

A Multiprocessor System-on-Chip for Real-Time Biomedical Monitoring and Analysis: ECG Prototype Architectural Design Space Exploration

IYAD AL KHATIB

Royal Institute of Technology (KTH)

FRANCESCO POLETTI

University of Bologna

DAVIDE BERTOZZI

University of Ferrara

LUCA BENINI

University of Bologna

MOHAMED BECHARA

American University of Beirut (AUB)

HASAN KHALIFEH

American University of Beirut (AUB)

AXEL JANTSCH

Royal Institute of Technology (KTH)

and

RUSTAM NABIEV

Karolinska

In this paper we focus on multiprocessor system-on-chip (MPSoC) architectures for human heart Electrocardiogram (ECG) real-time analysis as a Hardware/Software (HW/SW) platform offering an advance relative to state-of-the-art solutions. This is a relevant bio-medical application, with good potential market since heart diseases are responsible for the largest number of yearly deaths. Hence, it is a good target for an application-specific system-on-chip (SoC) and HW/SW co-design. We investigate a symmetric multi-processor architecture based on STMicroelectronics VLIW DSPs that process in real-time 12-lead ECG signals. This architecture improves upon state-of-the-art SoC designs for ECG analysis in its ability to analyze the full 12 leads in real-time, even with high sampling frequencies, and ability to detect heart malfunction for the whole ECG signal interval. We explore the design space by considering a number of hardware and software architectural options. Comparing our design with nowadays solutions from a SoC and application point of view shows that our platform can be used in real time and without failures.

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

Additional Key Words and Phrases: multiprocessor system-on-chip, embedded system design, electrocardiogram algorithms, real-time analysis, hardware space exploration

This research was supported partly by the Swedish International Development Agency (SIDA).

Authors' addresses: I. Al Khatib, ICT, KTH, Stockholm, Sweden, F. Poletti, DEIS, University of Bologna, Bologna, Italy, D. Bertozzi, ENDIF, University of Ferrara, Ferrara, Italy, L. Benini, DEIS, University of Bologna, Bologna, Italy, M. Bechara, ECE, AUB, Beirut, Lebanon, H. Khalifeh, ECE, AUB, Beirut, Lebanon, A. Jantsch, ICT, KTH, Stockholm, Sweden, R. Nabiev, Karolinska University Hospital, Karolinska, Huddinge, Stockholm, Sweden.

Permission to make digital/hard copy of part of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date of appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.

© 2007 ACM 1073-0516/01/0300-0034 \$5.00

1. INTRODUCTION

The advance in embedded systems and multiprocessor trends pave the way for the development of single-chip solutions for computationally-intensive bio-medical applications with potential health benefits for a large number of individuals. One important application, in this respect, is the real-time remote and accurate analysis of human heart activity, which has always been a challenging problem for biomedical engineers. Heart disorders like Cardiovascular Disease (CVD) and stroke are the leading cause of death in the world for both women and men of all ethnic backgrounds [Fuster 1999]. In 2003, CVD alone is responsible for 29.2% of the total global deaths according to the World Health Organization (WHO), and this percentage is increasing every year [Fuster 1999]. More than 50% of these deaths can be saved with a reliable combination of cost-effective monitoring and accurate analysis [Fuster 1999].

Heart activity is electrically recorded as a set of electrocardiogram (ECG) signals which can readily reveal a number of heart malfunctions [Lo et al. 2005; Code Blue; BIOPAC]. The most reliable ECG analysis technique is the 12-lead ECG, which requires the reading and analysis of twelve different signals sensed from the patient's body. The main challenge arises from the high computational demand for processing huge amounts of ECG data in parallel, under stringent time constraints, relatively high sampling frequencies, and life-critical conditions [Haraland et al. 2003]. The challenges become even more complex when the patient is mobile and remotely monitored (as in cases of homecare and emergency at the point-of-need) [HSFC 1999], because state-of-the-art biomedical equipment for heart monitoring lack the ability to provide large-scale analysis and remote, real-time computation at the patient's location. This necessitates the transmission of huge amounts of life-critical data over a communication link to a large set of computing devices on another location [Code Blue]. In cases of mobile patients, this requires a 100% functional always-ON wireless connection since losing a few heart-beat data may be life threatening.

To overcome the aforementioned challenges and the problem of transmitting life-critical data on a wireless link (that is not reliable nor secure enough), the solution is to parallel-process the complex biomedical computations of the 12-lead ECG on a wearable MPSoC. Hence, the solution only transmits secure remote-alarm signals and only reports on the *results* of the analysis. These result-reports, are much smaller in size (a few bytes) than the ECG data (in Mega bytes), and if transmission fails, they can be re-transmitted until reception is acknowledged by the healthcare remote-monitoring center since they are saved on an off-chip memory for every analyzed ECG data chunk.

This technical objective calls for the design of special-purpose SoC architectures, featuring increased energy efficiency while providing high computation capabilities. 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 a number of real-time analyses of input data with high sampling frequencies, leveraging the computation horsepower and the functional flexibility provided by many (up to twelve) concurrent DSPs. The proposed architecture addresses usability, security and safety of the patients in emergency situations and long-term treatments. Comparison between our design and previous work shows the advantages of our design from a SoC performance point of view and from an application point of view. For instance, the comparison in section 5 with the *relatively old and still-nowadays-used* Pan-Tompkins algorithm [Pan and Tompkins 1985], shows that the Pan Tompkins algorithm may fail while our solution does not fail to detect the correct heart period. The biochip system 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 paper goes through all the steps of the design methodology, from application functional specification to hardware definition and modeling. System performance has been validated through functional, timing accurate simulation on a virtual platform. A 0.13 μ m technology homogeneous power estimation framework leveraging industrial power models is used for power management considerations.

