Rapid Processor Customization for Design Optimization: A Case Study of ECG R-peak Detection

Mladen Milosevic, Emil Jovanov, Aleksandar Milenković
ECE Dept., The University of Alabama in Huntsville
Huntsville, AL, U.S.A.
{mladen.milosevic, jovanoe, milenka}@uah.edu

***Abstract*—In engineering wireless body area network platforms it is crucial to meet performance goals at minimal hardware cost and energy required. This paper describes a design flow that relies on processor customization of Tensilica's Xtensa processor cores. We introduce custom instructions to expedite wavelet processing used in an ECG R-peak detection application. We explore several instruction extensions and show that customized processor cores significantly reduce program execution time and energy requirements for this application.**

I. Introduction

Recent technological advances in sensors, low-power integrated circuits, and wireless communications have enabled the design of low-cost, miniature, lightweight, and intelligent physiological sensor nodes. These nodes, capable of sensing, processing, and communicating one or more vital signs, can be seamlessly integrated into wireless body area networks (WBANs) for wellness or health monitoring. The WBANs integrated with cloud-based medical services through the Internet are well-positioned to revolutionize the way people manage their health and wellness, by allowing inexpensive, non-invasive, continuous, ambulatory health or wellness monitoring, with almost real-time updates of medical records [1]. Such systems are a key technology in helping transition toward more proactive and affordable healthcare that focus on managing wellness rather than illness, preventive care, and early detection of disease.

System designers of wearable health monitoring systems face a number of challenges as these platforms must reliably operate for extended periods of time, under stringent resource constraints in energy, communication bandwidth, memory capacity, and processing power. Early prototypes of WBAN platforms rely on off-the-shelf embedded processors that are designed to support a broad range of applications, and they may not be well-suited for physiological monitoring applications. On the other side, recent advances in engineering complex systems-on-the-chip [2] allow designers to rapidly explore the design space and customize embedded processor cores to meet performance requirements and design constraints.

In this paper we describe a design flow that allows rapid design space exploration and hardware customization by utilizing Tensilica's Xtensa design environment [3]. As an example WBAN application we use an R-peak detection. This application processes an input electrocardiogram signal (ECG) to determine time interval between two heart beats using a discrete wavelet transform. We start with the algorithm development and tuning in Matlab, and port this application to Tensilica's Xtensa processing core. The next step in the design flow is application profiling to identify critical sections in the code. To accelerate critical sections and reduce the total energy consumed we introduce several custom instructions that are specifically designed for this application and integrate them into the Xtensa instruction set. We consider several design alternatives that differ in software implementation and hardware complexity. These candidate designs are then evaluated in a multi-dimensional design space that encompasses processor performance, hardware complexity, code size, and power consumption.

The results of our experimental evaluation confirm that instruction set customization can significantly improve performance and reduce energy of the R-peak detection algorithm at the cost of slightly increased hardware complexity. For example, our best performing configuration with instructions that expedite wavelet computation speeds up the R-peak detection 5.7 times (19 times in the wavelet transform), reduces code size for almost 2 times, and requires almost 4.5 times less energy relative to the base configuration. We believe that hardware customization can provide similar or higher improvements in other WBAN applications as well.

The rest of this paper is organized as follows. In Section II we describe the development and tuning of the R-peak detection algorithm. Section III describes five hardware/software configurations based on Tensilica's Xtensa 9 processor core, including two that support our custom instructions designed to expedite wavelet processing. Section IV gives the results of our experimental evaluation that

978-1-4577-1470-2/11/$26.00 ©2011 IEEE

compares the configurations for execution time, code size, energy consumption, and hardware complexity. Section V gives concluding remarks.

II. ECG R-PEAK DETECTION ALGORITHM

Electrocardiogram (ECG) represents electrical manifestation of the contractile activity of the heart and it is one of the most important physiological signals. The ECG is characterized by a recurrent wave sequence of P, QRS, T and U waves associated with each heart beat. RR-intervals or inter-beat intervals are one of the most frequently used parameters. RR-intervals are crucial in heart rate variability (HRV) analysis; lower HRV may indicate congestive heart failure, diabetic neuropathy, and other medical conditions. In order to extract RR-intervals from an ECG signal we need a reliable and precise algorithm to detect R peaks.

A number of approaches, varying in complexity and precision, have been developed for R-peak detection [4]. In this paper we focus on an R-peak detection algorithm that uses a discrete wavelet transform [5], specifically the Daubechies D4 (Figure 1, lines 1-7). The first step is to process a raw ECG signal using four recursive calls of the Daubechies D4 wavelet transform as shown in Figure 1 (lines 9-12). The absolute values of the processed signal are then filtered using a threshold filtering: all samples lower than a threshold are set to zero. The threshold is set to be 15% of the maximum in the processed signal. The processed samples are kept in a sliding window array. This array is searched for local maximums, which represent potential R-peak signals. Once the potential R-peaks are located in the processed signal, the original ECG signal is searched to precisely locate a potential R-peak. Finally, the potential R-peak is upgraded to a true R-peak if there were no other true R-peaks in the previous 200 ms.

