Adaptive Electrocardiogram Feature Extraction on Distributed Embedded Systems

Roozbeh Jafari, *Student Member*, *IEEE*, Hyduke Noshadi, *Student Member*, *IEEE*, Soheil Ghiasi, *Member*, *IEEE*, and Majid Sarrafzadeh, *Fellow*, *IEEE*

Abstract—Tiny embedded systems have not been an ideal outfit for high performance computing due to their constrained resources. Limitations in processing power, battery life, communication bandwidth, and memory constrain the applicability of existing complex medical analysis algorithms such as the Electrocardiogram (ECG) analysis. Among various limitations, battery lifetime has been a major key technological constraint. In this paper, we address the issue of partitioning such a complex algorithm while the energy consumption due to wireless transmission is minimized. ECG analysis algorithms normally consist of preprocessing, pattern recognition, and classification. Considering the orientation of the ECG leads, we devise a technique to perform preprocessing and pattern recognition locally in small embedded systems attached to the leads. The features detected in the pattern recognition phase are considered for the classification. Ideally, if the features detected for each heartbeat reside in a single processing node, the transmission will be unnecessary. Otherwise, to perform classification, the features must be gathered on a local node and, thus, the communication is inevitable. We perform such a feature grouping by modeling the problem as a hypergraph and applying partitioning schemes which yield a significant power saving in wireless communications. Furthermore, we utilize dynamic reconfiguration by software module migration. This technique, with respect to partitioning, enhances the overall power saving in such systems. Moreover, it adaptively alters the system configuration in various environments and on different patients. We evaluate the effectiveness of our proposed techniques on MIT/BIH benchmarks and, on average, achieve 70 percent energy saving.

Index Terms—Computational biology, ECG analysis, embedded systems, feature extraction.

1 Introduction

The electrocardiogram (ECG) is the record of variation of bioelectric potential with respect to time as the human heart beats. Due to its ease of use and noninvasiveness, ECG plays an important role in patient monitoring and diagnosis. Multichannel electrocardiogram (ECG) data provide cardiologists with essential information to diagnose heart disease in a patient. Our primary objective is to address the feasibility verification of implementing an ambulatory ECG analysis algorithm with real-time diagnosis functions for wearable computers. ECG analysis algorithms have always been very difficult tasks in the realization of computer-aided ECG diagnosis. Implementation of such algorithms becomes even harder for small and mobile embedded systems that should meet the given latency requirements while minimizing overall energy dissipation for the system. Distributed embedded systems are successfully deployed in various wearable computers. Distributed architectures have been developed for cooperative detection, scalable data transport, and other capabilities and services. However, the complexity of algorithms running on these systems has introduced a new set of challenges associated with resource constrained devices and their energy concerns. These obstacles may dramatically reduce the effectiveness of embedded distributed algorithms. Thus, a new distributed, embedded, computing attribute, dynamically reconfigurable, must be developed and provided to such systems. In these systems, reconfiguration capability, in particular, may be of great advantage. This capability can adaptively alter the system configuration to accommodate the objectives and meet the constraints for highly dynamic systems.

There have been exciting advances in the development of pervasive computing technologies in the past few years. Computation, storage, and communication are now more or less woven into the fabric of our society with much of the progress being due to the relentless march of Silicon-based electronics technology as predicted by Moore's Law. The emerging field of flexible electronics, where electronic components such as transistors and wires are built on a thin flexible material, offers a similar opportunity to weave computation, storage, and communication into the fabric of the very clothing that we wear, thereby creating an intelligent fabric (also called electronic textiles or e-textiles) [1]. The implications of seamlessly integrating a large number of communicating computation and storage resources, mated with sensors and actuators, in close proximity to the human body are quite exciting; for example, one can imagine biomedical applications where biometric and ambient sensors are woven into the garment of a patient to trigger and modulate the delivery of a drug. Realizing such novel applications is not just a matter of developing innovative materials for flexible electronics, along with accompanying sensors and actuators; the characteristics of the flexible electronics technology and

- *R. Jafari, H. Noshadi, and M. Sarafzadeh are with the Computer Science Department, University of California, Los Angeles, Los Angeles, CA 90095. E-mail: {rjafari, hyduke, majid}@cs.ucla.edu.*
- *S. Ghiasi is with the Department of Electrical and Computer Engineering, University of California, Davis, Davis, CA 95616. E-mail: soheil@ece.ucdavis.edu.*

Manuscript received 12 July 2005; revised 22 Feb. 2006; accepted 8 Mar. 2006; published online 26 June 2006.
Recommended for acceptance by N. Amato, S. Aluru, and D. Bader.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDSSI-0332-0705.

the requirements of the applications enabled by it necessitate radical innovation in system-level design. Electronic components built of flexible materials have characteristics that are very different from that of silicon and PCB-based electronics. Further, the operating scenarios of these systems involve environmental dynamics, physical coupling, resource constraints, infrastructure support, and robustness requirements that are distinct from those faced by traditional systems. This unique combination requires one to go beyond thinking of these systems as traditional electronic systems in a different form factor. Instead, rethinking and a complete overhaul of the system architecture and the design methodology for all layers of these systems is required.

2 Related Work

Several "wearable" technologies exist to continually monitor a patient's vital signs, utilizing low cost, well-established disposable sensors such as blood oxygen finger clips and electrocardiogram electrodes. The Smart Shirt from Sensatex [2] is a wearable health monitoring device that integrates a number of sensory devices onto the Wearable Motherboard from Georgia Tech [3]. The Wearable Motherboard is woven into an undershirt in the Smart Shirt design. Their interconnect is a flexible data bus that can support a wide array of sensory devices. These sensors can communicate via the data bus to a monitoring device located at the base of the shirt. The monitoring device is integrated into a single processing unit that also contains a transceiver. Several other technologies have been introduced such as MIThril from MIT [4], e-Textile from Carnegie Mellon University [5], Wearable e-Textile from Virginia Tech [6], and CustoMed and RFab-Vest from UCLA [7], [8]. The Lifeguard project being conducted at Stanford University is a physiological monitoring system comprised of physiological sensors (ECG/Respiration electrodes, Pulse Oximeter, Blood Pressure Monitor, Temperature probe), a wearable device with built-in accelerometers (CPOD), and a base station (Pocket PC). The CPOD acquires and logs the physiological parameters measured by the sensors [9]. The Assisted Cognition Project conducted at the University of Washington's Department of Computer Science explored the use of AI systems to support and enhance the independence and quality of life of Alzheimer's patients. Assisted Cognition systems use ubiquitous computing and artificial intelligence technology to replace some of the memory and problem-solving abilities that have been lost by an Alzheimer's patient [10]. Nevertheless, none of the above projects/systems supports the concept of scalability and adapting complex processing algorithms.