2. BACKGROUND

Biomedical sensors today exhibit increased energy efficiency, therefore prolonged lifetimes (up to 24 hours), and higher sampling frequencies (up to 10 kHz for ECG) and often provide for wireless connectivity [Ambu]. Unfortunately, a mismatch exists between advances in sensor technology and the capabilities of state-of-the-art heartbeat analyzers [Harland et al. 2003]. They cannot usually keep up with the data acquisition rate, and are usually wall-plugged, thus preventing mobile monitoring. We aim at using state of the art commercial sensors from Ambu Inc. silver/silver chloride ‘Blue Sensor R’ [Ambu]. Moreover, most of the nowadays solutions look at a part of the ECG signal to detect whether the heart beat was healthy or unhealthy. Hence, this makes no use of the huge amount of information that may be gained from the state-of-the-art sensors and the

modern advance in electronics (SoC in particular). For instance, although many hospitals use modern sensors that can give very accurate data recordings, they still use relatively old methods/platforms to analyze the recorded data. Consequently, those relatively old methods, may still fail, and even if they don't fail, they can not analyze the full ECG signal, but rather analyze a part of the heart signal. This, in turn, gives only partial knowledge about the heart, thus keeping many disease cases obscure and dependent on the nurse's eyes (i.e. not as accurate as modern computer processing).

2.1 Application Specific Background

Our application is the 12-lead ECG, which uses nine sensors on the patient's body. With 3 of these sensors, physicians can use a method known as the 3-lead ECG, which suffers from the lack of detection of many diseases or malfunction in the heart. Using 9 sensors gives a more detailed view of the heart, which maximizes the detection of heart problems. Hence, maximizing the number of sensors maximizes the ability for disease detection. By interconnecting the nine sensors for the 12-lead ECG we get twelve biomedical voltage signals; hence, this produces huge amounts of data especially when used for a long number of hours. Physicians use the 12-lead ECG method, because it allows them to view the heart in its three dimensional form; thus, enabling detection of many abnormalities that may not be apparent in the 3-lead technique.

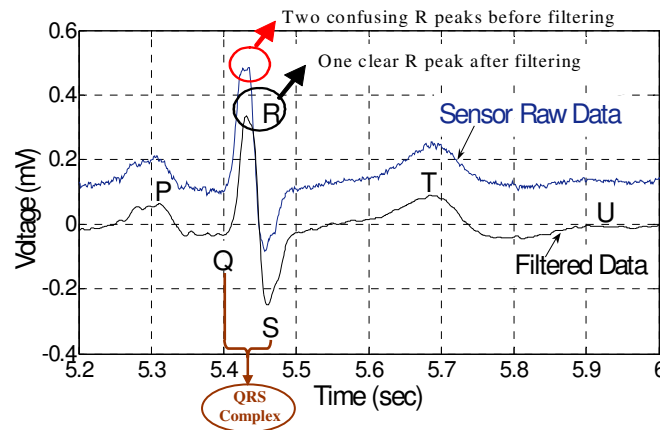


Fig. 1. ECG lead signal example: the upper curve is the Lead raw data and the lower curve is the filtered ECG lead.

Fig. 1 shows an example of a typical ECG signal, where the most important peaks are labeled: P, Q, R, S, T, and U. Each of these peaks and their relative inter-peak distances is related to a heart activity that is important for analysis. In addition, every combination

of different inter-peak interval values proves a type of heart malfunction. The higher the sampling frequency the more accurate is the analysis since there are cases of diseases, where two peaks are too close (especially the R and T peaks in the case of the R-on-T phenomenon [Segal 1997]) so that it becomes hard to detect the inter-peak distances and the heart period.

2.2 Previous Work

ECG monitoring and analysis have been explored in many companies and research organizations. However, we are not aware of any single-chip real-time analysis solution for full 12-lead ECG, which is able to accurately study the heart rhythmic period and can diagnose all the peaks: P, Q, R, S, T and U and their inter-peak intervals to result in a disease diagnosis. Previous work on ECG analysis can be classified into 4 types of solutions: (i) classical stationary-machine solutions [BIOPAC], (ii) SoC solutions [Chang et al. 2004; Freescale], (iii) Handheld device solutions [Hung et al. 2004; Jun and Hong-Hai 2004], and (iv) ASIC solutions [Desel et al. 1996]. The classical solutions do not allow for patient mobility or remote analysis since they are wall plugged, thus suffer from the need of many beds in the healthcare center. Moreover, in the classical medical technique for ECG analysis, the 12 lead signals are printed on eyeballing paper making the check of the different heart peaks and rhythms difficult and inaccurate due to its dependence on the physician's eyes. On the other hand, when using digital recording and filtering we can determine the peaks more accurately. The SoC solution in [Chang et al. 2004] does not run 12 lead analyses, but runs 1 lead per SoC. Consequently, to run 12-lead analyses with that solution means using 12 chips. One commercial solution [Freescale] takes 8 input sensor-lines and calculates lead signals and analyzes them on one DSP, hence it is time consuming. It only detects if the heart is healthy or unhealthy without analyzing diseases since it only detects the QRS without the P and T. Hence, it is not scalable. It uses 12 bits for the signals while we use 16 bits, thus, we add more accuracy to the analysis. The handheld solutions only read and transmit data. The ASIC solutions are just used for data acquisition before transmission.

3. THE PAN TOMPKINS ALGORITHM

The Pan Tompkins is the most used algorithm for heart beat abnormalities detection. Unlike our solution, which detects all peaks in order to check many abnormalities and diseases, the Pan Tompkins solution only detects whether the heartbeat is normal or abnormal. Hence, the Pan Tompkins algorithm is built to detect the QRS interval only. In what follows we discuss the Pan Tompkins solution, which we implemented depending

