the processor status flags: carry, zero, negative and overflow).

B. Crossbar Interconnects

The crossbar interconnects, both the D-Xbar and I-Xbar, are a Mesh-of-Trees (MoT) interconnection network to support high-performance communication between processors and memories [12]. The interconnects are intended to connect a number of processing cores (in our case 8 cores) to a multi-banked memory on data (i.e., 16 banks) and instruction sides (i.e., 8 banks). The total memory access latency is one clock cycle, however in case of multiple conflicting requests, for fair access to memory banks, a round-robin scheduler arbitrates access and a higher number of cycles is needed depending on the number of conflicting requests, with no latency in between. To reduce memory access time and increase shared memory throughput, a read broadcast can be used and no extra cycles are needed when such a broadcast occurs.

C. Instruction Memory Organization

As shown in Fig. 3, a significant amount of power (54% of the total power consumption) is consumed by the IM in the *mc-ref* architecture while executing the benchmark. This is due to dedicated IM banks for each core. Biomedical signal processing platforms often execute the same operations on different input data on multiple cores, thus the same instructions are read from the IM banks if the cores are in synchronization.

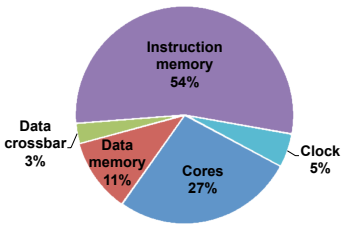

Fig. 3. Power Distributions in the *mc-ref* architecture

Nevertheless, all the IM banks are accessed, thus power would be wasted. However, this power can be reduced by minimizing the number of accesses to the IM banks by reading the identical instructions only once and broadcasting them to all the cores. The IM banks have mostly identical contents, only differ in few instructions due to different memory locations for the working data. However, a single instance of the compiled application can be used for all the cores, provided that the cores can access different working data with the same instruction words. To this end, MMUs translate the same decoded address to different physical memory addresses according to the PID numbers. Using the same compiled application for all the cores facilities IM sharing via the I-Xbar. As opposed to the *mc-ref* architecture, each core can access the entire IM, 96 kBytes in total. However, the instruction broadcasting is beneficial if the cores remain in synchronization. This can be limited due to possible DM conflicts as analyzed in Section IV-C2. Hence to exploit instruction broadcasting, we reorganize the DM and data broadcasting is applied to minimize the conflicts (cf. Section III-D).

Our proposed multi-core architecture enables two different IM organizations: interleaved and banked instructions. The first one, *ulpmc-int*, interleaves instructions across the banks to minimize IM conflicts in the case of synchronization lost between the cores. The second one, *ulpmc-bank*, maps instructions into the minimum number of IM banks. The *ulpmc-bank* is intended to reduce the memory leakage power consumption, by applying power gating to the unused IM banks, which has a significant impact on the overall power consumption at low workloads [9]. The only architectural difference between the *ulpmc-int* and *ulpmc-bank* is the selection bits assignment for the IM banks. The *ulpmc-int* selects the IM banks based on the least significant bits while the *ulpmc-bank* chooses the IM banks according to the most significant bits.

D. Data Memory Organization

The working data sets are individual for each core whereas read-only data can be shared between all cores. Application profiling of the benchmark for DM accesses shows a distribution of 76% private versus 24% shared accesses. Out of the shared accesses, 92% are on the CS random vector while 8% are on the Huffmann coding LUTs. To minimize data access conflicts, the proposed architecture offers two different sections in the DM: shared and private sections. The size of the private and shared sections are configurable and determined during compilation of applications. The working data is separate for each core, thus it is placed in the private section whereas the shared LUTs are linked into the shared section.

The decoded address is used as the physical memory address for a shared section access, whereas the address translation is applied to generate the physical memory addresses for a private section access. The private sections of each core are located into different memory banks, thus the private section is accessed without conflicts. Moreover, shared data (read-only data) is interleaved across the memory banks to minimize conflicts when shared data is accessed. More specifically, the CS random vector accesses are with a linear pattern, thus can be performed conflict free with the data broadcasting, provided that the cores are in synchronization. However, the data dependent Huffmann coding LUTs can produce data conflicts, due to all 8 cores processing different sample data.