3 Automated Feature Set Detection

Given the goal of classifying objects based on their attributes, the functionality of an automated pattern recognition system can be divided into two basic tasks: The description task generates attributes of an object using feature extraction techniques, and the classification task assigns a group label to the object based on the attributes with a classifier.

There are two different approaches for implementing a pattern recognition system: statistical and structural. Each approach utilizes different schemes within the description and classification tasks which incorporates a pattern recognition system. Statistical pattern recognition [11], [12] concludes from statistical decision theory to discriminate among data from different groups based upon quantitative features of the data. The quantitative nature of statistical pattern recognition, however, makes it difficult to discriminate among groups based on the morphological (i.e., shape-based or structural) subpatterns and their interrelationships embedded within the data. This limitation provided the impetus for development of structural approaches to pattern recognition.

Structural pattern recognition [13], [14] relies on syntactic grammars to discriminate among data from different groups based upon the morphological interrelationships (or interconnections) present within the data. Structural pattern recognition systems are effective for image data as well as time-series data.

We have investigated an accurate ECG processing algorithm based on structural pattern recognition (as depicted in Fig. 1) mapped onto our processing units (dot-motes) [15]. The algorithm consists of three stages: preprocessing, pattern recognition, and classification. We perform preprocessing and pattern recognition locally, i.e., within close proximity to the ECG leads. The preprocessing includes filtering, while the pattern recognition includes heartbeat detection (through the QRS complex detection), segmentation, as well as feature extraction. Once the features are extracted, they will be processed for classification.

The filtering is performed by finite impulse response (FIR) filters with cut-off frequencies of 5-150 Hz for a sampling rate of 360 samples/sec. The heartbeat detection is implemented with a QRS detector based on the algorithm of Pan and Tompkins [16] with some improvements that employ slope information. The scheme proposed by Laguna et al. [17] is used to extract the fiducial points. All offset and onset points are detected based on the location and convexity of the R point. We detect each point onset by locating the largest isoelectric region before the point. Then, we search for the inflection point followed by largest negative slope for convex R-wave or largest positive slope for concave R-wave. We also detect the point offset by searching for significant up slope following the end of the last down slope for P, T, and S offsets in particular. Consequently, features related to heartbeat intervals and ECG morphology are calculated for each heartbeat. The list of features is included in Table 1 and are based on [18] and [19] with minor additions. In addition, a sample filtered ECG signal which was automatically segmented by our tool is depicted in Fig. 2.

We extract a total of 23 features from the ECG signals, and each derives from one of the groups below:

RR Interval Features: We extract four features based on RR Intervals. The RR interval is the interval between two successive heartbeat fiducial points, obtained from the maximum of the R-wave. The pre-RR interval is the RR-interval between a detected heartbeat and the previous one. The post-RR interval is the interval between a given heartbeat and next detected one. The average-RR interval is the average of all detected RR intervals, and the local average-RR interval is the average of the 10 most recent RR-intervals.

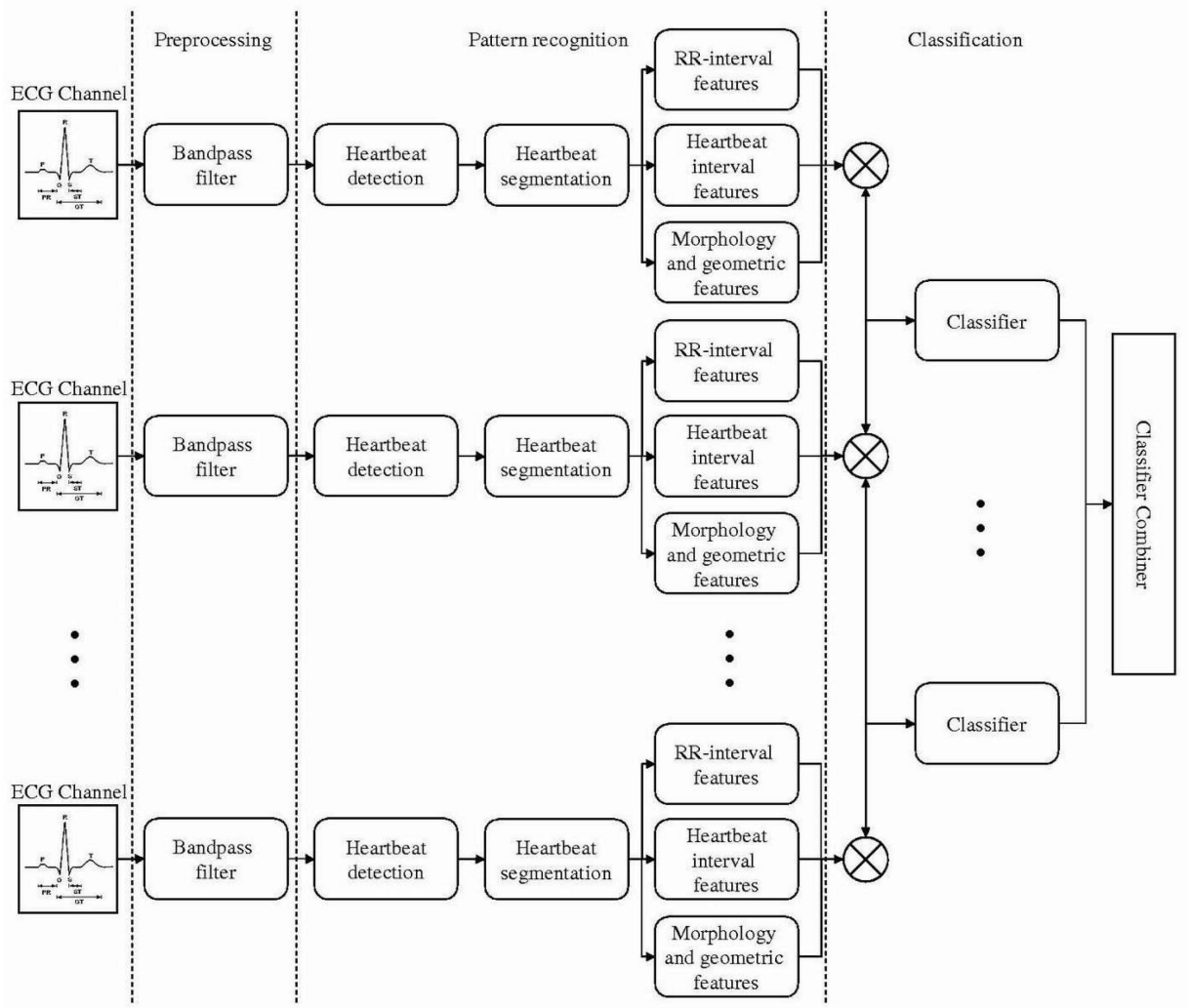

Fig. 1. ECG analysis schematic.