on the already existing Pan Tompkins algorithm [Pan and Tompkins 1985] in order to finally compare it with our solution. Thus, we show the efficiency in our HW/SW co-design by having real-time accurate analysis of the ECG signal via MPSoC. Many ECG instruments require the presence of an accurate QRS detector (see Fig. 1). Such detection is usually difficult due to the presence of noise in the ECG signal. The Pan-Tompkins algorithm combines the three known types of QRS detection: linear digital filtering, non-linear transformations, and decision rule algorithms. The technique consists of the following stages: analog filter, band-pass filter, derivative, squaring, and windowing. The algorithm was made to run on a Z80 (Zilog) or an NSC800 (National Semiconductor) microprocessor. Hence it is quite fast, but it may fail. The processing is done in integer arithmetic so that the algorithm operates in real-time [Pan and Tompkins 1985] without consuming excessive power. The algorithm detects QRS complexes only 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 $F_s=200\text{Hz}$. The resultant vector is passed through a band-pass filter implemented using a cascaded low-pass (cut-off frequency 15Hz) and high-pass (cut-off frequency 5Hz) filter stages to remove high-frequency noise, P-waves, T-waves, and other artifacts. The result is a 3dB pass-band from about 5Hz - 12Hz, which approximates the desirable pass-band to maximize the QRS energy (5Hz - 15Hz). In this stage, not only noise is removed, but also the SNR is improved so that lower thresholds can be used to enhance detection sensitivity. The band-pass filtered signal is then passed through a local peak detection algorithm which identifies and marks all the peaks found in the signal. It also uses a set of two moving threshold parameters $T1$ and $T2$ ($T1=2.T2$) to select candidate QRS complexes. If within a certain time interval, defined as the peak-search time, no new QRS peak is found which has larger amplitude than the $T1$ threshold, a search-back routine is then executed with $T2$ threshold level. Both $T1$ and $T2$ thresholds are adaptive and thus are modified based on the amplitude of the new peak found.

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. The digital band pass filter implemented is a cascade of a low pass and a high pass filter. The transfer functions of those filters are shown in (1) for the low pass filter, and (2) for the high pass filter.

$$H(z) = (1-Z^{-6})^2 / (1-Z^{-1})^2$$

(1)

$$H(z) = (-1+32Z^{-16}+Z^{-32}) / (1-Z^{-1})^2$$

(2)

This is the result of subtracting a first order low pass filter from an all-pass filter. The derivative used is a five-point derivative with the transfer function shown in (3).

$$H(z) = (1/8T)(-Z^{-2} - Z^{-1} + 2Z^{-1} + Z^2)$$

(3)

The squaring function squares the signal point by point according to the operation in (4).

$$Y(nT) = [x(nT)]^2$$

(4)

In addition to slope information, the moving window integrator helps in obtaining information about other waveform features. This is calculated from (5):

$$Y(nT) = (1N)[x(nT - (N-1)T) + x(nT - (N-2)T) + \dots + x(nT)]$$

(5)

where N is the number of samples in the width of the integration window, and this value must be approximately equal to the widest QRS complex. For the given sampling frequency of 200Hz, $N=30$ is used.

Using the band-pass filter, lower thresholds can be used because of the improved SNR ratio. Two adaptive thresholds are used for QRS detection, one is double the other. The higher threshold is used for the first analysis of the signal, while the other is used in the search-back technique whenever no QRS is detected within a certain time interval. The set of thresholds applied to the output of the moving window integrator are in (6) below:

$$\begin{aligned}
 SPKI &= 0.125PEAKI + 0.875SPKI \\
 NPKI &= 0.125PEAKI + 0.875NPKI \\
 THRESHOLDI1 &= NPKI + 0.25(SPKI - NPKI) \\
 THRESHOLDI2 &= 0.5THRESHOLDI1
 \end{aligned}$$

(6)

PEAKI is the overall peak, SPKI is the most recent running estimate of the signal peak, NPKI is the most recent running estimate of the noise peak, THRESHOLDI1 is the first threshold applied, and THRESHOLDI2 is the second threshold applied. In case the QRS complex is found using the second threshold, the following in (7) is used :

$$SPKI = 0.25PEAKI + 0.75SPKI$$

(7)

The same technique is implemented for peak detection of the output signal of the band-pass filter with THRESHOLDF1 and THRESHOLDF2. Moreover, the first thresholds of the two sets (THRESHOLDI1 and THRESHOLDF1) are halved in case of irregular heart rates. The average RR interval in (8) and (9) corresponds to the mean of the last recent 8 RR intervals. For managing irregular heart rates, another average RR interval is calculated from the most recent 8 beats having RR intervals that fall within certain limits.

$$RR\ AVERAGE1 = 0.125(RR_{n-7} + RR_{n-6} + \dots + RR_n)$$

(8)

where RR_n is the most recent RR interval.

$$RR\ AVERAGE2 = 0.125(RR'_{n-7} + RR'_{n-6} + \dots + RR'_n)$$

(9)

where RR'_n is the most recent RR interval that fell between the acceptable low and high RR-interval limits in (10):

$$\begin{aligned}RR\ LOW\ LIMIT &= 92\%RR\ AVERAGE2 \\RR\ HIGH\ LIMIT &= 116\%RR\ AVERAGE2 \\RR\ MISSED\ LIMIT &= 166\%RR\ AVERAGE2\end{aligned}$$

(10)

If no QRS is found during the RR MISSED LIMIT interval, the maximal detected peak is considered to be a QRS candidate, and the lower of the two thresholds is applied for the search-back technique. In this way, they avoid using long memory buffers for storing past ECG data, and require less computation time to implement the search-back technique, since we only need the values within RR MISSED LIMIT. Whenever the RR interval is less than 360ms, and the maximal slope occurring during the detected QRS is less than half of the QRS waveform that preceded it, this QRS complex is identified as a T-wave and not a QRS complex.

4. ACF-BASED ALGORITHM

Since MPSoC technology provides scalable computation horsepower, it allows running more computation-demanding applications [Ruggiero et al. 2005], which are (from an application view point) more accurate analysis algorithms. Consequently we propose the ACF based algorithm. In what follows in this section we discuss our Autocorrelation Function (ACF) based algorithm, which adds the advantage of detection of more diseases and also analyzing the whole heart signal with all peaks and all inter-peak intervals.

