

Performance Analysis and Design Space Exploration for High-End Biomedical Applications: Challenges and Solutions

Iyad Al Khatib
IMIT, ICT, KTH
Royal Institute of
Technology, Sweden
+4687904111
iyad@imit.kth.se

Davide Bertozzi
ENDIF
University of Ferrara
Ferrara, Italy
+390532974832
dbertozzi@ing.unife.it

Axel Jantsch
IMIT, ICT, KTH
Royal Institute of
Technology, Sweden
+4687904124
axel@imit.kth.se

Luca Benini
DEIS
University of Bologna,
Bologna, Italy
+390512093782
lbenini@deis.unibo.it

ABSTRACT

High-end biomedical applications are a good target for specific-purpose system-on-chip (SoC) implementations. Human heart electrocardiogram (ECG) real-time monitoring and analysis is an immediate example with a large potential market. Today, the lack of scalable hardware platforms limits real-time analysis capabilities of most portable ECG analyzers, and prevents the upgrade of analysis algorithms for better accuracy. Multiprocessor system-on-chip (MPSoC) technology, which is becoming main-stream in the domain of high-performance microprocessors, is becoming attractive even for power-constrained portable applications, due to the capability to provide scalable computation horsepower at an affordable power cost. This paper illustrates one of the first comprehensive HW/SW exploration frameworks to fully exploit MPSoC technology to improve the quality of real-time ECG analysis.

Categories and Subject Descriptors

C.3 [Special-Purpose and Application-Based Systems]: Microprocessor/microcomputer applications, Real-time and embedded systems

General Terms

Performance, Design, Experimentation.

Keywords

Multiprocessor System-on-Chip, biomedical, electrocardiogram algorithms, real-time analysis, design space exploration.

1. INTRODUCTION

Heart disorders remain by far the leading cause of death in the world for both women and men of all ethnic backgrounds [1]. ECG monitoring during normal activity using Holter devices has

become a standard procedure for the detection of diseases such as cardiac arrhythmias [2].

Unfortunately, limited processing power and tight power budgets of Holter devices have traditionally limited their functionality to data acquisition and recording of full ECG traces or abnormal events. Record analysis and diagnosis are performed offline at the medical center. Remote real-time ECG monitoring through dedicated telemedicine links involve transmission of a huge amount of life-critical data and a 100% functional always-on connection.

The recent advances in silicon integration technology are paving the way for real-time ECG monitoring and analysis, thus allowing to promptly react to life-threatening heart malfunctions and to relax the requirements on telemedicine links. In particular, boosting clock frequencies of monolithic single-core microprocessors has clearly reached a point of diminishing returns and in the next few years performance gains will mainly come from an increase in the number of processor cores per chip. Although evidence of this trend is unmistakable in the domain of high-end microprocessors [3], the need to optimize performance per watt is likely to lead even portable devices to deploy multiple processor cores operating in parallel at lower clock speeds. Scalable chip multiprocessing is therefore emerging as a major design paradigm shift to provide scalable computation horsepower in a power efficient fashion. On one hand, this trend is accelerated by the stringent demands posed by increasingly complex software applications, but in turn will force to radically revise programming models.

Heart activity monitoring and analysis provide a promising application domain for on-chip multiprocessor systems. It is electrically recorded as a set of ECG signals which can reveal a number of heart malfunctions [4]. The most reliable ECG analysis technique is the 12-lead ECG, which requires processing twelve different signals sensed from the patient's body and which renders a 3D view of the heartbeat. The migration towards 12-lead ECG is facing computational challenges, especially in portable devices, where the need to meet tight computation and power budgets results in processing fewer leads. This is a limitation to the development of more sophisticated analysis algorithms, which are needed to overcome the concerns posed by heartbeat analysis: physiological variability of QRS complexes, base-line wander, muscle noise, artifacts due to electrode motion and patient mobility.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CODES+ISSS'07, September 30–October 3, 2007, Salzburg, Austria.
Copyright 2007 ACM 978-1-59593-824-4/07/0009...\$5.00.

Multi-Processor System-on-Chip technology provides a disruptive means of overcoming the aforementioned challenges. It consists of an energy-efficient and scalable hardware/software (HW/SW) platform which can meet computation demands at an affordable power cost. In this paper, we introduce a novel MPSoC architecture for ECG analysis which improves upon state-of-the-art mostly for its capability to perform 12-lead real-time analyses of input data with high sampling frequencies, leveraging the computation horsepower and the functional flexibility provided by many concurrent VLIW DSPs. This biomedical chip builds upon some of the most advanced industrial components for MPSoC design (multi-issue VLIW DSPs, system interconnect from STMicroelectronics, and commercial off-the-shelf biomedical sensors), which have been composed in a scalable and flexible platform. Therefore, we have ensured its reusability for future generations of ECG analysis algorithms and its suitability for porting of other biomedical applications, in particular those collecting input data from wired/wireless sensor networks.

The specific application domain addressed in this work calls for specific design methodologies for which full system modeling accuracy is critical, since the resulting architecture needs to be highly predictable. Uncontrolled run-time fluctuations of the system might lead to incorrect detection of life-threatening events in the heartbeat traces. For this purpose, the most daunting challenge for system designers perhaps consists of accurately predicting the impact of the communication and I/O architectures on system performance. Even more critical is the ability to capture the interaction between these two sub-systems.

Finally, our architecture allows to compare existing analysis algorithms and to develop new ones, while keeping low-level implications of software decisions under control through the support of a virtual platform. In this paper, we illustrate the pros and cons of deploying more complex analysis techniques than the traditional Pan-Tompkins algorithm [5], from a comprehensive hardware and software viewpoint.

2. BACKGROUND

The biomedical application we choose for our design is the ECG, which is a diagnosis tool used by medical doctors and nurses to check the status of the heart. For the purpose of designing an algorithm and code (hence SW), understanding ECG signals is firstly done. Each ECG signal is an electrical recording of the heart activity taken from many sensors connected on the patient's body. Each ECG signal comes from a connection that is referred to- in the biomedical field- as a "lead". The more the leads the larger the information set of data we can get for the heart activities. One of the latest techniques for monitoring heart activity is the 12-lead ECG, which relies on nine sensors placed on the patient's body (Figure 1-a). Before the 12 lead technique, physicians used only three sensors (RA, LA, and LL as shown in Figure 1-a) in a method known as the 3-lead ECG, which suffers from the lack of information about the whole of the heart. The 12-lead ECG technique, offers a view of the heart in its three dimensional form and can thus detect many more abnormalities. The cost for this increased amount of information is higher number of computations, more sophisticated monitoring and analysis of large data sets, and stringent requirements on the underlying portable hardware platform.