Heartbeat Interval Features: We extract five features related to heartbeat intervals. QRS duration is the time between QRS offset and QRS onset. T-wave duration is the time between T-wave onset and T-wave offset. The PR, ST, and QT duration are additions to the automated classification system. ST duration is the time between S-wave offset and T-wave onset. The PR duration is the time between P-wave onset and R. The QT duration is the time between Q-wave onset and T-wave offset. All of these features are obtained by first determining the start and end point of each interval, and then subtracting the end point from the start point.

Geometric Points: We calculate the signal DC shift level by taking the average base line of the previous five successive detected heartbeats. The maximal positive and the minimal negative peaks are detected by computing the voltage difference between each sample in the heartbeat and DC shift level. In addition, we extract the number of samples in a 70-100 percent range of absolute peak value. Finally, we compute the slope velocity of Q-onset-R as well as R-S segments.

ECG Morphology Features: We extract eight features based on ECG morphologies arranged into four groups. Two groups consist of samples from heartbeat segments and two groups consist of samples from fixed intervals. Within each group, one feature consists of samples from the original ECG signal, while the other feature is extracted from the normalized ECG signal. The normalization is done through scaling down the amplitude of samples by standard deviation of the same heartbeat.

We extract samples from heartbeat segments in ECG morphology 1 and 2 (see Fig. 3). In morphology 1, 10 samples between QRS onset and offset are extracted, and in morphology 2, nine samples between S-wave offset and T-wave offset are obtained. The number of samples collected is also contingent upon the sampling rate and scales with various sampling rates accordingly (the aforementioned numbers are for the original sampling rate of 360 samples per second).

We extract samples from a fixed interval in ECG morphology 3 and 4 (see Fig. 4). In morphology 3, 10 samples between $R - 50ms$ and $R + 100ms$ are extracted, and in morphology 4, eight samples between between $R + 150ms$ and $R + 500ms$ are acquired.

For all ECG morphologies, the elements that fall in between two samples are estimated using linear polarization. We have repeated such feature extraction for three input sampling rates of 360, 200, and 100 samples per second. Three hundred and sixty samples/second is the original sampling rate for the MIT/BIH [20] benchmarks

TABLE 1
Features Categorized by Groups

Group label	Features
RR Interval	Pre-RR interval
	Post-RR interval
	Average-RR interval
	Local average-RR
Heartbeat Interval	QRS duration
	T-wave duration
	PR duration
	ST duration
	QT duration
Geometric	Baseline (DC level)
	Number of samples in 70%-100% of R-amplitude
	Maximum positive peak
	Maximum negative peak
	Slope velocity between Q-onset and R
	Slope velocity between R and S
Morphology 1A	ECG morphology (10 samples for sampling rate: 360 s/sec) between Q and S
Morphology 1B	Normalized ECG morphology (10 samples for sampling rate: 360 s/sec) between Q and S
Morphology 2A	ECG morphology (9 samples for sampling rate: 360 s/sec) between S and T-wave offset
Morphology 2B	Normalized ECG morphology (9 samples for sampling rate: 360 s/sec) between S and T-wave offset
Morphology 3A	ECG morphology (10 samples for sampling rate: 360 s/sec) between R-50 ms and R+100 ms
Morphology 3B	Normalized ECG morphology (10 samples for sampling rate: 360 s/sec) between R-50 ms and R+100 ms
Morphology 4A	ECG morphology (8 samples for sampling rate: 360 s/sec) between S + 150 ms and S + 500ms
Morphology 4B	Normalized ECG morphology (8 samples for sampling rate: 360 s/sec) between S + 150 ms and S + 500ms

and the sampling rates of 200 and 100 samples/second was acquired by downsampling the input.

Despite our objective is to minimize the communication among processing nodes before the classification phase, this study does not investigate the problem of classification. Therefore, we did not implement a classifier for our platform. However, any classifier suitable for constrained embedded systems may be deployed.

4 Software Profiling

To measure the execution delay of our heartbeat detection and feature extraction program, we used Avrora [21], a microcontroller simulator framework developed at the University of California, Los Angeles. Avrora is a precise and flexible simulator that preserves all timing and behavior of the instrumented program, while allowing

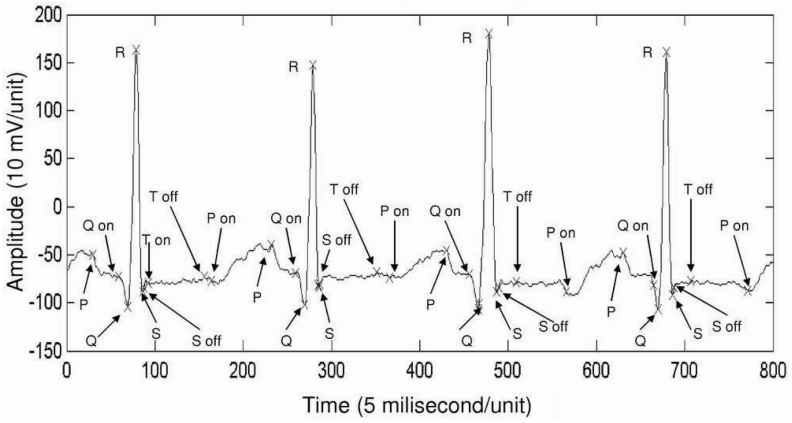

Fig. 2. Automatic ECG segmentation performed on filtered signal.