The original R-peak detection algorithm, which is geared toward post-processing of ECG signals, is adapted for real-time implementation. We apply the wavelet transform on input vectors with 64 samples. Four rounds of the wavelet transform give four processed samples that are used in detecting R-peaks as described above. The maximum used in the threshold filtering is determined on a 5 second training period, rather than on an entire ECG signal.

The real-time R-peak detection algorithm is first verified in Matlab. As an input we used a selected subset of ECG recordings from the MIT Physionet database [6]. The selected ECG recordings cover a wide range of possible ECG signals, differing in amplitude, sampling frequency, offset, and units of measure (mV/μV). Figure 2 illustrates the R-peak detection algorithm for one exemplary ECG signal from the database. The signal is sampled with sampling frequency of 750 Hz, has an offset of -6 mV, and units of measure are in millivolts. The original ECG signal (blue) is processed by the 4 rounds of wavelet transform to get the processed signals (red). The potential R-peaks are marked with green circles, and detected R-peaks are marked with red circles. The visual inspection of the R-peaks on the selected set of ECG signals confirms high precision and robustness of the algorithm – the number of false positives and false negatives is negligible and present only in cases of poor quality ECG signals.

```
1.   function [s d] = D4_Transform(S)
2.     N = length(S);
3.     s1 = S(1:2:N-1) + sqrt(3)*S(2:2:N);
4.     s2 = S(2:2:N) - sqrt(3)/4*s1 - (sqrt(3)-
       2)/4*[s1(N/2) s1(1:N/2-1)];
5.     s = (sqrt(3)-1)/sqrt(2) * (s1 -
       [s2(2:N/2) s2(1)]);
6.     d = (sqrt(3)+1)/sqrt(2) * s2;
7.   end

8.   function D4 = wavelet_trans(S)
9.     [S1 D1] = D4_Transform(S);
10.    [S2 D2] = D4_Transform(S1);
11.    [S3 D3] = D4_Transform(S2);
12.    [S4 D4] = D4_Transform(S3);
13.  end
```

Figure 1 Daubechies D4 transform used in ECG processing.

III. PROCESSOR CUSTOMIZATION

Wearable devices for health monitoring are powered by batteries. To increase the operating time and allow for small and lightweight devices, reducing energy consumed by typical health monitoring applications is of utmost importance. Whereas application specific integrated circuits require minimal energy, such solutions lack flexibility offered by microprocessor-based systems. On the other side, general-purpose embedded processors may offer a good performance for a broad range of applications, but they may be suboptimal for a given application. An alternative approach is to use customizable processors that allow designers rapid and risk-free configuration and customization. We use a Tensilica's Xtensa customizable processor and demonstrate how it can be customized for the R-peak detection algorithm.

Tensilica offers a set of tools and techniques for rapid design exploration and customization of the Xtensa embedded processors to fit specific needs of the target application. The Xtensa Xplorer design environment allows designers to quickly profile the application software, configure the Xtensa core, and add new instructions to optimize performance.

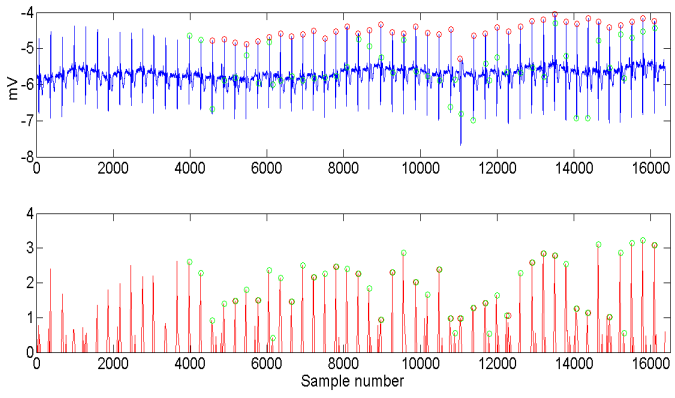

Figure 2 R-peak detection algorithm: an illustration.

System designers can explore multiple processor configurations and architectural enhancements by making area, speed, power and code-density design tradeoffs, based on real-time feedback from the design environment. Once the design requirements are met, Xtensa Processor Generator automatically creates a tailored application specific embedded processor, including a matching tool chain.