IV. Experimental Setup and Results

To explore the power/performance trade-offs between the architectures, we have built the reference (*mc-ref*) and the proposed designs. The designs are implemented in a 90 nm low leakage process technology trading peak performance for significant leakage power reduction, especially in the memories. The reference benchmark is executed on the designs for various workloads while exploiting voltage scaling to accomplish minimum power solutions. The scaling of the operating voltages is limited to the transistor threshold voltage level to avoid performance variability and functional failure issues occurring mainly at sub-threshold voltages. The power values at scaled voltages are calculated regarding the fact that the power decreases with the square of the supply voltage.

A. Power Characterization Framework

The evaluation and implementation flow for the architectures is shown in Fig. 4. The processing core is described in LISA (Language for Instruction Set Architectures) [14], which enables rapid design space exploration for the software as well as hardware aspects of the system. Synopsys Processor Designer (PD) is used to generate the RTL description of the core, a cycle accurate instruction set simulator as well as the necessary software tools (assembler, linker) for creating program binaries from the LISA specification. Additionally, the tool chain is extended by a custom C compiler, which is based on the PD built-in CoSy compiler development system. The C compiler allows for easier benchmark development. The design flow contains a custom regression test for cycle accurate verification of the LISA model simulation against the behavioral simulation of the generated HDL code. The HDL code is integrated into the multi-core architectures written in VHDL, providing the crossbar interconnects and memory banks. The complete system architecture is then synthesized, placed, routed and optimized to have a full layout design. This design is then post-layout simulated using the memory contents extracted from the compiled benchmark program binary. The resulting trace file is finally used to perform an accurate power analysis of the complete system.

B. Design Point Exploration

Fig. 5 and Fig. 6 show the power consumption of the *mc-ref* and the proposed architecture optimized with different clock

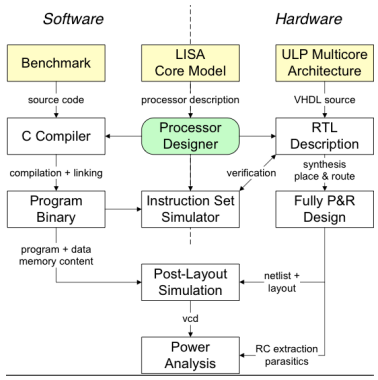

Fig. 4. System Evaluation and Implementation Flow

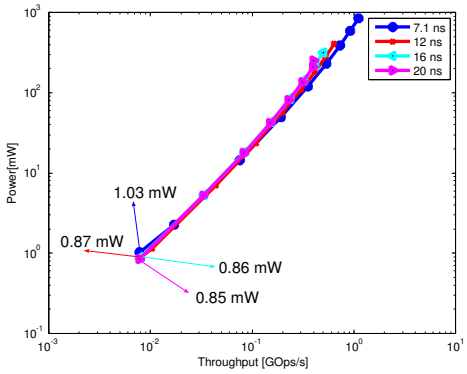

Fig. 5. The *mc-ref* Design: Power Values for Various Clock Constraints

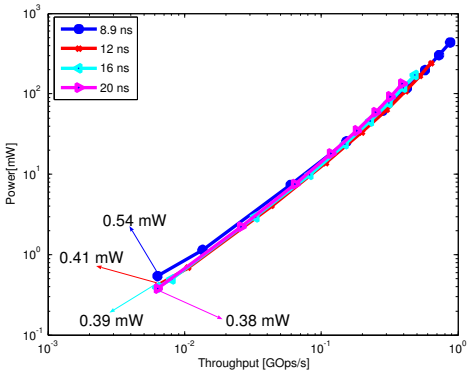

Fig. 6. The Proposed Design: Power Values for Various Clock Constraints

constraints. During this experiment, the designs are supplied by the minimum voltage levels required for the respective throughputs. Both designs operate at around 20 ns when optimized for area. However the *mc-ref* and the proposed architecture operate up to 7.1 ns and 8.9 ns clock periods, respectively when optimized for speed. This difference is due to the I-Xbar, leading almost to 1.8 ns additional delay in the longest delay path of the proposed architecture. This path occurs with the direct branch instruction when the branch address is read from the DM. However, the targeted application groups do not require such high clock frequency, thus the delay due to the I-Xbar does not raise any vital timing issue.