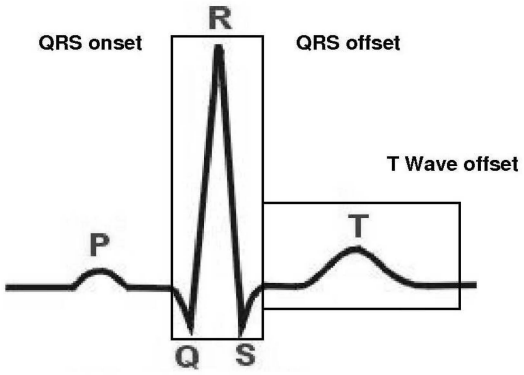

Fig. 3. The sampling intervals of ECG morphologies 1 and 2. Morphology 1 consists of samples extracted from QRS onset and offset. Morphology 2 consists of samples from S-wave offset and T-wave offset.

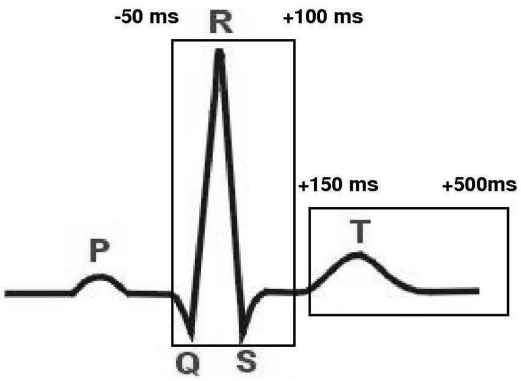

Fig. 4. The sampling intervals of ECG morphologies 3 and 4. Morphology 3 consists of samples between 50 ms before and after the fiducial point (FP). Morphology 4 consists of samples between 150 ms and 500 ms after the fiducial point.

user-defined profiling of application information. With Avrora, users can easily profile application-specific information such as branch frequency, maximum stack size, and memory access by adding custom program monitors.

For our experiments, we implemented a program monitor on Avrora that generates the control flow graph (CFG) while measuring the execution frequency and delay of each basic block. Since the CFG of our system is very large, only the major processes are shown in Fig. 5. The CFG is dynamically generated based upon our compiled and assembled ECG program, while Avrora simulates the program execution. Unlike static analysis, parts of a program that are not executed during the simulation will not be accounted for. Also, the generated graph will accurately reflect compiler optimizations. Hardware interrupts, which occur intermittently during execution, are accounted for as well.

Delay analysis for each function is performed during CFG generation for practicality since basic block information may be too detailed. Function delay is measured as the duration when execution enters a function to when execution exits. Calls to other functions are accounted for, while interrupts are not. Since execution delay may be inconsistent due to functions containing different execution paths, the average function delay is gathered from each execution instance. However, for our purposes, the functions that extract features from heartbeat signals all consist of a single execution path. Therefore, we lost no precision in our analysis. The delays of feature detection modules for sampling rate of 360 samples/second are illustrated in Fig. 6.

5 Target Architecture Model

Networked sensor nodes containing constrained, often battery-powered, embedded computers can densely sample phenomena that were previously difficult or costly to observe. Sensor nodes can be placed anywhere on a patients' body. Due to the mobility of such systems, wireless sensor networks are expected to be both autonomous and long-lived, surviving environmental hardships while conserving energy as much as possible.

It is well-known that the amount of energy consumed for a single wireless communication of one bit can be many orders of magnitude greater than the energy required for a single local computation [22]. Thus, we focus on the energy used for wireless communication. In our model, since all nodes are placed within close proximity of each other, we assume they communicate directly and multihop communication is not required. Therefore, the total energy consumed for in-network processing is:

$$\varepsilon(n) = b(n) \times e(n), \tag{1}$$

where $b(n)$ is the number of packets transmitted and $e(n)$ is the average amount of energy required to transmit one packet. In our design, we consider the Collision Free Model (CFM), which simplifies the programming by abstracting out all of the details of low level channel contention and packet collision from the algorithm designers. By abstracting reliable communication as an atomic operation, programming based on CFM bears a resemblance to existing algorithm design in parallel and distributed computation. CFM does not really capture the impact of packet collision that distinguishes wireless communication from wired communication, which makes performance analysis under CFM not very accurate. However, for the sake of simplicity, we consider CFM in our design.

6 Dynamic Reconfiguration

Sensor nodes are composed of embedded systems as well as general-purpose software, introducing a tension between resource and energy constraints and the layers of indirection required to support true general-purpose operating systems. TinyOS [23], the state-of-the-art sensor operating system, tends to prioritize embedded system constraints over general-purpose OS functionality. TinyOS consists of a collection of software components written in the NesC language [24], ranging from low-level parts of the network stack to application-level routing logic. Our target operating system, SOS, is a new operating system for mote-class sensor nodes that takes a more dynamic point on the design spectrum [25]. SOS consists of dynamically-loaded modules and a common kernel, which implements messaging, dynamic memory, and module loading and unloading, among other services. Dynamic reconfigurability is one of our primary assumptions. In the domain of embedded computing, reconfigurability is the ability to modify the software on individual nodes of a network after the network has been deployed and initialized. This provides the ability to incrementally update the sensor network after it is deployed, add new software modules, and remove unused software modules when they are no longer needed.