System designers can build a processor core by selecting and configuring a broad range of features, including: (a) choosing among several basic processor cores that differ in performance and cost; (b) configuring the number and type of functional units (e.g., integer multipliers or dividers, multiply-and-accumulate units); (c) enabling special instructions (e.g, zero-loop, conditional stores); (d) selecting and configuring size and organization of instruction and data caches; (e) configuring specialized memories and buses, (f) memory protection options, and so on.

The Xtensa instruction set architecture (Xtensa ISA) can be extended with user-defined instructions. Tensilica Instruction Extension (TIE) language [3] can be used to specify new states, register file extensions, new instructions, coprocessor extensions, and new data transfer interfaces. For new instructions the TIE language allows designers to specify schedules – pipeline stages at which instructions use input operands and produce outputs.

To evaluate effectiveness of the proposed hardware customization for the R-peak detection application, we consider five hardware/software configurations, shown in Table I, named BASE, FLPC, FIXP, FIPM, and DW3O. As we target a low-end embedded application, we use an ultra low-power and low-complexity Xtensa 9 processor core as the BASE configuration. It includes a fully pipelined integer 32-bit multiplier, 1 kilobyte instruction cache, and 1 kilobyte data cache. With maximum frequency of 250 MHz, the estimated power consumption is only 12mW [3].

The R-peak detection application is ported from Matlab to C with minimal changes. The input ECG signal is represented as single-precision floating-point data and all wavelet operations and the R-peak detection are carried out on this data. The BASE processor configuration does not include a floating-point co-processor, so floating-point computations are emulated in software. Profiling the BASE configuration confirms our expectation that the wavelet computation is indeed a performance bottleneck.

The second processor configuration FLPC adds a floating-point co-processor with support for instructions that can operate on single-precision floating-point numbers. We expect this configuration to significantly improve performance, at the cost of additional complexity caused by the floating-point co-processor.

The next step in design space exploration is to re-design the R-peak detection software to read and process ECG samples represented in fixed-point data formats. We opt for a 16:16 format (both integer and fractional portions are 16-bit long), because it provides a good balance between the range and precision, without any assumptions on the scale and units of the input ECG signal. The FIXP configuration has a hardware configuration identical to BASE, but the software implementation is re-designed to read and compute fixed-point numbers. The critical operations are multiplication, addition, and subtraction of fixed-point numbers (see lines 3-6 in Figure 1).

TABLE I. HARDWARE/SOFTWARE CONFIGURATIONS.

Name	Hardware Extensions			Software Impl.
	FPU Coproc.	*HW FixedP MUL*	*Wavelet OPER(1-3)*	*Data types for ECG samples*
BASE	×	×	×	Floating-point
FLPC	√	×	×	Floating-point
FIXP	×	×	×	Fixed-point
FIPM	×	√	×	Fixed-point
DW3O	×	√	√	Fixed-point

```
1.  operation mul {out AR outR, in AR inpR1, in
    AR inpR2}{}
2.  {
3.      wire do_signed = 1'b1;
4.      wire cbit = 0;
5.      wire[63:0] do_const = 32768;
6.      wire[63:0] temp = TIEmul(inpR1[31:0],
        inpR2[31:0], do_signed);
7.      wire[63:0] temp1 =
        TIEadd(temp,do_const,cbit);
8.      assign outR = temp1[47:16];
9.  }
```

Figure 3 TIE language description of fixed-point multiplier.

Profiling of the new R-peak detection application shows that about 70 percent of the execution time is spent in fixed-point multiplication. The fixed-point addition is identical to integer addition. However, fixed-point multiplication requires one integer multiplication and one addition. A custom instruction, shown in Figure 3, is added to support this operation. This hardware/software configuration is named FIPM. We expect this configuration to yield significant performance improvements at the cost of added complexity.

Finally, the last configuration DW3O adds 3 more characteristic instructions to accelerate typical operations found in D4 wavelet transform (lines 3-6 in Figure 1). Rather than including separate hardware resources for each of these operations, they all share a fixed-point multiplier. We expect this configuration to further improve performance at the cost of additional hardware complexity.

IV. RESULTS

In our experimental evaluation we compare performance, code density, and hardware complexity for all five configurations. We consider execution time spent in the D4 wavelet transform and the total time spent in the R-peak detection algorithm. We also consider the code size for the wavelet transform procedure and the R-peak detection procedure. These parameters are reported by the Xtensa Xplorer design environment. In addition, we evaluate the total energy required for each hardware/software configuration assuming a 45 nm technology process.

The main results of our experimental evaluation are shown in Table II. As expected, the BASE and FIXP configurations have the worst performance, requiring ~18.4 and ~15.7 million clock cycles for the R-peak detection. The majority of time is spent in the D4 wavelet transform, 15.9 and 13.3 million of clock cycles, respectively. However, the processor core is rather small and requires mere 22 kilogates. The FLPC configuration with a floating-point co-processor improves performance of the wavelet processing for more than 4.4 times relative to the BASE configuration, and the performance of the R-peak detection for slightly over 3 times. However, it requires an additional 12 kilogates, for the total hardware complexity of slightly over 34 kilogates.