As shown in Fig. 5 and Fig. 6, a 12 ns clock constraint provides an energy efficient design point with high data throughput for both architectures. At this clock constraint, the designs are able to operate at a wide range of throughputs, and consume slightly more energy than the corresponding

slower designs. When the voltages of all the designs reach the threshold, the *mc-ref* and the proposed architecture, optimized for 12 ns achieve 15.5% and 24.1% power savings with respect to their corresponding highest throughput optimized designs and consume only slightly higher power than the area optimized designs. As a consequence of this experiment, we optimized both designs with a 12 ns clock constraint.

TABLE I
AREA RESULTS OF THE ARCHITECTURES (1GE = 3.136 μm^2)

	reference	proposed
	mc-ref	*ulpmc-int / ulpmc-bank*
Total	1108.1	1128.8
Cores	81.5	87.3
IMs	429.4	429.4
DMs	576.7	576.7
D-Xbar	20.5	23.0
I-Xbar	-	12.4

Table I shows the area results of the considered architectures. The logic area in the proposed design increases almost 20% with respect to the *mc-ref* architecture, notably due to the I-Xbar (12.4 kGE) and broadcasting mechanism implemented in the crossbars. However, the area difference between the designs is insignificant, less than 2%, since the memories occupy are largest area, almost 90% of the total area.

C. Experimental Results

1) Energy efficiency of the Core : TamaRISC is more energy efficient than other state-of-the-art cores for biomedical signal processing. It consumes only 15.6 pJ/Ops at 1.0 V. For the same supply voltage level (1.0 V), yet 130 nm process Kwong et al. [15] report 47 pJ/cycle energy consumption for their 16-bit core where the number of clock cycle per instruction is higher than one. In another work, Ickes et al. [16] introduce a 32-bit core implemented in 65 nm, and the energy consumption of the core [16] is estimated for 1.0 V between 19.7 pJ/Ops and 27.0 pJ/Ops. Compared to these state-of-the-art processing cores, our optimized core consumes less energy per operations notably due to its simplified architecture as well as reduced instruction set, as explained in Section III-A.

2) Multi-Core Architectures Comparison: To execute the benchmark the *mc-ref* architecture requires 90.20k clock cycles whereas the *ulpmc-int* and the *ulpmc-bank* versions of the proposed architecture require 90.40k and 101.8k clock cycles, respectively. These differences are mainly due to the shared data access conflicts. The CS random vector with linear pattern is accessed conflict free, because the cores are in synchronization, and thus benefit from the data broadcasting. However, the data dependent Huffman coding LUTs cause memory conflicts since all the cores process different input data. To reduce the conflicts these LUTs are placed into the private section of the DM. In that case, the proposed architecture with *ulpmc-int* version requires almost 90.20k cycles, as the *mc-ref* architecture. However, the *ulpmc-bank* version needs 94.00k clock cycles, only 4% increase with respect to the *mc-ref* architecture. This is due to the data dependent program flow in the Huffman coding which leads the cores to lose the synchronization. Thus, the *ulpmc-bank* version suffer from the IM conflicts due to the banked instruction organization.

TABLE II
DYNAMIC POWER DISTRIBUTIONS AT 8 $MOps/s$ AND 1.2 V

	reference	proposed	
	mc-ref	*ulpmc-int*	*ulpmc-bank*
Total	0.64 mW	0.45 mW	0.38 mW
Cores	0.18 mW	0.25 mW	0.21 mW
IM	0.36 mW	0.05 mW	0.05 mW
DM	0.07 mW	0.06 mW	0.06 mW
D-Xbar	0.02 mW	0.03 mW	0.02 mW
I-Xbar	-	0.03 mW	0.01 mW
Clock Tree	0.03 mW	0.04 mW	0.04 mW
Efficiency	-	29.7%	40.6%