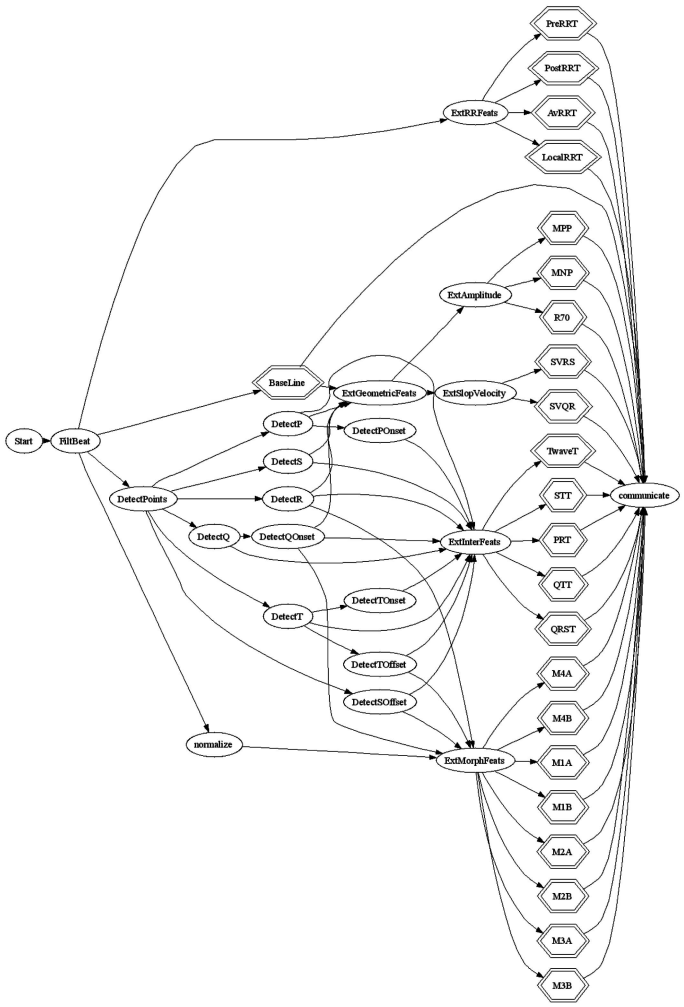

Fig. 5. Graph generated from profiling analysis containing only major blocks required for feature extraction.

The growing tensions between large, hard to update networks and complex applications with incremental patches has made reconfigurability an issue that can no longer be ignored. SOS supports a mechanism that enables over the air reprogramming of the sensor nodes. Using this method, software modules may be modified, added, or removed.

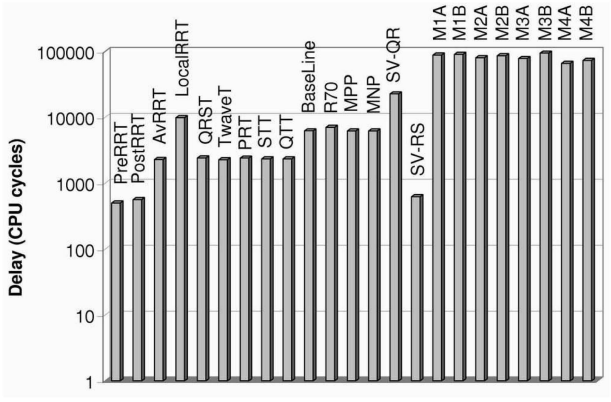

Fig. 6. Delays corresponding to ECG feature detection modules extracted in profiling phase.

7 Feature Set Partitioning

A hypergraph is a generalization of a graph, where the set of edges is replaced by a set of hyperedges. A hyperedge extends the notion of an edge by allowing more than two vertices to be connected by a hyperedge. Formally, a hypergraph $H = (V, E^h)$ is defined as a set of vertices V and a set of hyperedges E^h, where each hyperedge is a subset of the vertex set V [26], and the size a hyperedge is the cardinality of this subset. Let w_i denote the weight of vertex $v_i \in V$. A K-way vertex partition $\Pi = \{V_1, V_2, \ldots, V_k\}$ of H is said to be balanced with an overall load imbalance tolerance $\epsilon \ll 1$ if each V_i satisfies the following equation:

$$W_k \leq W_{avg}(1+\epsilon), \text{ for } \quad k = 1, 2, \ldots, K, \tag{2}$$

where

$$W_k = \sum_{v_i \in V_k} w_i \tag{3}$$

$$W_{avg} = \left(\sum_{v_i \in V} w_i \right) / K. \tag{4}$$

In a partition of H, a hyperedge that has at least one vertex in a partition is said to *connect* that partition. Connectivity set Λ_j of a hyperedge e_j is defined as the set of

TABLE 2
Benchmark Statistics

Record #	Channel used	Patient	Medication
100	MLII	male, age 69	Aldomet, Inderal
101	MLII	female, age 75	Diapres
102	V5	female, age 84	Digoxin
103	MLII	male, age not recorded	Diapres, Xyloprim
104	V5	female, age 66	Digoxin, Pronestyl
105	MLII	female, age 73	Digoxin, Nitropaste, Pronestyl
106	MLII	female, age 24	Inderal
107	MLII	male, age 63	Digoxin
108	MLII	female, age 87	Digoxin, Quinaglute
109	MLII	male, age 64	Quinidine
111	MLII	female, age 47	Digoxin, Lasix
112	MLII	male, age 54	Digoxin, Pronestyl
113	MLII	female, age 24	None
114	V5	female, age 72	Digoxin
115	MLII	female, age 39	None
116	MLII	male, age 68	None
117	MLII	male, age 69	None
118	MLII	male, age 69	Digoxin, Norpace
119	MLII	female, age 51	Pronestyl
121	MLII	female, age 83	Digoxin, Isordil, Nitropaste
122	MLII	male, age 51	Digoxin, Lasix, Pronestyl
123	MLII	female, age 63	Digoxin, Inderal
124	MLII	male, age 77	Digoxin, Isordil, Quinidine
200	MLII	male, age 64	Digoxin, Quinidine
201	MLII	male, age 68	Digoxin, Hydrochlorthiazide, Inderal, KCl
202	MLII	male, age 68	Digoxin, Hydrochlorthiazide, Inderal, KCl
203	MLII	male, age 43	Coumadin, Digoxin, Heparin, Hygroton, Lasix
205	MLII	male, age 59	Digoxin, Quinaglute
207	MLII	female, age 89	Digoxin, Quinaglute
208	MLII	female, age 23	None
209	MLII	male, age 62	Aldomet, Hydrodiuril, Inderal
210	MLII	male, age 89	None
212	MLII	female, age 32	None
213	MLII	male, age 61	Digoxin
214	MLII	male, age 53	Digoxin, Dilantin
215	MLII	male, age 81	None
217	MLII	male, age 65	Digoxin, Lasix, Quinidine
219	MLII	male, age not recorded	Digoxin
220	MLII	female, age 87	Digoxin
221	MLII	male, age 83	Hydrochlorthiazide, Lasix
222	MLII	female, age 84	Digoxin, Quinidine
223	MLII	male, age 73	None
228	MLII	female, age 80	Digoxin, Norpace
230	MLII	male, age 32	Dilantin
231	MLII	female, age 72	None
232	MLII	female, age 76	Aldomet, Inderal
233	MLII	male, age 57	Dilantin
234	MLII	female, age 56	None

partitions connected by e_j. Connectivity $\lambda_j = |\Lambda_j|$ of a hyperedge e_j denotes the number of partitions connected by e_j. A hyperedge h_j is said to be cut (external) if it connects more than one partition (i.e., $\lambda_j > 1$), and uncut (internal) otherwise (i.e., $\lambda_j = 1$). Therefore, the definition of cut-size is as follows:

$$\text{cutsize}(\Pi) = \sum_{e_j \in E^h} (\lambda_j - 1). \quad (5)$$

Hence, the cutsize is equal to the number of cut nets. The hypergraph partitioning is defined as dividing it into two or more parts such that the cutsize is minimized, while a given balance criterion among the partition weights is achieved. The hypergraph partitioning problem is known to be NP-hard [27].