Figure 1-b shows a typical ECG signal for a healthy heart. The important peaks on the ECG signal are : P, Q, R, S, and T, each of

which refers to some heart activity. Figure 1-c shows real recorded signals from 12-leads. Many healthcare centers, till now, still print these signals on an eyeballing paper; therefore, they may be confusing for the doctor's eyes. On the other hand, when using digital recording and filtering we can determine the peaks more accurately especially that digital computing becomes applicable.

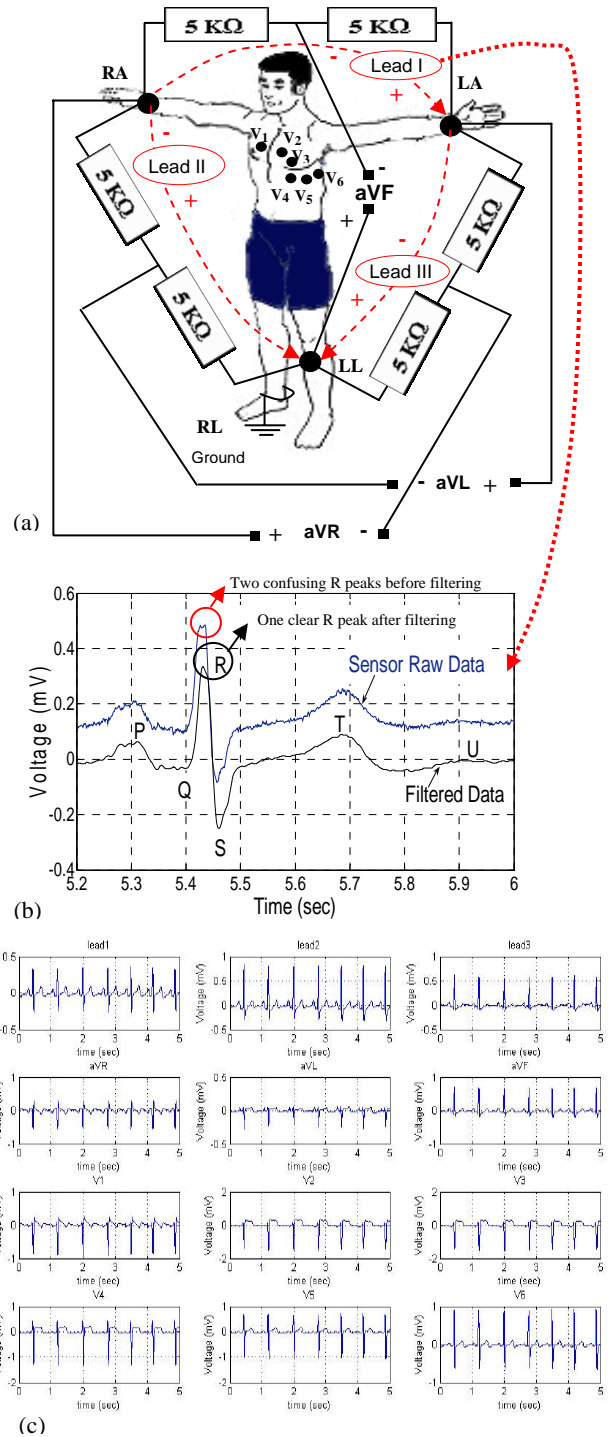


Figure 1. 12 lead ECG: (a) Sensors on a human body. (b) The QRS complex. (c) Complete 12 lead readout.

- **Computation and communication parallelism.** Storage of processing results and the next computation round should be handled in parallel for efficient execution. Therefore, the deployment of DMA engines is highly attractive in this case. However, two conflicting requirements should be taken into account. On one hand, storage data size needs to be kept to the minimum to extend the observation window (given the finite memory size), while on the other hand sufficiently long data transfers are required to amortize the cost of DMA transfer programming.

In a step further to implement the HW/SW co-design based on the above analysis, we are pushed to found a guiding (flow of logical steps) methodology that we refer to in order to keep on optimizing the life-critical *biomedical algorithm, SW* and *HW*. This type of methodology was induced throughout the work of the design and implementation, which we present in a flow of steps and their inter-relations for HW/SW co-design for high-end biomedical applications. This methodology is summarized in the flowchart presented in Figure 2, where we divide the work into three main layers: *biomedical layer, SW layer*, and *HW layer*. The starting point is always the right choice of an application (problem) that has medical significance (heart diagnosis via ECG analysis in our case). In the first layer, the biomedical specifications for the problem must be set pedantically in order to design an algorithm that suits the *life-critical application* as discussed in the first point of the aforementioned analysis of this section. At this stage and at every stage, we need to keep in mind and thus in the design flow the relationship with and the effects on other layers and stages. Hence, we find a major need to look at whether the biomedical problem can be partitioned (e.g. period analysis and peak analysis for ECG signals). This defines one sort of parallelism that is reflected in the SW code in a later stage. Then the SW design requires defining parameters needed to respect in the analysis (e.g. real-time limits in seconds in our case and also power consumption as the device should be wearable and self powered). After defining these specs, a look at the algorithm from a SW parallelization viewpoint is essential since it shall affect both time and power consumption spec results in the simulation stage further in the HW layer. Once the parallelism is defined, the parallelized code is generated. A note worth mentioning is that it is possible to parallelize some loops without the need to parallelise the biomedical algorithm itself to minimize time-consumption. At this stage we have a block of functional code but not yet optimized to best suit the HW computation and communication requirements. This code and its dependencies (e.g. the data chunks and their specs as number of samples and length of recording) inspire the HW designer to define the HW needs for accurate modelling. The result is an un-optimized HW with interface mechanisms that is ready to load the SW to go further to the HW/SW co-simulation stage. Since this is a life-critical application, it calls upon cycle and signal accurate co-simulation. The results of the co-simulation are the major outcome to test and compare with specs in above stages in higher layers. For a life-critical application, like real-time ECG analysis, a step back to compare with the high layer (biomedical) standards and specs is a must. This may affect the optimization of the algorithm, SW, and HW, respectively. When the HW/SW co-simulation results are pedantically investigated and accepted, we get the optimised HW and its relative SW in a stage guiding the *optimized physical solution*.