4.1 Filtering

Data provided by biomedical sensors suffers from several types of noise: DC-offset and hum noise, patient movements, and signal interference [Company-Bosch and Hartmann 2003]. To overcome all the problems related to sensor noise, we designed an IIR filter (implemented in hardware on a dedicated chip feeding an external SDRAM memory) that outputs its results in 16-bit binary format. Our IIR filter is of order 3, because it proves enough for our ECG analysis. Fig. 1 shows an example of our filter results. Since we use this step of filtering, then the algorithm for ECG analysis does not have to take care of filtering and suppressing noise. The filtering was discussed in more detail in Khatib et al. 2006. A note worth mentioning is that an A/D step (before filtering) is needed, and the more the data in the analog to digital conversion the better the analysis since cases like

the R-on-T phenomenon require higher resolution of input data. For this reason, we use a 16 bit A/D conversion in comparison to the 12 bit solution, which is used in some commercial products nowadays [Freescale].

4.2 Algorithm

Our proposed ECG-analysis algorithm (Fig. 2) is conceived to be parallel and hence scalable from the ground up. Since each lead senses and analyzes data independently, each lead can then be assigned to a different processor. So, to extend ECG analysis to 15-lead ECG for example or more, then what is required is to just change the number of processing elements in the system. The program reads a data file in chunks of four seconds. We discuss below the reason for the choice of the 4-second chunks. The data file mainly holds the values of the ECG at the lead in binary format. So by reading the data continuously every 4 seconds, we would be emulating a real sensor sending continuous data to an intermediate buffer that holds 4 seconds of data sampled at a certain frequency, typically 1000Hz. We used an autocorrelation function (ACF) based-methodology to calculate the period and other parameters of the heartbeat since it gives more accurate results than the conventional methods searching for the distance between two peaks. The autocorrelation we use as shown in (11) has a certain number of Lags (L) to minimize the computation for our specific application as discussed below. We validated our algorithm over several medical traces [Physiobank].

$$R_y[y] = \sum y[n].y[n-k]$$

(11)

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 10,000Hz, respectively. To run this algorithm with (11) it takes around 1.75 million multiplications. 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 Fig. 3. 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.

From a technical viewpoint, real-time processing of ECG data would allow a finer-granularity analysis with respect to the traditional eyeball monitoring of the paper ECG readout.

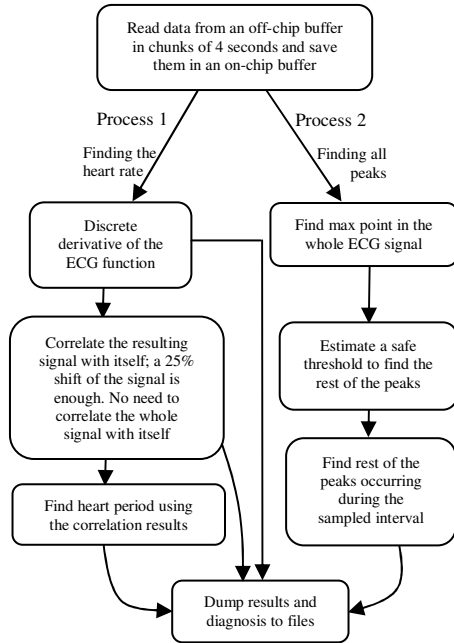


Fig. 2. Algorithm for analyzing each lead of the 12-lead ECG data for 250, 1000, and 10,000Hz sampling frequencies.

5. ALGORITHM COMPARISON

We realize a need to compare with the existing and still-widely-used Pan Tompkins algorithm since many medical institutes and hospitals depend on this solution, which we prove to fail. Our solution is not amplitude dependent, but time dependent. In this respect, we show that the Pan Tompkins solution may and will fail in some cases, while our solution doesn't fail. The reason for that is that we implemented an algorithm that does not depend on the amplitude peak values (P, Q, R, S, T, or U) for looking at the period between two peaks, but rather it looks at the relationship of the function repetitiveness to itself using the autocorrelation characteristics. Below we discuss these advantages.

Although the Pan Tompkins algorithm (discussed in section 3) consumes many computations, it is quite fast since it detects slopes and amplitudes. However, to compare

its efficiency and applicability, we ran it over the standard MIT/BIH arrhythmia database [MIT-BIH Database 1980].

In this respect, our ACF based solution runs over all the MIT/BIH arrhythmia data base cases without any failure, and it detected the heart period with an average percentage error of 3%. The error was calculated as discussed in the case studies below. However, when we ran the Pan Tompkins implementation on our platforms, we found out that the Pan Tompkins fails and may lead to erroneous results. Below are two examples of the failure of the Pan Tompkins solution.

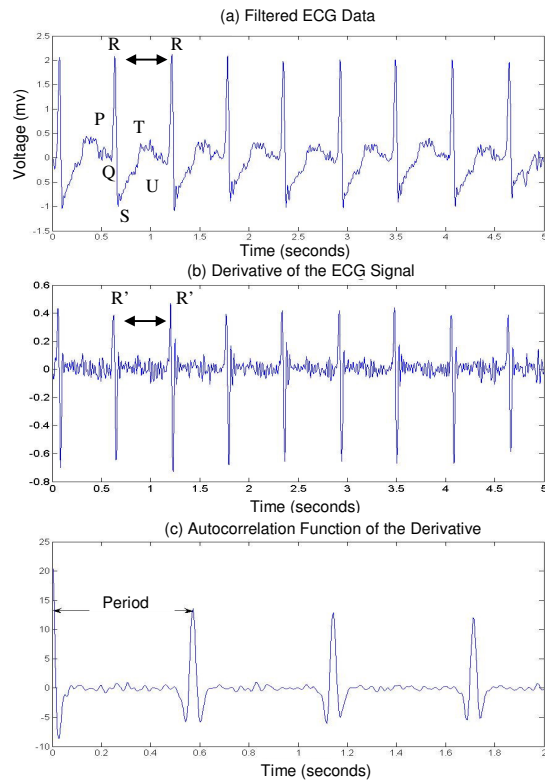


Fig. 3. Heart period analysis: (a) ECG signal peaks P, Q, R, S, T, and U; (b) derivative amplifying the R peaks, that we label as R'; (c) autocorrelation of the derivative with clear significant periodic peaks.

We list two cases of the MIT-BIH arrhythmia database, which failed (with a high percentage error) when using the Pan Tompkins solution, while our solution showed a much better performance.