During the software partitioning, it is quite important to be able to divide the system specification into clusters so that the intercluster (intermote) connections are minimized. Hypergraphs can be used to naturally represent feature extraction

TABLE 3
Number of Queries Exchanged among Processing Units: Sampling Rate = 360 Sample/Sec

Record #	# of queries exchanged with adaptive partitioning	# of queries exchanged with fixed configuration	Transmission power saving with adaptive partitioning (%)
100	10	10	0.00
101	76	154	50.65
102	121	157	22.93
103	53	96	44.79
104	37	281	86.83
105	187	258	27.52
106	35	186	81.18
107	64	162	60.49
108	112	302	62.91
109	58	490	88.16
111	144	378	61.90
112	63	218	71.10
113	63	100	37.00
114	4	57	92.98
115	19	70	72.86
116	52	457	88.62
117	53	78	32.05
118	26	169	84.62
119	13	204	93.63
121	86	179	51.96
122	32	160	80.00
123	5	21	76.19
124	49	183	73.22
200	17	48	64.58
201	2	217	99.08
202	41	71	42.25
203	19	233	91.85
205	83	170	51.18
207	232	547	57.59
208	20	292	93.15
209	225	272	17.28
210	143	183	21.86
212	7	119	94.12
213	63	552	88.59
214	145	177	18.08
215	181	603	69.98
217	117	237	50.63
219	35	160	78.13
220	74	155	52.26
221	23	244	90.57
222	6	170	96.47
223	144	328	56.10
228	54	468	88.46
230	15	27	44.44
231	4	16	75.00
232	13	189	93.12
233	47	328	85.67
234	73	192	61.98
Average	65.52	216.00	69.67

TABLE 4
Number of Queries Exchanged among Processing Units: Sampling Rate = 200 Sample/Sec

Record #	# of queries exchanged with adaptive partitioning	# of queries exchanged with fixed configuration	Transmission power saving with adaptive partitioning (%)
100	5	5	0.00
101	58	84	30.95
102	173	276	37.32
103	58	170	65.88
104	49	188	73.94
105	169	252	32.94
106	41	343	88.05
107	68	229	70.31
108	101	314	67.83
109	62	478	87.03
111	194	347	44.09
112	56	114	50.88
113	26	156	83.33
114	17	141	87.94
115	48	265	81.89
116	154	361	57.34
117	51	69	26.09
118	63	273	76.92
119	2	237	99.16
121	88	255	65.49
122	12	127	90.55
123	15	105	85.71
124	61	223	72.65
200	1	222	99.55
201	206	296	30.41
202	35	194	81.96
203	132	495	73.33
205	78	222	64.86
207	201	497	59.56
208	32	319	89.97
209	244	387	36.95
210	163	339	51.92
212	37	312	88.14
213	123	207	40.58
214	153	187	18.18
215	136	180	24.44
217	25	160	84.38
219	68	286	76.22
220	42	160	73.75
221	85	443	80.81
222	12	183	93.44
223	117	528	77.84
228	62	433	85.68
230	16	157	89.81
231	34	177	80.79
232	46	217	78.80
233	125	529	76.37
234	72	271	73.43
Average	79.50	258.60	69.26

algorithms. The vertices of the hypergraph are modeled as features, their weights represent the computational time required for features detection, and the hyperedges resemble the number of times a set of features is triggered simultaneously. Partitioning the graph such that the cut-size is minimized while the partitions are balanced can reduce the communication that is required among various processing units for classification phase. The vision is that all features selected must be classified at a local node, thus, in the events where selected features reside on distributed nodes, internode communication is inevitable. A high quality hypergraph partitioning algorithm greatly affects the feasibility, quality, and the cost of the resulting system.

We employed a hypergraph partitioning algorithm that is based on the multilevel paradigm. In the multilevel paradigm, a sequence of successively coarser hypergraphs is constructed. A bisection of the smallest hypergraph is computed and used to obtain a bisection of the original hypergraph by successively projecting and refining the bisection to the next level finer hypergraph. We have used hMETIS, a program for partitioning hypergraphs implemented for PCs [28]. The same algorithm can be easily ported on a mobile computer such as a Pocket PC to facilitate dynamic reconfiguration. The vision is that the hypergraph information is collected real-time from the processing nodes of the wearable computer. Subsequently, the algorithm running on the motes are reconfigured. The number of partitions is determined as described below:

The preprocessing tasks, as well as pattern recognition, must be completed before the next heartbeat arrives. Let the heartbeat be N beats per minute (bpm). Therefore, the heartbeat rate period can be obtained from:

$$T_{heartbeat} = 60/N. \tag{6}$$

Let the time required for preprocessing and pattern recognition be t_{pre} and t_{recog}, respectively.

$$t_{pre} + t_{recog} < \delta \times (T_{heartbeat}), \quad \text{where} \quad \delta < 1. \tag{7}$$

The factor δ is selected to be 0.9 to ensure a margin that prevents overloading the processing units. Therefore, the maximum CPU time that may be assigned to pattern recognition is $\delta \times (T_{heartbeat}) - t_{pre}$, where t_{pre} is fixed and can be computed from the profiling stage. As described earlier, the weight on vertices represents the required computational time for each feature. In addition, W_k is already outlined in (4). Therefore, the following objective should be accommodated:

$$\begin{gathered} \text{Minimize} \quad K \\ s.t. \\ W_k < t_{recog} \quad \forall k = 1..K. \end{gathered} \tag{8}$$

To determine the value of K, we consider the total time required for pattern recognition on all features, T_{recog} (extracted from profiling analysis). It is trivial that the lowerbound on K can be obtained from the following equation:

$$K = T_{recog}/t_{recog}. \tag{9}$$

Once partitioning is performed based on the value of K, the solution may be imbalanced and violates the constraint described in (8). In this case, K must be incremented and the features are repartitioned until a feasible solution is determined.