Based on all the above analyses, we came up with a virtual platform with mixed modelling abstractions so to trade-off accuracy with simulation time. The MPSIM modelling and simulation environment [11] was chosen as the starting point for an intensive extension and customization effort for the target application domain and to meet its requirements.

We chose SystemC as the reference description language and simulation engine, due to its ability to perform efficient HW/SW co-design. The entire on-chip HW/SW architecture was modelled with clock-cycle accuracy extended to the chip boundaries. While retaining this level of accuracy, processor cores were modelled at the level of their instruction set, while only the on-chip bus was modelled with signal accuracy due to its criticality. As an effect, we were able to achieve simulation speeds up to 200 Kcycles/sec on a P4@3.5GHz.

The virtual platform was augmented to support non-functional metrics, such as power dissipation and breakdown. Industry-provided and technology-homogenous 0.13 μ m power models of all system components were deployed for power analysis [12][13].

4. MPSOC ARCHITECTURE

In order to process filtered ECG data in real-time, we choose to deploy a parallel MPSoC architecture. The key point of these systems is to break up functions into parallel operations, thus speeding up execution and allowing individual cores to run at a lower frequency with respect to traditional monolithic processor cores. Technology today allows the integration of tens of cores onto the same silicon die, and we therefore designed a parallel system with up to 13 masters and 16 slaves Figure 3. Since we are targeting a platform of practical interest, we choose advanced industrial components [12]. The processing elements are multi-issue VLIW DSP cores from STMicroelectronics, featuring 32kB instruction and data caches and with maximum clock speed of 400MHz. These cores have 4 execution units and rely on a highly optimized cross-compiler in order to exploit the parallelism. They try to combine the flexibility of programmable cores and the computation efficiency of DSP cores. This way, the platform can be used for applications other than the 12-lead ECG, so to make it cost-effective. Each processor core has its own private memory (up to 512KB), which is accessible through the bus, and can access an on-chip shared memory (8KB are enough for our target application) for storing computation results, prior to swapping to the off-chip memory. Other relevant slave components are a semaphore slave, implementing the test-and-set operation in hardware and used for synchronization purposes by the processors or for accessing critical sections, and an interrupt slave, which distributes interrupt signals to the processors. Interrupts to a certain processor are generated by writing to a specific location mapped to this slave core. The STBus interconnect from STMicroelectronics [14] was instantiated as the system communication backbone. STBus can be instantiated as both: shared-bus or crossbar (partial or full), thus allowing efficient interconnect design and providing flexible support for design space exploration and for platform scalability. In our first implementation, we target a shared bus to reduce system complexity (Figure 3) and assess whether application requirements can already be met or not with this configuration. We then explore also a crossbar-based system (Figure 4).

The inherent increased parallelism exposed by a crossbar topology allows decreasing contention on shared communication resources, thus reducing overall execution time. In our implementation, only the instantiation of a 3x6 crossbar was interesting for the experiments.

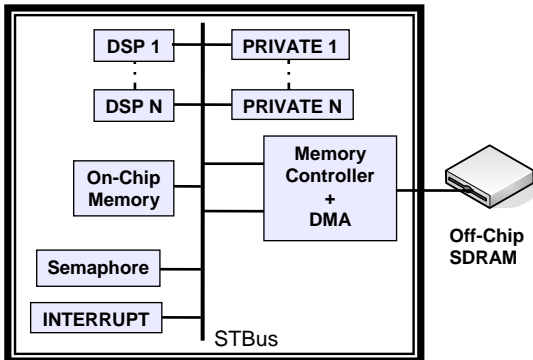


Figure 3. Single bus architecture with STBus interconnect.

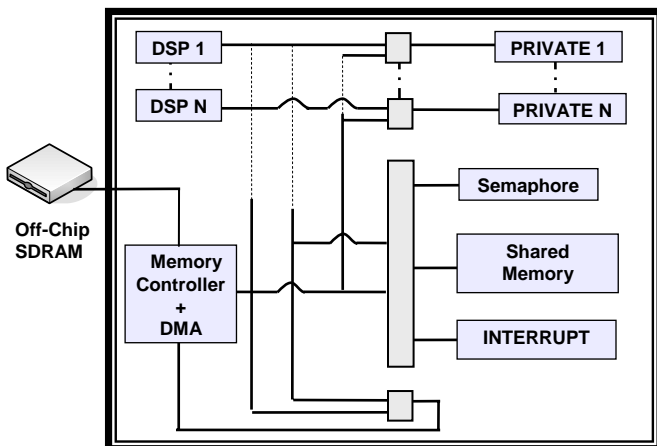


Figure 4. Crossbar architecture with STBus interconnect .

We put a private memory on each branch of the crossbar, which can be accessed by the associated processor core or by a DMA engine for off-chip to on-chip data transfers. Finally, we have a critical component for system performance which is the memory controller. It allows efficient access to the external 64MB SDRAM off-chip memory. A DMA engine is embedded in the memory controller tile, featuring multiple programming channels. The controller tile has two ports on the system interconnect, one slave port for control and one master port for data transfers. The overall controller is optimized to perform long DMA-driven data transfers and can reach the maximum speed of 600MB/s. Embedding the DMA engine in the controller has the additional benefit of minimizing overall bus traffic with respect to traditional standalone solutions. Our implementation is particularly suitable for I/O intensive applications such as the one we are targeting in this work.