With only the broadcasting mechanism implemented in the I-Xbar, the IM is accessed totally 428740 times in both *ulpmc-int* and *ulpmc-bank* versions while the number of access in the *mc-ref* architecture is 90100 per core, adding up to 720800. Therefore, the number of accesses in the proposed architecture is reduced by 40% with respect to the *mc-ref* architecture. However, as a result of the DM organization together with the broadcasting mechanisms, the cores remain mostly in synchronization, and thus the number of accesses is reduced to 90220 (87% reduction with respect to the *mc-ref* architecture) for both *ulpmc-int* and *ulpmc-bank* versions. This leads to significant power savings on the IM of both versions, as shown in Table II (86% reduction with respect to the one in the *mc-ref* architecture). Additional power costs of the I-Xbar are only 0.03 mW and 0.01 mW in the *ulpmc-int* and the *ulpmc-bank* versions, respectively. Moreover, the broadcasting in the I-Xbar and D-Xbar do not lead to any significant additional power consumption in the proposed architecture. However, the cores in both versions of the proposed architecture consume more power than the ones in the *mc-ref* architecture. This is due to the signal activity increase caused by the I-Xbars. In the *mc-ref* architecture, the cores are directly connected to the IM banks whereas in the proposed architecture, there exist the I-Xbar between them, leading to more signal activities on the instruction paths. However, as a consequence of reduced memory powers the *ulpmc-int* and the *ulpmc-bank* versions accomplish 29.7% and 40.6% active power savings, respectively compared to the *mc-ref* architecture for the same workloads. The *ulpmc-bank* version achieves higher active power saving due to less signal activity on the instruction paths than the *ulpmc-int* version. As seen from the third column of the table, the cores and I-Xbar consume less power in the *ulpmc-bank* version than the ones in the *ulpmc-int* version, because the instructions are read only from one IM bank instead of multiple banks. This leads to less signal activities at the output nets of the I-Xbar and, thus less power consumptions.

At nominal voltage (1.2 V), the *mc-ref* architecture achieves 664.5 MOps/s while the *ulpmc-int* and *ulpmc-bank* operate up to 662.3 MOps/s and 636.9 MOps/s, respectively. When the supply voltages reach the threshold level, the *mc-ref*, *ulpmc-int* and *ulpmc-bank* architectures still accomplish around 10 MOps/s. Fig. 7 shows power consumptions of the architectures

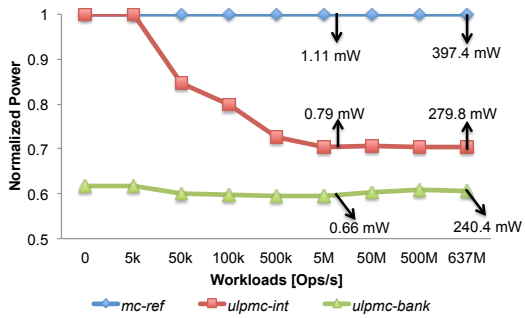

Fig. 7. Normalized Power Consumptions at Various Workloads

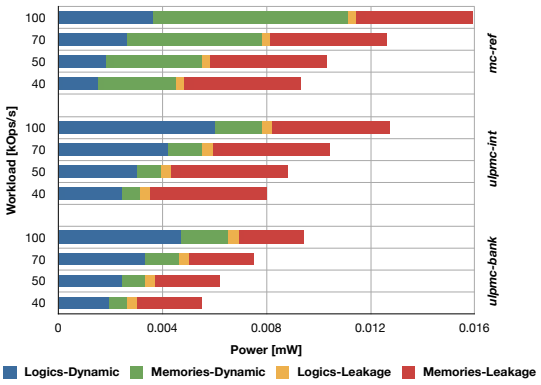

Fig. 8. Dynamic vs Leakage Power Consumptions for Various Workloads

normalized to *mc-ref* design power consumption for various workloads. During this experiment, both voltage and frequency scaling are applied for workloads higher than 10 MOps/s, however for workloads lower than this, only frequency scaling is used and the supply voltages are kept at the minimum level. As seen from this figure, the *ulpmc-bank* design is more energy efficient than the *ulpmc-int* and *mc-ref* designs for a given workload requirement. More specifically, at the highest workload (636.9 MOps/s) that all the designs can achieve, the *mc-ref* architecture consumes around 397.4 mW, whereas the *ulpmc-int* and the *ulpmc-bank* designs consume 279.8 mW and 240.4 mW, respectively. Thus the *ulpmc-int* achieves around 29.6% while the *ulpmc-bank* accomplishes 39.5% power savings with respect to the *mc-ref* architecture. As the workload requirement becomes low, around 10 MOps/s, the *mc-ref* architecture consumes 1.11 mW whereas the *ulpmc-int* and the *ulpmc-bank* consume 0.79 mW and 0.66 mW, respectively. Therefore, the *ulpmc-bank* accomplishes 40.5% power saving with respect to the *mc-ref* architecture.