8 Simulation Analysis

This section presents various simulation analysis performed to exhibit the effectiveness of our technique. All experiments were carried out with ECG signals from MIT-BIH Arrhythmia database. The MIT-BIH Arrhythmia database contains 48 half-hour excerpts of two-channel ambulatory ECG recordings, obtained from 47 subjects studied by the BIH Arrhythmia Laboratory between 1975 and 1979. The recordings were digitized at 360 samples per second per channel with 11-bit resolution over a 10 mV range. We used all 48 complete records freely available from PhysioNet [29]. We also repeated the experiments by downsampling all the benchmarks to 200 and 100 samples per second. As illustrated in Table 2, each MIT-BIH record has the recordings of two channels. Yet, we only used the first channel. The second channel was not used for the sake of simplicity. Originally, in MIT/BIH benchmarks, the electrodes placed on the chest were selected due to their small noise level.

We performed profiling analysis on the algorithm described in Section 3 using Avrora to compute the computational delay of feature detection modules. The ECG algorithm was ported both for dot-motes (SOS) and PCs. The algorithm for PC was written in C language. The simulation for feature and hypergraph extraction was done on PC due to a number of software instability that we encountered in SOS. As for hypergraph partitioning, we utilized hMETIS. The MIT/BIH benchmarks were used with three sampling rates as illustrated in Tables 3, 4, and 5. The original sampling rate was 360 samples/sec while 200 and 100 samples/sec were acquired by downsampling the data. In Tables 3, 4, and 5, two scenarios for configuration were considered. In one scenario, features were adaptively assigned to processing units based on hypergraph partitioning (adaptive partitioning). In the other scenario, the optimized configuration was determined using hypergraph partitioning on benchmark 100 and remained fixed throughout our experiments (fixed configuration). The number of partitions were obtained from (9) for each benchmark. Table 3 figures the number of queries exchanged in both scenarios. Considering that the experiments were carried out through simulations, we were unable to measure the wireless power consumption. However, given the number of features we examined—23, each query may be incorporated in a wireless packet of dot-motes (30 bytes). Therefore, taking into account (1), the wireless power consumption is proportional to the number of queries exchanged. On average, the communication energy consumption was reduced by approximately 70 percent in all sets of experiments. The wireless communication overhead for partitioning was negligible due to the small size, sparsity, and slowly changing nature of our hypergraphs. The reconfiguration was performed only once for each benchmark. Therefore, its effect on the performance of the system was negligible.

TABLE 5
Number of Queries Exchanged among Processing Units: Sampling Rate = 100 Sample/Sec

Record #	# of queries exchanged with adaptive partitioning	# of queries exchanged with fixed configuration	Transmission power saving with adaptive partitioning (%)
100	11	11	0.00
101	71	152	53.29
102	101	267	62.17
103	45	86	47.67
104	15	186	91.94
105	196	594	67.00
106	48	296	83.78
107	78	181	56.91
108	47	188	75.00
109	30	414	92.75
111	94	354	73.45
112	31	242	87.19
113	38	149	74.50
114	43	120	64.17
115	21	70	70.00
116	15	284	94.72
117	25	103	75.73
118	28	185	84.86
119	26	169	84.62
121	70	257	72.76
122	37	407	90.91
123	13	136	90.44
124	60	147	59.18
200	14	181	92.27
201	128	250	48.80
202	45	231	80.52
203	104	272	61.76
205	85	109	22.02
207	311	557	44.17
208	25	253	90.12
209	227	623	63.56
210	168	291	42.27
212	107	365	70.68
213	51	598	91.47
214	113	183	38.25
215	300	532	43.61
217	6	174	96.55
219	29	374	92.25
220	58	95	38.95
221	12	242	95.04
222	11	98	88.78
223	154	331	53.47
228	217	494	56.07
230	71	287	75.26
231	14	185	92.43
232	24	301	92.03
233	139	425	67.29
234	61	178	65.73
Average	75.35	263.06	71.36

9 Conclusion

We proposed a technique for software partitioning in tiny embedded systems. Our target application was an ECG analysis algorithm which is generally classified as a complex medical application. We addressed the problem of mapping such an application onto resource constrained embedded systems while extending the lifetime of the system. This was achieved by reducing the energy consumption due to the wireless communications. We demonstrated the effectiveness of our technique on various ECG excerpts from MIT/BIH benchmarks. On average, the energy consumption rate was reduced by 70 percent.

References

[1] D. Meoli and T. May-Plumlee, "Interactive Electronic Textile Development: A Review of Technologies," *J. Textile and Apparel, Technology and Management,* vol. 2, no. 2, 2002.

[2] Sensatex, http://www.sensatex.com, 2006.

[3] S. Park, K. Mackenzie, and S. Jayaraman, "The Wearable Motherboard: A Framework for Personalized Mobile Information Processing (PMIP)," *Proc. 39th Design Automation Conf.,* pp. 170-174, 2002.

[4] R. DeVaul, J.G.M. Sung, and A. Pentland, "Mithril 2003: Applications and Architecture," *Wearable Computers, Proc. Seventh IEEE Int'l Symp.,* pp. 4-11, 2003.

[5] D. Marculescu, R. Marculescu, and P. Khosla, "Challenges and Opportunities in Electronic Textiles Modeling and Optimization," *Proc. 39th Design Automation Conf.,* pp. 175-180, 2002.

[6] T. Martin, M. Jones, J. Edmison, and R. Shenoy, "Towards a Design Framework for Wearable Electronic Textiles," *Wearable Computers, Proc. Seventh IEEE Int'l Symp.,* pp. 190-199, 2003.

[7] R. Jafari, A. Encarnacao, A. Zahoory, F. Dabiri, H. Noshadi, and M. Sarrafzadeh, "Wireless Sensor Networks for Health Monitoring," *MobiQuitous '05: Proc. Second Ann. Int'l Conf. Mobile and Ubiquitous Systems,* 2005.

[8] R. Jafari, F. Dabiri, P. Brisk, and M. Sarrafzadeh, "Adaptive and Fault Tolerant Medical Vest for Life-Critical Medical Monitoring," *SAC '05: Proc. 2005 ACM Symp. Applied Computing,* pp. 272-279, 2005.

[9] Lifeguard Monitoring System, http://lifeguard.stanford.edu, 2006.

[10] H. Kautz, O. Etzioni, D. Fox, and D. Weld, "Foundations of Assisted Cognition Systems," technical report, Univ. of Washington, Computer Science Dept., 2003.