Each processor core programs the DMA engine to periodically transfer input data chunks onto their private on-chip memories. Moved data corresponds to 5 seconds of data acquisition at the sensors: 10kB at 1000Hz sampling frequency, transferred on average in 319279 clock cycles (DMA programming plus actual data transfer) on a shared bus with 12 processors. The consumed

bus bandwidth is about 6Mbyte/sec, which is negligible for an STBus interconnect, whose maximum theoretical bandwidth with 1 wait state memories exceeds 400Mbyte/sec. Then each processor performs computation independently, and accesses its own private memory for cache line refills. Different solutions can be explored, such as processing more leads onto the same processor, thus impacting the final execution time. Output data, amounting to 64 byte, are written to the on-chip shared memory, but their contribution to the consumed bus bandwidth is negligible. In principle, when the shared memory is filled beyond a certain level, its content can be swapped by the DMA engine to the off-chip SDRAM, where the records of 8 hours of computation can be stored.

In the above description, we have reported the worst case system configurations. In fact, fewer cores can be easily instantiated if needed but this is a first un-optimized HW design as discussed in Section 3. This architectural template is very scalable and allows for further future increase in the number of processors. This will allow to run in real time even more accurate ECG analyses for the highest sampling frequency available in sensors.

5. SOFTWARE ARCHITECTURE

The software architecture is fully distributed and an independent RTEMS operating system [15] runs on top of each processor core. Kernel images reside in the private memory of each core. A middleware support for inter-processor communication was ported to the system, and a special purpose library was used for DMA programming [16]. Next, we will devote our attention the algorithm for ECG monitoring and analysis.

5.1 Algorithms/ Software Description

A few algorithms have been developed for detection of heart abnormalities. Most of the solutions available try to locate the QRS interval (see Figure 1-b) in order to estimate the heart period by calculating the distance of two consecutive R peaks. In spite of their lightweight complexity, such algorithms do not provide enough confidence in analyzing highly irregular heartbeats, associated with specific patients and/or arrhythmias. Moreover, even when they work fine in defining the QRS interval, the other peaks (which represent other heart activities) will still be obscure for the doctors/nurses looking at the algorithm results.

At the biomedical layer, a choice for the algorithm is a basic block for the success. We start looking at the most widely used algorithm for heart beat detection in healthcare centers is the Pan Tompkins algorithm [5]. The Pan Tompkins solution is built to detect the QRS interval only, and its low complexity makes it suitable for porting on a large number of low-end portable devices. The disadvantages are of course many, e.g. lack of 100% success, confusing peaks and thus diseases, lack of full analysis of all heart activities, and oftentimes a lack of sufficient informative content provided to medical doctors.

In a step to overcome the limitations of nowadays and long-lived solutions, and relying on the performance accelerations guaranteed by MPSoC platforms, we came up with more computation demanding analysis (biomedical) algorithms than the traditional QRS detection technique, and pointed out their pros and cons from a comprehensive hardware-software implementation viewpoint. In what follows we refer to our two biomedical algorithms as the: ACF-based and FFT-based solutions. An overview of the three algorithms follows.

5.2 Pan Tompkins Algorithm

The Pan-Tompkins algorithm detects only the QRS complex of the ECG signal via three detection steps: linear digital filtering, non-linear transformations, and decision rule algorithms. The technique consists of an analog filter, a band-pass filtering stage, derivative, squaring, and windowing stage. The algorithm was originally made to run on a Z80 (Zilog) or an NSC800 (National Semiconductor) microprocessor. Hence, it runs relatively fast in today's platforms. However, it may fail in giving correct results. The processing is done in integer arithmetic so that the algorithm operates in real-time [5] without consuming significant power. The algorithm detects QRS complexes using slope, amplitude and width information.

First, an analog filter is used to band limit the ECG signal at 50Hz. Then, an A/D converter samples the signal at $F_s=200\text{Hz}$. Since only the Q, R, and S peaks and waves are needed in the Pan Tompkins algorithm, then there is no harm in filtering the P and T waves. Therefore, after the A/D conversion, the signal passes through band-pass filter and high-pass filter stages to remove high-frequency noise, P-waves, T-waves, and other artifacts. The resultant signal is then passed through a *local peak detection* algorithm which identifies and marks all the peaks found in the signal. This algorithm uses a set of thresholds in order to be able to select candidate QRS complexes. These thresholds are adaptive and thus are modified based on the amplitude of the new peak found. However, the first thresholds and the relations between them are defined by the SW implementer, i.e. suffer from human subjectivity and depend on the SW developer's experience in QRS complexes, which usually is not very thorough. The filtered signal is then sent to the non-linear transformation stage, where the derivative of the signal is calculated. The derivative contains information about the slope of the QRS. The squaring process intensifies the slope of the frequency response of the differentiated signal to help detect false peaks like the T-waves. A moving window integrator obtains information about the width of the QRS complex. This result is then passed through the same local peak detection and threshold setting algorithms as the original band-pass filtered signal to identify QRS slope information. All the candidate QRS peaks found in both filtered and transformed waveforms are then compared. Only those appearing in both processed waveforms are classified to be valid QRS complexes. The output is a stream of pulses indicating the locations of the QRS complexes. Such an algorithm not only relies on slope information, but also on the amplitude and width information of the QRS complex. In the Pan Tompkins algorithm, the human factor plays a role in the choice of thresholds and the relations therein.

5.3 Analysis via the ACF-Based Algorithm

In principle, traditional ECG analysis starts from a reference point in the heart cycle (the R-peak is commonly used as the reference point). As a consequence, accurate detection of the R-peak of the QRS complex is a prerequisite for the reliable functionality of ECG analyzers [12]. However, as an effect of ECG signal high variability, R-peak detection might be inaccurate. For instance, in the R on T phenomena, a T peak may be wrongly taken for an R peak, and then the R-T interval will be considered as an R-R interval, and the period will be wrong. Hence, other QRS parameters will be consequently inaccurate. As a result, traditional techniques may fail in detecting some serious heart disorders such as the R-on-T phenomenon (associated with

premature ventricular complexes) [17]. Our approach takes a different perspective: instead of looking for the R-peaks and then detecting the period, we detect the period first (via autocorrelation) and then look for the peaks. We use an autocorrelation function to calculate the heartbeat period without looking for peaks. Then, we can restrict our analysis to a time window equal to the period and detect all peaks. Although potentially more accurate, our algorithm incurs a higher computational complexity: 3.5 million multiplications, which have been reduced to 1.75 million through a number of code (SW-layer) optimizations.