5.1 First Case for Pan Tompkins Failure

The first case when the Pan Tompkins algorithm fails is when it runs on RECORD 217 (Male, age 65) data taken from Lead II [MIT-BIH Database 1980]. The trace characteristics are: paced beats, pacemaker fusion beats, and normal beats. The period using the eyeballing techniques, the Pan Tompkins algorithm, and our ACF-based solution are 0.5361 seconds, 0.8722 seconds, and 0.5500 seconds, respectively. The eyeballing period was calculated from the first 2 R-peaks of the ECG data. The error when using the Pan Tompkins solution (which is the difference between the periods calculated in Pan Tompkins and eyeballing divided by the Period using the eyeballing technique) is: 62.7%. Although it converges in a similar time as our ACF solution (<3.5s), but the results can be devastating for a patient. In the ACF case the error (which is the difference between the periods calculated in the ACF method and the eyeballing method divided by the period using the eyeballing technique) is only 2.6%, which is acceptable and still gives the physician a good margin to choose the right disease.

5.2 Second Case for Pan Tompkins Failure

The second case when the Pan Tompkins algorithm fails is when it is run on RECORD 208 (Female, age 23) data taken from Lead II [MIT-BIH Database 1980]. The trace characteristics are: ventricular couplets. The period using the eyeballing techniques, the Pan Tompkins algorithm, and our ACF-based solution are 0.6111 seconds, 0.2583 seconds, and 0.6556 seconds, respectively. In this case, the error using the Pan Tompkins solution is: 57.8%, and the nurses and medical team can easily realize the advantages of our HW/SW solution that can give more accurate results than the nowadays solutions and can converge in real-time to the solution.

The reason for the failure of the Pan Tompkins solution is that due to its afore-discussed algorithm, where it tries to find correct thresholds that will finally depend on the human choice. Moreover, it detects amplitude peaks, i.e. when the T and R waves are near, like the case of R-on-T phenomenon, the Pan Tompkins can not differentiate the R from the T and an R-T interval will be confused to be considered as an R-R interval, as revealed by the tests. However, using our ACF based SoC solution, we minimize the number of computations (although in millions) to suit the HW/SW co-design and find a method that takes the time axis as its point of reference and looks at the periodicity. Accordingly, we detect the period first, without a need for a threshold for the amplitudes.

Comparing our solution with that of the widely used Pan Tompkins shows that we do not waste any power or time on looking for thresholds, but rather use a concrete form of

ACF and ACF-derivative to find if the heartbeat is periodic, and we calculate the period immediately through the ACF. In this way, we leave no space for failure since we simply do not look at the amplitude axis. At the same time, we are able to design the hardware MPSoC that can cope with the required software and algorithm.

6. MPSOC ARCHITECTURE

After observing and proving that the ACF-based algorithm is clearly more accurate (while also providing for more analysis since it checks all the peaks), let us build a system that can handle its computations and let us tune the hardware platform for this algorithm. In order to process filtered ECG data in real-time, we choose to deploy a parallel Multi-Processor System-on-Chip 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 (Fig. 4). Since we are targeting a platform of practical interest, we choose advanced industrial components [Loghi et al. 2004a]. The processing elements are multi-issue VLIW DSP cores from STMicroelectronics, featuring 32kB instruction and data caches. These cores have 4 execution unit stages and rely on a highly optimized cross-compiler in order to exploit the parallelism. They leverage the flexibility of programmable cores and the computation efficiency of DSP cores. Moreover, these features allow reusing this platform for other biomedical applications other than the 12-lead ECG, thus making it cost-effective. Each processor core has its own private memory (512KB each), which is accessible through the bus, and can access an on-chip shared memory (8KB are enough for this application) for storing computation results. 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 was instantiated as the system communication backbone. STBus can be instantiated as both: a shared bus or as a partial or full crossbar. Thus it allows efficient interconnect design and provides flexible support for design space exploration.

In our first implementation, we target a shared bus to reduce system complexity (see Fig. 4), and we assess whether application requirements can already be met or not with

this configuration. We then explore also a crossbar-based system, which is sketched in Fig. 5. 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. 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.

In the above description, we have reported the worst case system configurations. In fact, fewer cores can be easily instantiated if needed. In contrast, 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 (10KHz, and 15 leads, for instance). The entire system has been simulated by means of the MPSIM simulation environment [Loghi et al. 2004a], which provides for cycle-accurate functional simulation of complete MPSoCs at a simulation speed of 200Kcycles/second (on average), running on a P4@3.5GHz. The simulator provides also a power characterization framework leveraging 0.13 μ m technology-homogeneous industrial power models from STMicroelectronics [Loghi et al. 2004b; Bona et al. 2004]. We believe that for life-critical applications, low-level accurate simulation is worth doing, although potentially slow, in order to perfectly understand system level behaviour and have a predictable system with minimum degrees of uncertainty. Each processor core programs the DMA engine to periodically transfer input data chunks onto their private on-chip memories. Moved data corresponds to 4 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 6Mbytes/sec, which is negligible for an STBus interconnect, whose maximum theoretical bandwidth with 1 wait state memories exceeds 400Mbytes/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 bytes, 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 history of 8 hours of computation can be stored. Data can also be remotely transmitted via a telemedicine link.

7. EXPERIMENTS AND RESULTS

We ran experiments in order to check the limits to respect the time figure of merit since our MPSoC is a real-time application based system. So we ran experiments to check the performance of each system design with increasing frequencies (up to 10KHz). We also ran experiments to look for optimizing the algorithm together with the design by changing some algorithm parameters and looking into the overall performance of the specific biomedical application on each MPSoC design. We also ran experiments by distributing the application on different numbers of DSP cores for each design (shared bus, crossbar, and partial crossbar). The results of these experiments are presented below. As a first exploration, we have compared the performance of an ARM7TDMI with that of the ST220 DSP, in order to verify the performances of the chosen VLIW with respect to the computation kernel of our specific application. In order to have a safe comparison, we set similar dimensions of the cache memory (32KB) for the two solutions, and we run two simulations for the processing of one ECG-Lead at 250Hz sampling frequency. We run a performance-comparison between two application-specific cores. We adopt this one core solution, because 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. Hence, the performance of the ARM7 core serves as a reference to assess the computation efficiency of the VLIW DSP core for the same specific application. In Fig. 6, we can observe that the LX220 DSP results in a better behavior in both: execution time and energy consumption. 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:

- (i) The ST220 has better software development tools, which result in a smaller executable code. The size of the executable code for the ARM is 1.7 times larger than that of the ST220
- (ii) 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)
- (iii) 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.