The processor core with a customized fixed-point multiplication instruction, FIPM, achieves performance improvement over the BASE configuration for over 14 times in the wavelet transform function, and 5.2 times in the R-peak detection algorithm. This result is achieved at the cost of additional ~6 kilogates relative to the base configuration, which is one half of the complexity required by the floating-point co-processor. Finally, the extensions that support characteristic operations found in the wavelet transform, DW3O, achieve the best performance, improving the wavelet transform function performance for almost 19 times relative to the BASE configuration. However, the overall R-peak performance is improved only slightly relative to the FIPM. This is not surprising, if we know that the fixed-multiplier is a shared resource, and the time spent is the wavelet transform cease to be a bottleneck at this design point.

The results in Table II for code size show that the customized instructions used in the FIPM and DW3O configurations significantly reduce the code size in the wavelet transform procedure, whereas the size in the R-peak detection function remain fairly constant.

Figure 4 shows a statistical estimation for the total energy spent in the R-peak detection (including the wavelet processing), broken into dynamic and leakage energy. For each hardware/software configuration we consider two design points with 60 MHz and 250 MHz clock frequencies. We can see that the FIPM and DW3O configurations provide 4- to 5-fold reductions in the total energy required, which further underscores the benefits of the custom instructions.

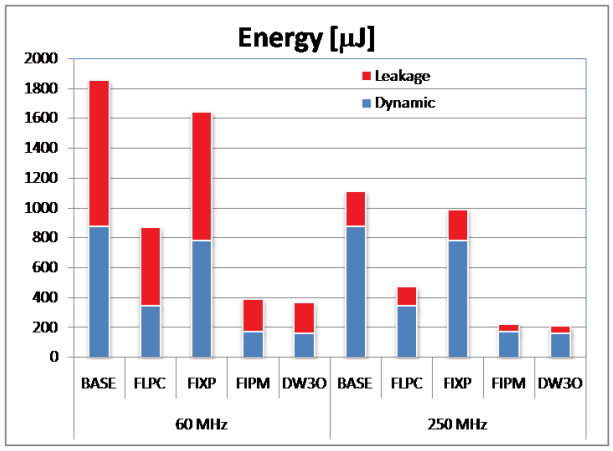

Figure 4 Energy profiles for hardware/software configurations.

TABLE II. EXECUTION TIME, CODE SIZE, AND COMPLEXITY

Name	Execution Time [cycles]		Code Size [bytes]		Complexity
	D4 wavelet	*R_peak_ detect*	*D4 wavelet*	*R_peak_ detect*	[gates]
BASE	15,889,359	18,390,999	6908	1140	22,292
FLPC	3,574,784	5,969,956	6344	1152	34,070
FIXP	13,321,202	15,705,408	5352	1104	22,292
FIPM	1,111,040	3,495,148	4984	1104	28,301
DW3O	838,656	3,222,862	3628	1104	31,097

V. CONCLUSIONS

Optimizing performance and reducing implementation cost and battery requirements is of utmost importance in engineering wearable health monitoring systems. In this paper we demonstrate how rapid customization of processor cores with application-specific instructions helps improve performance and reduces energy requirements for the ECG R-peak detection application. We believe that this approach will prove useful in optimizing processor cores used in wearable platforms for other health monitoring applications.

ACKNOWLEDGMENT

We are grateful to Tensilica for their generous software donation.

REFERENCES

[1] A. Milenkovic, C. Otto, and E. Jovanov, "Wireless sensor networks for personal health monitoring: Issues and an implementation," *Computer Communications*, vol. 29, no. 13-14, pp. 2521-2533, Aug. 2006.

[2] Chris Rowen, *Engineering the Complex SOC: Fast, Flexible Design with Configurable Processors*. Prentice Hall, 2004.

[3] "Tensilica," 23-Jun-2011. [Online]. Available: http://www.tensilica.com/. [Accessed: 23-Jun-2011].

[4] B.-U. Kohler, C. Hennig, and R. Orglmeister, "The principles of software QRS detection," *IEEE Engineering in Medicine and Biology Magazine*, vol. 21, no. 1, pp. 42-57, Feb. 2002.

[5] Awadhesh Pachauri and Manabendra Bhuyan, "Robust Detection of R-Wave Using WaveletTechnique," presented at the World Academy of Science, Engineering, and Technology 56, 2009.

[6] "PhysioNet," 24-Jun-2011. [Online]. Available: http://physionet.org/. [Accessed: 24-Jun-2011].