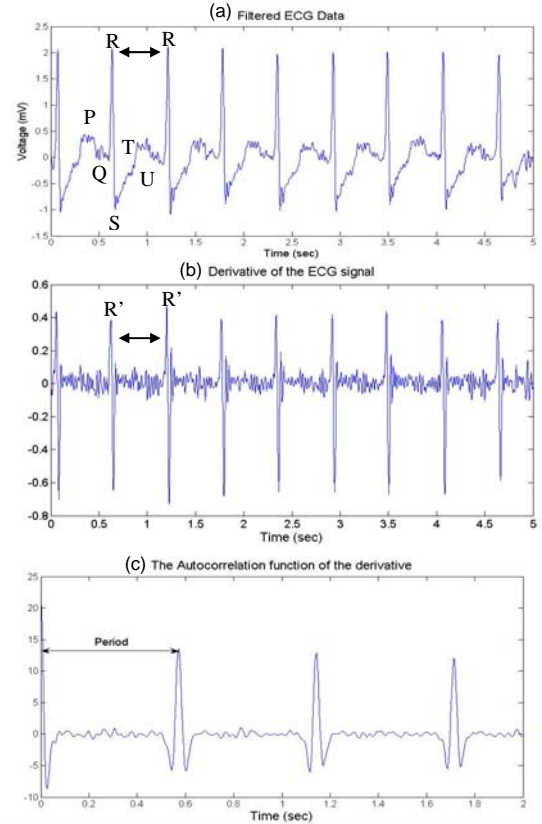


Figure 5. ACF-based algorithm for Heart period analysis.

The autocorrelation we use as shown in (1) has a certain number of Lags (L) to minimize the computation for our specific application as discussed below.

$$R_y[k] = \sum_{n=-\infty}^{n=\infty} y[n] \times y[n - k] \quad (1)$$

where, R_y is the autocorrelation function, y is the filtered signal under study, n is the index of the signal y , and k is the number of lags of the autocorrelation (L has an effect on the performance due to the high number of multiplications). We run the experiments for $n = 1250, 5000$ and $50,000$ relative to the sampling frequencies of $250, 1000,$ and $10,000\text{Hz}$, respectively.

In order to minimize errors and execution time we use the derivative of the ECG filtered signal since if a function is periodic then its derivative is periodic. Hence, the autocorrelation function of the derivative can give the period as shown in Figure 5. The

advantage of taking the derivative, and thus adding some overhead to the code, is that the fluctuations taking place in the signal and especially those around the peaks would be reduced to a near-zero-value. In order to be able to analyze ECG data in real-time and to be reactive in transmitting alarm signals to healthcare centers (in less than 1 minute), a minimum amount of acquired data has to be processed at a time without losing the validity of the results. For the heart beat period, we need at least 4 seconds of ECG data in order for the ACF to give correct results [18].

Our proposed ECG-analysis algorithm was conceived to be parallel and hence scalable from the ground up, so to be able to extend it to any number of leads for analysis for future generations of biomedical devices.

5.4 Analysis via the FFT-Based Algorithm

At this step, we are still investigating algorithm optimization before going to the HW design. Looking for a less demanding analysis method from a computation viewpoint, while still retaining higher accuracy than the Pan Tompkins, we revert to the FFT algorithm to minimize the number of computations needed to estimate the autocorrelation coefficients. In our implementation of the FFT algorithm, data are read from the ECG signal with a specified sampling frequency into a buffer with a limited buffer capacity of 4 seconds. Then, we differentiate the resultant signal so as to enhance its shape. Only the first quarter of the differentiated samples will be used as an input into the FFT block. This technique allows us to save time and unnecessary calculations while still getting to similar results. The differentiated data need to be reordered so that they are correctly used as an input to the FFT block. We perform decimal to binary conversion of the indices of the elements in this array, then reverse the bits of the binary representation of those indices, then perform binary to decimal conversion of the result to get the reordered indices. By correctly applying the butterfly method of the FFT and using the twiddle factors at every stage, FFT data are calculated. The algorithm uses two 'For' loops, the outer loop loops $\log(N/4)$ times (which represents the number of stages in the butterfly diagram), where N is the number of samples increased to the nearest power-of-2 value. The other loop loops $N/4$ samples, since at every stage; $N/4$ FFT values are calculated. We finally get the FFT data in the last stage. Although the FFT-based algorithm results are faster to compute, we observe a loss in the quality of the output autocorrelation plot in the end, i.e. the autocorrelation function coefficients (which is the final plot in both FFT-based and ACF-based algorithms) are not as accurate as the ones described in our ACF-based Algorithm, hence we win on computations and lose on accuracy.

5.5 MPSoC Pros for ECG Analysis

The distinctive features of the above algorithms with respect to the traditional Pan Tompkins algorithm will be now highlighted. We aim at proving user-perceived practical advantages of deploying more computation-demanding algorithms, and therefore the advantages that MPSoC technology can bring to the ECG domain in view of its scalable and energy efficient computation horsepower. A detailed comparison between analysis algorithms is beyond the scope of this paper.

The biomedical advantages that both our algorithms (ACF-based and FFT-based) have with respect to traditional solutions are mainly: (1) eliminating failure in calculating the heart period

since we use a time based autocorrelation instead of using human-chosen thresholds, (2) ability to learn about all the heart activities since even if the Pan Tompkins gave a good detection, it already filters out the P and T waves, (3) ability to detect more diseases, (4) increased accuracy by increased sampling frequency, and (5) better scalability and parallelism.

Hence, both the ACF-based algorithm and the FFT-based algorithm share the aforementioned advantages over the Pan Tompkins; however, we also can look at the medical implications when comparing the ACF-based algorithm with the FFT-based algorithm. In this respect, the FFT-based algorithm gives a faster result than the ACF-based algorithm, but it may not be as accurate, since the peak of the autocorrelation coefficients is surrounded by many similar peaks which are shown in Figure 6 and Figure 7.