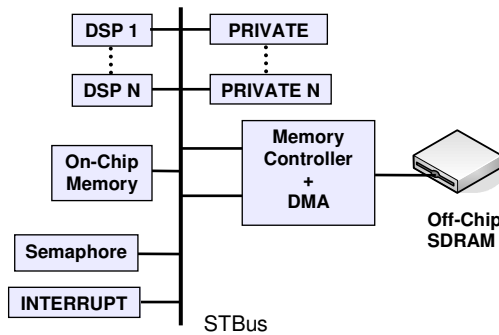


Fig. 4. Single bus architecture with STBus interconnect.

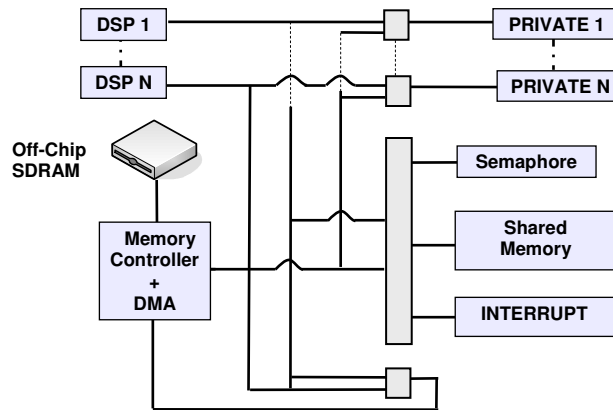


Fig. 5. Full Crossbar architecture with STBus interconnect.

Let us therefore select the best processor core for our computation kernel, from the performance and energy viewpoints. We now want to optimally configure the system to satisfy the application requirements at the minimum hardware cost. We, therefore, measure the execution time and the energy dissipation for an increasing number of DSP

cores in order to find the optimal configuration of the system. Since commercially available ECG solutions target sampling frequencies ranging from 250 to 1000Hz, we performed the exploration for these two extreme cases for the 12-lead ECG signal. We analyze a chunk of 4secs of input data, which provides a reasonable margin for safe detection of heartbeat disorders. Note that the computation workload for the processor cores increases in a polynomial manner with increasing sampling-frequency (due to the specific application algorithm).

Fig. 7 shows that if we increase the number of processors, the execution time scales linearly, which proves that second order effects typical of multi-processor systems (e.g. bus contention reducing the offered bandwidth to the processor cores with respect to the requested one) has only negligible effects on system performance, proving that the system is well configured and a single shared bus communication architecture is well suited for this application. However, this does not mean that the amount of data moved across the bus is negligible: around 100KB (at 1000Hz). This data is, however, read by the processor core throughout the entire execution time, thus absorbing only a small portion of the bus bandwidth. In this regime, the bus performance is still additive.

Moreover, the perfect scalability of the application is also due to memory controller performance. In fact, at the beginning of the computation each processor loads processing data from the off-chip to the on-chip memory, hence, requiring peak memory controller bandwidth. The architecture of the memory controller proves capable of providing the required bandwidth in an additive fashion. By looking at the 1000Hz plot (Fig. 8), we observe that 1 DSP is able to process an ECG-lead in slightly more than 3 seconds. Therefore, we still have about 1 second left (before the 4secs deadline), which is enough to perform additional analysis of the results of the individual lead-computations and converge to a decision about the heart period and malfunctions. Looking forward, we try to understand how our solution situates itself with respect to the demand for higher sampling frequencies raised by the need to perform higher accuracy analysis and the evolution of state-of-the-art sensor technology. We, therefore, measure and plot the maximum sampling frequency at which our MPSoC solution can be operated while still meeting real-time requirements. This frequency translates to 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 Fig. 9, 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%.

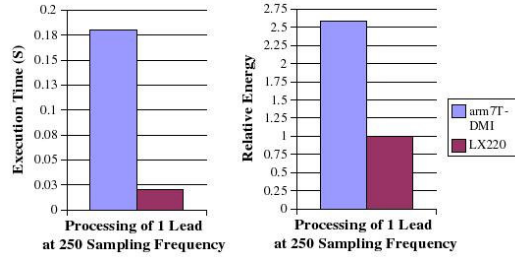


Fig. 6. Comparing ARM7TDMI with a ST200 DSP performances, in processing 1 Lead at 250Hz sampling frequency.

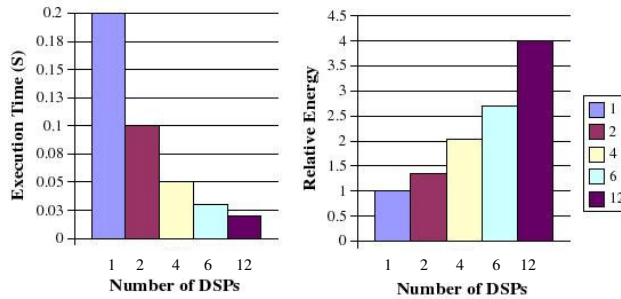


Fig. 7. Execution Time and Relative Energy Consumption of shared bus at 250Hz sampling frequency.

We simulate the 12-processor system to get the upper bound on system performance and push the architecture to the limit. For the same reason, we restrict the analysis period to 3.5secs, which is the minimum value of the input data-chunk time-span derived from the biomedical algorithm. The results presented in Fig. 9 show that with shared-bus architecture, the maximum sampling frequency that the MPSoC can handle without going beyond the real-time constraint is only around 2200Hz.

This 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.