Finally, Fig. 8 shows the dynamic and the leakage power consumptions of the circuits logics and memories (both instruction and data memories) for the workloads lighter than 100 kOps/s. As shown in the figures, the *mc-ref* and the *ulpmc-int* designs leak almost the same amount of power, whereas the *ulpmc-bank* design has 38.8% less leakage power consumption than the *mc-ref* design. This is due to the power gating on the unused IM banks. The leakage power consumption of the architectures become comparable with their dynamic power consumptions at around 50 kOps/s workload. As a result, even though the *ulpmc-int* design is more efficient than the *mc-ref* design in terms of dynamic power dissipation, the *ulpmc-int* architecture falters for the low workloads regarding the total power consumption. Notably, as seen in Fig. 7, the power consumption of the *ulpmc-int* becomes almost equal with the *mc-ref*'s around 5 kOps/s. However, the *ulpmc-bank* architecture maintains its efficiency, 38.8% power saving for the same workload whereby the architectures almost only leak.

V. Conclusion

In this paper, we have explored power/performance trade-offs between different multi-core architectures for wearable health monitoring systems which exploit existing highly parallel computation opportunities in bio-signal analysis as well as near threshold computing to extend the lifetime of the health monitoring systems. We have showed that an energy-efficient multi-core system is achieved by instruction and data memory sharing together with the broadcasting mechanisms. To exploit the broadcasting mechanism, the cores require to be in synchronization. To this end, the data memory is divided into shared and private sections where read-only and working data are mapped into, respectively. Unused instruction memory banks are power gated to reduce the leakage power consumption. Our results show that the proposed architecture achieves 39.5% and 38.8% power savings with respect to the prior art at high (637 MOps/s) and low (5 kOps/s, whereby the circuits almost only leak) workload requirements, respectively.

References

[1] World Health Organization, *Cardiovascular diseases*, 2009. [On-line]. Available: http://www.who.int/topics/cardiovascular diseases/.

[2] Toumaz Technology, 2009. [On-line]. Available: http://www.toumaz.com/public/news.php?id=92.

[3] Corventis, 2009. [On-line]. Available: http://www.corventis.com/AP/nuvant.asp.

[4] Yazicioglu R. F., et al.: "Ultra-low-power wearable biopotential sensor nodes," Proc. of IEEE EMBC, 2009.

[5] Rincon, F.; et al., "Development and Evaluation of Multilead Wavelet-Based ECG Delineation Algorithms for Embedded Wireless Sensor Nodes," IEEE Transactions on Information Technology in Biomedicine, 15, 854–863, 2011.

[6] Hanson, S.; et al., "Exploring Variability and Performance in a Sub-200-mV Processor," IEEE Journal of Solid-State Circuits, 43, 881–891, 2008.

[7] Zhai, B.; et al., "A 2.60 pJ/Inst subthreshold sensor processor for optimal energy efficiency," Symposium on VLSI Circuits, Honolulu, 2006.

[8] Dreslinski, R. G.; et al., "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits," Proc. of the IEEE, 98, 253–266, 2010.

[9] Dogan, A.; et al., "Power/Performance Exploration of Single-core and Multi-core Processor Approaches for Biomedical Signal Processing," Proc. of PATMOS , 2011.

[10] Dreslinkski, R. G.; et al., "An Energy Efficient Parallel Architecture Using Near Threshold Operation," 16th International Conference on Parallel Architecture and Compilation Techniques, 2007.

[11] Yu P.; et al., "An Ultra-Low-Energy Multi-Standard JPEG Co-Processor in 65 nm CMOS With Sub/Near Threshold Supply Voltage," IEEE J. Solid-State Circuits, 45, 668–680, 2010.

[12] Rahimi, A.; et al., "A fully-synthesizable single-cycle interconnection network for Shared-L1 processor clusters," Proc. of DATE, 2011.

[13] Mamaghanian, H.; et al., "Compressed Sensing for Real-Time Energy-Efficient ECG Compression on Wireless Body Sensor Nodes," IEEE Transactions on Biomedical Engineering, 12, 120–129, 2011.

[14] Synopsys. Datasheet Available: http://www.synopsys.com/Systems/Block-Design/processorDev/Pages/default.aspx.

[15] Kwong, J., et al., "An Energy-Efficient Biomedical Signal Processing Platform," IEEE J. Solid-State Circuits, 46, 1742–1753, 2011.

[16] Ickes, N., et al., "A 10 pJ/cycle ultra-low-voltage 32-bit microprocessor system-on-chip," Proc. of ESSCIRC, 2011.