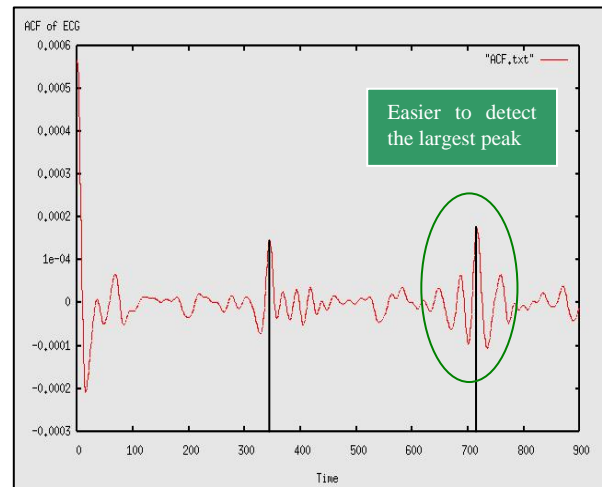


Figure 6. Period detection using the ACF-based direct method to calculate the autocorrelation coefficients for a patient.

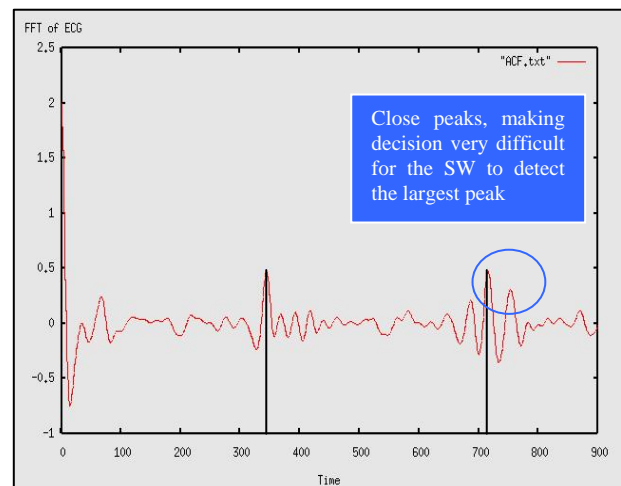


Figure 7. Period detection using FFT-based algorithm to calculate the autocorrelation coefficients for the same patient in Figure 6.

Performing this same operation on different cases, we deduced that the peaks resulting from the FFT method yielded same results (same period) as with the direct method to calculate the ACF but we had a little difference between the data resulting from the two methods; the second peaks for the FFT-based method (Figure 7) were lower in amplitude relative to the maximum at the origin than those for the direct method (Figure 6), which will require more difficult thresholds choice for the FFT case.

From a computation viewpoint, both the ACF direct method and the FFT method are clearly more demanding with respect to the Pan-Tompkins. The relative decrease of complexity that the FFT method is able to provide in the computation for our specific algorithm and SW design (in SW layer, i.e. dependencies and specs) of the autocorrelation function is well documented in Figure 8, where the effect of input data (N) on the number of algorithm multiplications is illustrated.

5.6 Summing Up

The biomedical algorithm and SW analysis carried out so far proves the improvements to abnormalities detection that heartbeat analyzers can enjoy, provided that new platforms leveraging MPSoC technology can be deployed. Our work is a first step in the direction of a full exploitation of MPSoC technology for this purpose, covering design methodologies and HW/SW architectures. In order to better highlight the potential performance achievable with our architecture and design flow, we coded and mapped the most computation-demanding ACF-based algorithm on the MPSoC platform. With respect to this algorithm, we illustrate HW/SW design space exploration in Section 6, trying to move from algorithm-specific insights to general purpose indications for platform tuning.

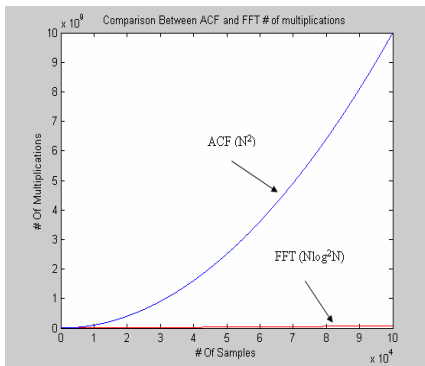


Figure 8. Number of multiplications for the direct ACF method and the FFT method for our biomedical algorithm.

6. HW/SW DESIGN SPACE EXPLORATION

The first analysis was done to profile the execution of the code on the SW layer and to determine the best coding solution in terms of energy, execution time, and precision (in the HW layer). Furthermore, we have explored the design space searching for the best platform configuration for the 12-lead ECG data analysis. Alternative system configurations have been devised for different levels of residual battery lifetime, trading off power with accuracy.

6.1 Processor Cores

Our platform tuning effort was structured in three steps. First, we determined the best coding solution in terms of energy, execution time and precision. Second, we looked for the most efficient hardware platform configuration for the application at hand. Third, we analyzed the scalability of the platform to support future applications featuring a more aggressive parallelism. In doing this, we tried to remove the communication bottleneck for the given number of processor cores, hence came up with a platform configuration which is again computation-limited.

6.2 Code Exploration

We ran two different code implementations: (a) one using *floating point* variables and (b) one using *fixed point integers* [19] with an exponent of 22. Figure 9 shows the results for the two different code implementations from *time (execution time)* and *energy (relative)* points of view. The ST220 processor core runs at 200MHz. We have performed the analysis for 3, 6 and 12 leads; furthermore we process each lead on a separate core.

We found that the precision of the results obtained with fixed point code, by using 64 bit integer data types representation, almost matches the results obtained with floating point code for a large number of input data traces. On the contrary, the time needed to process data, and also the energy required, decreases up to 5 times. This is mainly due to the fact that, like many commercial DSPs, our processor cores do not have a dedicated floating point unit. Therefore, floating point computations are emulated by means of a C SW-library linked at compile time. Figure 9 also shows that even with 12 concurrent processors, the bus is not saturated, since we observe negligible effects on the stretching of task execution times. In contrast, adding more processors determines a linear increase in energy dissipation.

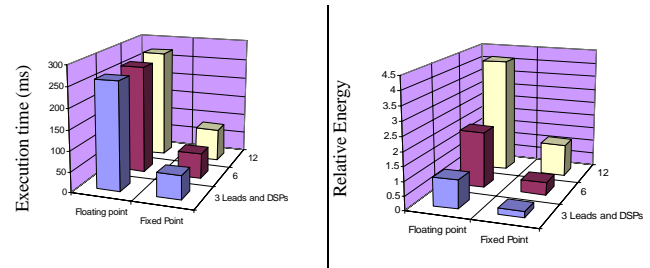


Figure 9. Comparison between different code implementations for the analysis of the 3-lead, 6-lead and 12-lead ECG. Data analysis for each lead is computed on a separate processor core. Sampling frequency of input data was 250Hz. System operating frequency is 200 MHz.