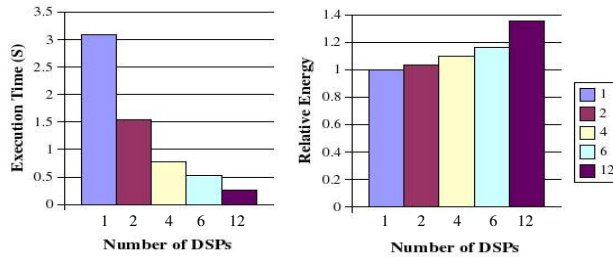


Fig. 8. Execution Time and Relative Energy Consumption of shared bus at 1000Hz sampling frequency.

We observe that the performance with a partial crossbar closely matches that of a full-crossbar (less than 2% average difference) but with almost 3 times less hardware resources. We found the optimal partial crossbar configuration (5x5 instead of a 13x13) by accurate characterization of shared bus performance. On a shared bus, we increased the number of processors and observed when the execution times started deviating as an effect of bus contention.

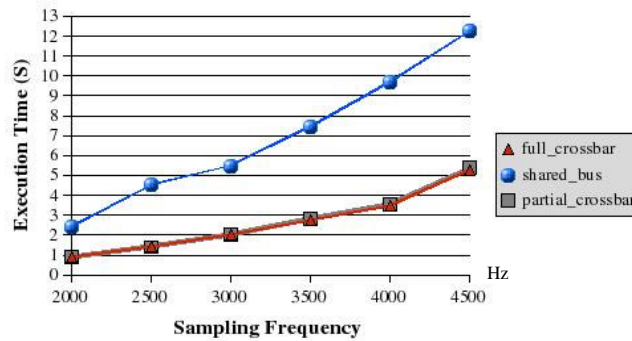


Fig. 9. Critical sampling Frequencies for 3 architectures: (1) shared bus, (2) full crossbar, and (3) partial crossbar.

With up to 4 cores connected to the same shared communication resource, this latter is able to work in an additive regime. Although the architecture cannot work in real-time at more than 4000Hz, we wanted to measure the execution time under non-real-time computation. In fact, from the execution time to process 3.5secs of input data, we can derive the amount of buffering that is required to store incoming data from the lead. By knowing the overall capacity of the off-chip memory, we then derive the maximum analysis time that we can afford at such high frequencies. Results are shown in Fig. 10.

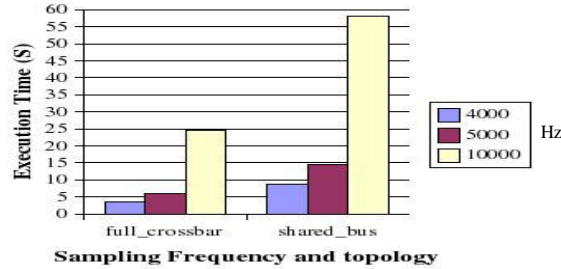


Fig. 10. Analysis-time investigation with increasing frequencies (up to 10KHz) for the ECG application.

Let us focus on the shared bus case, with 10KHz. The execution time for analyzing 3.5secs is a bit less than 1 minute (around 57 seconds), which is 16 times more than the real-time constraint. As a consequence, we can still decide to perform this kind of analysis, but we need to buffer 16 input data chunks while we are processing one chunk. Since an off-chip SDRAM memory can be 512MB, we can perform 3.5 minutes analysis before saturating the memory. With a full crossbar, this time amounts to around 14 minutes.

8. SOLUTIONS COMPARISON

We compare our platform to the existing solutions from an application point of view and SoC point of view. From the SoC view point, the advantages of our design are mainly in the performance for real time analysis and power consumption, where our experiments show that we can converge to a full analysis in a time less than the time needed to read the new coming data from the heart. Some more results of comparison are shown in Table I. From the application side, we are interested in the comparison with nowadays applications used on SoCs (Table I) after we have compared it (in section 5) with the nowadays used Pan Tompkins algorithm.

Comparing our application-based MPSoC designs, we can choose the best architecture relative to the biomedical purpose. Hence, for a solution that competes with and performs better than the existing commercial solutions [Freescale] (input sampling frequency from 250 to 1000Hz), we adopt our shared bus system architecture (Fig. 4) since its advantages over existing solutions are as follows:

- The ECG SoC available solution nearest to ours is the one presented in Chang et al. [2004], but our design performs better on analysis time-consumption. Our solution is easier to deploy since the 12 leads are input to one SoC instead of having 12 SoCs (i.e. cheaper to scale with increasing leads)
- In our designs, we can do full 12-lead ECG analysis at relatively high frequencies. Our design is optimal especially that we can offer the choice of the SoC architecture (shared bus, crossbar, or partial crossbar) based on the biomedical need of a frequency range.

Table I shows the advantage of our three designs to the best available designs we are aware of in the research and in the market.

Table I. Comparison between ECG-analysis SoC research-solution [Chang et al. 2004], available SoC commercial-solution [Freescale], and our three MPSoC designs: ST-PCB is partial crossbar, ST-CB is Full Crossbar, and ST-SB is the Shared Bus solution. Fs is the sampling frequency

<i>Solution</i>	Data bits	Memory	Fs (Hz)	Analysis Time	SoC Input	Pre-filter	Application results
[Chang et al. 2004]	10	8KB Cache	250	No study	1	Notch	Only QRS, only decide if the heart is healthy or unhealthy
[Freescale]	12	No available information	800	No study	8	IIR	Only QRS, only decide if the heart is healthy or unhealthy
ST-PCB	16	Off chip = 512MB 32kB I-cache/DSP 32kB D-cache/DSP	4,000	< 3.5s	12	IIR	Full 12-LEAD: heart-period discovery; P,Q,R,S,T peaks detection; allows disease detection
ST-CB	16	Off chip = 512MB 32kB I-cache/DSP 32kB D-cache/DSP	4,000	< 3.5s	12	IIR	Full 12-LEAD: heart-period discovery; P,Q,R,S,T peaks detection; allows disease detection
ST-SB	16	Off chip = 512MB 32kB I-cache/DSP 32kB D-cache/DSP	2,200	< 4s	12	IIR	Full 12-LEAD: heart-period discovery; P,Q,R,S,T peaks detection; allows disease detection