6.3 Exploration of Computation Resources

We then compared the performance of an ARM7TDMI general purpose processor core with that of our ST220 DSP cores, in order to assess the increased computation efficiency provided by the chosen VLIW DSPs when put at work with the computation kernel of our specific ACF application. In order to have a safe comparison, we set similar dimensions of the cache memory (32KB). Execution statistics are taken for processing one ECG-Lead at 250Hz sampling frequency. We count execution cycles to make up for the different clock frequencies. We adopt this single-core architecture in the first un-optimized HW/SW co simulation iteration, since our first aim is to investigate the computation efficiency of the two cores for our specific biomedical application, and de-emphasize system level

interaction effects such as synchronization mismatches or contention latency for bus access. In Figure 10, we can observe that the ST220 DSP proves more effective both in execution time and energy consumption, as expected. In detail, the ARM core is 9 times slower than the ST220 in terms of execution time, and it consumes more than twice the energy incurred by the DSP. These results can be explained based on three considerations from the SW and HW layers:

- The ST220 has better software development tools, which result in a smaller executable code.
- The ST220 is a VLIW DSP core, therefore it is able to theoretically achieve the maximum performance of 4 instructions per cycle (i.e., 1 bundle).
- A metric which is related to both previous considerations is the static instructions per-cycle, which depends on the compiler efficiency and on the multi-pipeline execution path of the ST220. For our application, this metric turns out to be 2.9 instructions-per-bundle for ST220.

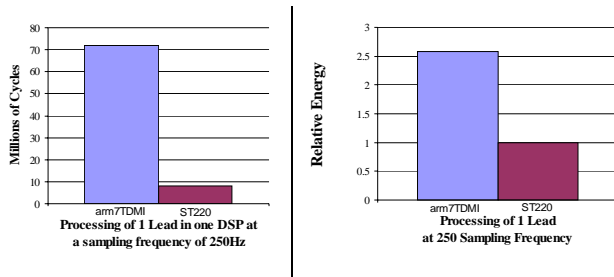


Figure 10. Comparing ARM7TDMI with ST220 DSP performances, when processing 1 Lead at 250Hz sampling frequency.

6.4 Required Level of Hardware Parallelism

Based on these findings HW/SW optimization is investigated, hence we can retain a multi-DSP architecture processing a fixed point coding implementation of the ACF algorithm as an effective solution for ECG analysis. Now the problem of finding the minimum number of DSP cores to activate for the application at hand arises.

We found that when sampling ECG data from biomedical sensors at 250 Hz, 1 DSP core is enough to meet the real-time requirements of the ACF SW-method (12 leads analysis): 0,2 seconds are taken, largely meeting the required 4 seconds for real-time computation of input data frames. Interestingly, even increasing the 1 KHz sampling frequency, and therefore increasing the computation workload of processor cores, 1 DSP is still able to complete the 12-lead analysis in slightly more than 3 seconds. This leaves about 1 second left to run diagnosis algorithms online (as a partition of the main algorithm). The increased sampling frequency led to a 15x increment of execution time and to a 90% increase in overall system energy. By increasing the number of processor cores in both cases, we got good scalability results (relative to execution time) up to 6 cores, with diminishing returns with increased parallelism. Unfortunately, the energy increases, since the power overhead of having more cores running is not offset by execution time savings. This was expected, for the deployed processor cores are not general purpose processors and are therefore more power-consuming. However, more processor cores can be used if the 1 second margin is not satisfactory, or if the platform is augmented with dynamic

voltage and frequency scaling (DVFS) support (this is work in progress). This latter solution would make the multiprocessor solution efficient not only from a performance perspective, but even from an energy viewpoint. The large slacks that are available in the current implementation make application of DVFS very promising.

Interestingly, using multiple cores to decrease execution time and have more margins for online diagnosis incurs a higher energy cost for the 250 Hz case than the 1000 Hz case, as illustrated in Figure 11. This is due to the fact that the increased workload in the 1 KHz case justifies an increased hardware parallelism.

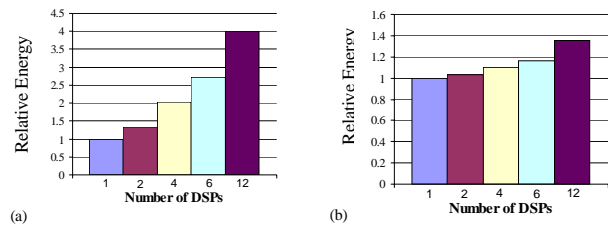


Figure 11. Energy scalability for: (a) the 250Hz, and (b) the 1000 Hz case.

6.5 Scalability

One main advantage of chip multiprocessing lies in its capability to meet the demand for higher sampling frequencies that are raised by the need to perform higher accuracy analysis and by the evolution of state-of-the-art sensor technology.

In order to prove this, we conducted some experiments [18] and measured the maximum sampling frequency at which our MPSoC platform can be operated while still meeting real-time requirements of the ACF application. We consider a fully parallel 12-lead application spread over 12 processors. The resulting 2.2 KHz frequency indicates poor scalability. The reason for this is mainly the interconnect performance, which does not scale any more. In fact, *bus busy* (the number of bus busy cycles over the total execution time) at the critical frequency of 2200Hz is almost 100% (99.95%), i.e., the bus is fully saturated. This is due to the fact that the amount of data being transferred across the bus increases linearly with the sampling frequency. In order to make the platform performance more scalable, we revert to a full-crossbar solution for the communication architecture. The benefits are clearly observed in Figure 12, where the maximum analyzable frequency (with respect to real-time constraints) amounts now to 4000Hz, i.e. nearly twice as much as the performance with a shared bus. Moreover, we observe that average bus transaction latencies at the critical frequency are still very close to the minimum latencies, thus indicating that the crossbar is very lightly loaded. Another informative metric is the bus efficiency (number of cycles during which the bus transfers useful data over the *bus busy* cycles), which amounts to 71.83%.

This very good performance is an effect of the lack of contention on the crossbar branches, which is in turn due to the high performance of the memory controller and to the matching of the application traffic patterns with the underlying parallel communication architecture. As a consequence, *with a full crossbar the system performance is no more interconnect-limited but computation-limited*. Since the computation workload of the system grows in a polynomial manner with the sampling frequency, it rapidly increases task execution times and reduces the available slack time

with respect to the deadline. We observe that the performance with an optimized partial crossbar closely matches that of a full-crossbar (less than 2% average difference) but with almost 3 times less hardware resources.

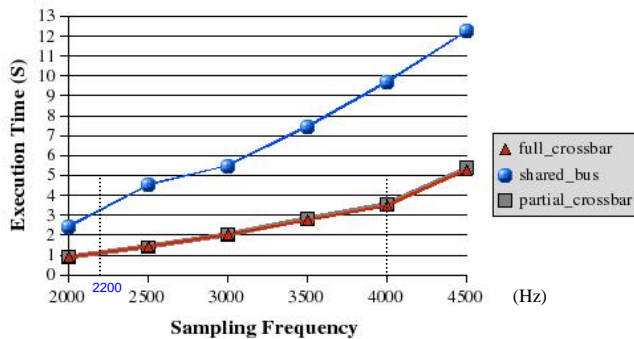


Figure 12. Critical sampling Frequencies for 3 architectures: (1) shared bus, (2) full crossbar, and (3) partial crossbar.

7. CONCLUSION

This paper illustrates the potential advantages that MPSoC technology can bring to real-time ECG analysis. They can be summarized as follows: larger time margin to run diagnosis algorithms, energy efficiency (provided DVFS support is available, and which is left for future work in this paper), improved scalability to challenging higher sampling frequencies and to more accurate ECG analysis algorithms. At the same time, this work goes through an application-specific design methodology for the ECG domain, which envisions full system modeling accuracy, high HW/SW parallelism exploitation, and computation and communication parallelism. Finally, the paper illustrates, for practical case studies, the advantages of deploying more computation-demanding analysis algorithms for the quality of ECG analysis.

8. REFERENCES

- [1] Fuster, V., Epidemic of Cardiovascular Disease and Stroke: The Three Main Challenges, *Circulation*, Vol. 99, Issue 9, March 1999, 1132-1137.
- [2] Jovanov, E., Gelabert, P., Adhami, R., Wheelock, B., Adams, R., Real Time Holter Monitoring of Biomedical Signals, *In Proceedings of the DSP Technology and Education Conference (DSPS'99)*, Aug 1999, Houston, Texas.
- [3] Oklobdzija, V., and Krishnamurthy, R., High-Performance Energy-Efficient Microprocessor Design (Series on Integrated Circuits and Systems), Springer; 1 edition August 9, 2006.
- [4] Fuster, V. Epidemic of Cardiovascular Disease and Stroke: The Three Main Challenges, *Circulation*, Vol. 99, Issue 9, March 1999, 1132-1137.
- [5] Pan, J. and Tompkins, W., A Real-Time QRS Detection Algorithm, *IEEE Transactions on Biomedical Engineering*, Vol. BME-32, No. 3, March 1985.
- [6] Hung, K., Zhang, Y. T., and Tai, B. Wearable Medical Devices for Tele-Home Healthcare, *In Proceedings of the 26th Annual International Conference on the IEEE EMBS*, San Francisco, CA, USA, September 1-5, 2004.
- [7] Jun, D., and Hong-Hai, Z., Mobile ECG detector through GPRS/Internet, *In Proceedings of the 17th IEEE Symposium on Computer-Based Medical Systems (CBMS'04)*, 2004.
- [8] Freescale™ semiconductor, Personal Electrocardiogram (ECG) Monitor, <http://www.freescale.com/>
- [9] Harland, C., Clark, T., and Prance, R. High resolution ambulatory electrocardiographic monitoring using wrist-mounted electric potential sensors, *Measurement Science and Technology*, Vol. 14, 2003, 923-928.
- [10] Medardoni, S., Ruggiero, M., Bertozzi, D., Benini, L., Strano, G., Pistritto, C., Capturing the interaction of the communication, memory and I/O subsystems in memory-centric industrial MPSoC platforms, *In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition, 2007 (DATE '07)*, Nice, France, 16-20 April 2007.
- [11] Loghi, M., Angiolini, F., Bertozzi, D., Benini, L., and Zafalon, R. Analyzing On-Chip Communication in a {MPSoC} Environment, *In Proceedings of Design and Test in Europe Conference (DATE)*, February 2004, 752-757.
- [12] Loghi, M., Poncino, M., and Benini, L. Cycle-Accurate Power Analysis for Multiprocessor Systems-on-a-Chip, *GLSVLSI04: Great Lake Symposium on VLSI*, April 2004, 401-406.
- [13] Bona, A., Zaccaria, V., and Zafalon, R. System level power modeling and simulation of high-end industrial network-on-chip", *In Proceedings of Design and Test in Europe Conference (DATE)*, February 2004, 318-323.
- [14] STBus Interconnect, STMicroelectronics: STBus main features, www.st.com/stonline/products/technologies/soc/stbus.htm
- [15] Bouyssounouse, B., Sifakis, J., Embedded Systems Design: The Artist Roadmap for Research and Development, Springer, 2006.
- [16] Poletti, F., Poggiali, A., and Marchal, P., Flexible Hardware/Software Support for Message Passing on a Distributed Shared Memory Architecture, *In Proceedings of Design and Test in Europe Conference (DATE'05)*, Nice, France, April 16-20, 2005, 736-741.
- [17] Aaron Segal: EKG tutorial, EMT-P, 1997, <http://www.drsegal.com/medstud/ecg/>
- [18] Al Khatib, I., Poletti, F., Bertozzi, D., Benini, L., Bechara, M., Khalifeh, H., Jantsch, A., Nabiev, R., A Multiprocessor System-on-Chip for Real-Time Biomedical Monitoring and Analysis: Architectural Design Space Exploration, *In Proceedings of 43rd Design Automation Conference (DAC'06)*, San Francisco, California, USA, July, 24-28, 2006.
- [19] ARM DAI 0033A Note 33: Fixed Point Arithmetic on the ARM, September 1996.