9. CONCLUSIONS

We present an application-specific MPSoC architecture for real-time ECG analysis as a solution for a large medical problem (CVD and stroke), which leads yearly to the highest number of deaths. Our solution leverages the computation horsepower of many (up to 12)

concurrent DSP cores to process ECG data. This solution paves the way for novel healthcare delivery scenarios (e.g. mobility) and for accurate diagnosis of heart-related diseases. We describe the design methodology for the MPSoC and explore the configuration space looking for the most effective solution, performance and energy-wise. We present three interconnect architectures (single bus, full crossbar, and partial crossbar) and compare them with existing solutions. The sampling frequencies of 2200Hz and 4000Hz, with 12 DSPs are found to be the critical points for our Shared-Bus design and Crossbar architecture, respectively. We compare our solution with the nowadays existing solutions in research, in the market, and in many hospitals. Our solutions offer significant advance steps in SoC HW/SW co-design, and it proves no single failure case when run and tested on the standard MIT-BIH arrhythmia database, while the widely used Pan Tompkins solution failed in some cases, of which we reveal two cases.

REFERENCES

- AMBU, INC., *Biomedical devices company*, www.ambuusa.com
 BIOPAC SYSTEMS INC., <http://biopac.com/>
 BONA, A., ZACCARIA, V., AND ZAFALON, R. 2004. System level power modeling and simulation of high-end industrial network-on-chip. In *Proceedings of Design and Test in Europe Conference (DATE)*, Paris, France, February 2004, 318-323.
 CHANG, M., LIN, Z., CHANG, C., CHAN, H., AND FENG, W. 2004. Design of a System-on-Chip for ECG signal processing, *The 2004 IEEE Asia-Pacific Conference on Circuits and Systems*, Tainan, Taiwan, December 2004.
 CODE BLUE, MEDICAL WIRELESS SENSOR NETWORKS, <http://www.eecs.harvard.edu/~mdw/proj/codeblue>
 COMPANY-BOSCH, E., HARTMANN, E. 2003. ECG Front-End Design is Simplified with MicroConverter, *Journal of Analog Dialogue*, Vol. 37, 10-15.
 DESEL, T., REICHEL, T., RUDISCHHAUSER, S., AND HAUER, H. 1996. A CMOS Nine Channel ECG Measurement IC, *2nd International Conference on ASIC*, Shanghai, People's Republic of China, October 1996, 115-118.
 FREESCALE™ SEMICONDUCTOR, Personal Electrocardiogram (ECG) Monitor, <http://www.freescale.com/>
 FUSTER, V. 1999. Epidemic of Cardiovascular Disease and Stroke: The Three Main Challenges, *Circulation*, Vol. 99, Issue 9, 1132-1137.
 HARLAND, C., CLARK, T. AND PRANCE, R. 2003. High Resolution Ambulatory Electrocardiographic Monitoring Using Wrist-Mounted Electric Potential Sensors, *Measurement Science and Technology*, Vol. 14, 923-928.
 HSFC-HEART AND STROKE FOUNDATION OF CANADA 1999. The Changing Face of Heart Disease and Stroke in Canada 2000, *Annual report 99*.
 HUNG, K., ZHANG, Y. T., AND TAI, B. 2004. Wearable Medical Devices for Tele-Home Healthcare, *Engineering in Medicine and Biology Society, IEMBS apos;04 26th Annual International Conference of the IEEE EMBS*, Issue 2 Vol. 7, San Francisco, CA, USA, September 2004, 5384-5387.
 JUN, D., AND HONG-HAI, Z. 2004. Mobile ECG detector through GPRS/Internet, In *Proceedings of the 17th IEEE Symposium on Computer-Based Medical Systems (CBMS'04)*, Bethesda MD, USA, June 2004, 485-489.
 KHATIB, A. I., BERTOZZI, D., POLETTI, F., BENINI, B., JANTSCH, A., BECHARA, M., KHALIFEH, H., HAJAR, M., NABIEV, R., AND JONSSON, S. 2006. MPSoC ECG Biochip: a Multiprocessor System-on-chip for Real-time Human Heart Monitoring and Analysis, In *Proceedings of Computing Frontiers 2006 (CF'06)*, Ischia, Italy, May 2006, 21-28.
 LO, B., THIEMJARUS, S., KING, R., AND YANG, G. 2005. Body Sensor Network– A Wireless Sensor Platform for Pervasive Healthcare Monitoring, In *Adjunct Proceedings of the 3rd International Conference on Pervasive Computing (PERVASIVE'05)*, Munich, Germany, May 2005, 77-80.
 LOGHI, M., ANGIOLINI, F., BERTOZZI, D., BENINI, L., AND ZAFALON, R. 2004a. Analyzing On-Chip Communication in a MPSoC Environment, In *Proceedings of Design and Test in Europe Conference (DATE'04)*, Paris, France, February 2004, 752-757.
 LOGHI, M., PONCINO, M., AND BENINI, L. 2004b. Cycle-Accurate Power Analysis for Multiprocessor Systems-on-a-Chip, *GLSVLSI04: Great Lake Symposium on VLSI*, Boston, MA, USA, April 2004, 401-406.
 MIT-BIH ARRHYTHMIA DATABASE 1980. Tape directory and format specification, Document BMEC TR00, *Mass. Inst. Tech.*, Cambridge 1980.

- PAN, J. AND TOMPKINS, W. 1985. A Real-Time QRS Detection Algorithm, *IEEE Transactions on Biomedical Engineering*, Vol. BME-32, No. 3, 230-236.
- PHYSIOBANK, Physiologic signal archives for biomedical research, <http://www.physionet.org/physiobank/database/ptbdb/>
- RUGGIERO, M., ACQUAVIVA, A., BERTOZZI, D., AND BENINI, L. 2005. Application-Specific Power-Aware Workload Allocation for Voltage Scalable MPSoC Platforms, In *Proceedings of the International Conference on Computer Design (ICCD 2005)*, San Jose, CA, USA, October 2005.
- SEGAL, A. 1997. *EKG tutorial, EMT-P (1997)*, <http://www.drsegal.com/medstud/ecg/>

Received August 2006; revised August 2007; accepted October 2007.