[11] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification,* second ed., John Wiley and Sons, Inc., Jan. 2000.

[12] A.K. Jain, R.P.W. Duin, and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 1, pp. 4-37, 2000.

[13] A. Graps, "An Introduction to Wavelets," *IEEE Computational Sciences and Eng.,* vol. 2, no. 2, pp. 50-61, 1995.

[14] T. Pavlidis, *Structural Pattern Recognition,* series in electrophysics, Springer-Verlag, vol. 1, 1977.

[15] Crossbow Technology Inc., http://www.xbow.com, 2006.

[16] J. Pan and W.J. Tompkins, "A Real-Time QRS Detection Algorithm," *IEEE Trans. Biomedical Eng.,* vol. 32, no. 3, pp. 230-236, 1985.

[17] P. Laguna, R.G. Mark, A. Goldberger, and G.B. Moody, "A Database for Evaluation of Algorithms for Measurement of QT and Other Waveform Intervals in the ECG," pp. 673-676, 1997.

[18] P. de Chazal, M. O'Dwyer, and R.B. Reilly, "Automatic Classification of Heartbeats Using ECG Morphology and Heartbeat Interval Features," *IEEE Trans. Biomedical Eng.,* vol. 51, no. 7, pp. 1196-1206, 2004.

[19] I. Christov and G. Bortolan, "Ranking of Pattern Recognition Parameters for Premature Ventricular Contractions Classification by Neural Networks," *Physiological Measurement,* vol. 25, no. 5, pp. 1281-1290, 2004.

[20] G.B. Moody and R.G. Mark, "The MIT-BIH Arrhythmia Database on CD-ROM and Software for Use with It," *Computers in Cardiology,* pp. 185-188, 1990.

[21] B.L. Titzer, D. Lee, and J. Palsberg, "Avrora: Scalable Sensor Network Simulation with Precise Timing," *IPSN '05, Proc. Fourth Int'l Conf. Information Processing in Sensor Networks,* 2005.

[22] V. Shnayder, M. Hempstead, B. Rong Chen, G.W. Allen, and M. Welsh, "Simulating the Power Consumption of Large-Scale Sensor Network Applications," *SenSys '04: Proc. Second Int'l Conf. Embedded Networked Sensor Systems,* pp. 188-200, 2004.

[23] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler, "The Emergence of Networking Abstractions and Techniques in Tiny OS," *Proc. First Symp. Networked System Design and Implementation (NSDI '04),* pp. 1-14, 2004.

[24] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The NESC Language: A Holistic Approach to Networked Embedded Systems," *PLDI '03: Proc. ACM SIGPLAN 2003 Conf. Programming Language Design and Implementation,* pp. 1-11, 2003.

[25] C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava, "A Dynamic Operating System for Sensor Nodes," *MobiSys '05: Proc. Third Int'l Conf. Mobile Systems, Applications, and Services,* pp. 163-176, 2005.

[26] S. Dutt and W. Deng, "A Probability-Based Approach to VLSI Circuit Partitioning," *DAC '96: Proc. 33rd Ann. Conf. Design Automation,* pp. 100-105, 1996.

[27] M.R. Garey and D.S. Johnson, *Computers and Instractability: A Guide to the Theory of NP-Completeness.* San Francisco, Calif.: W.H. Freeman, 1979.

[28] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel Hypergraph Partitioning: Application in VLSI Domain," *DAC '97: Proc. 34th Ann. Conf. Design Automation,* pp. 526-529, 1997.

[29] "Physiobank—Physiologic Signal Archives for Biomedical Research," http://www.physionet.org/physiobank/, 2006.

Roozbeh Jafari received the BSc degree in electrical engineering in 2000 from the Sharif University of Technology, Tehran, Iran. He received the MSc degree from the State University of New York at Buffalo in electrical engineering in 2002. He then joined the University of California, Los Angeles, where he received the MS degree in computer science in 2004 and he is currently pursuing his PhD degree in computer science. He is mainly interested in embedded system design and analysis and medical and biological applications. He is a student member of the IEEE.

Hyduke Noshadi received the BSc degree in computer science from the University of California, Los Angeles (UCLA) in 2006. He is currently working toward the MSc degree in computer science. His research interest is in embedded system design and analysis. He is a student member of the IEEE.

Soheil Ghiasi recieved the BS degree from the Sharif University of Technology, Tehran, Iran, in 1998, and the MS and PhD degrees in computer science from the University of California, Los Angeles (UCLA) in 2002 and 2004, respectively. He received the Harry M. Showman prize from the UCLA College of Engineering in 2004. Currently, he is an assistant professor in the Department of Electrical and Computer Engineering at the University of California, Davis. His research interests include different aspects of embedded and reconfigurable system design. He is a member of the IEEE.

Majid Sarrafzadeh (M'87, SM'92, F'96) (http://www.cs.ucla.edu/majid) received the BS, MS, and PhD degrees in 1982, 1984, and 1987, respectively, from the University of Illinois at Urbana-Champaign in electrical and computer engineering. He joined Northwestern University as an assistant professor in 1987. In 2000, he joined the Computer Science Department at the University of California at Los Angeles (UCLA). His recent research interests lie in the area of embedded and reconfigurable computing, VLSI CAD, and design and analysis of algorithms. Dr. Sarrafzadeh is a fellow of the IEEE for his contribution to "Theory and Practice of VLSI Design." He is also a member of the IEEE Computer Society. He received a US National Science Foundation Engineering Initiation award, two distinguished paper awards in ICCAD, and the best paper award in DAC. He has served on the technical program committee of numerous conferences in the area of VLSI Design and CAD, including ICCAD, DAC, EDAC, ISPD, FPGA, and DesignCon. He has served as a committee chair of a number of these conferences. He is on the executive committee/steering committee of several conferences such as ICCAD, ISPD, and ISQED. Professor Sarrafzadeh has published approximately 250 papers, is a coeditor of the book *Algorithmic Aspects of VLSI Layout* (World Scientific, 1994), and coauthor of the books *An Introduction to VLSI Physical Design* (McGraw Hill, 1996) and *Modern Placement Techniques* (Kluwer, 2003). Dr. Sarrafzadeh is on the editorial board of the *VLSI Design Journal*, an associate editor of *ACM Transactions on Design Automation* (TODAES), and an associate editor of the *IEEE Transactions on Computer-Aided Design* (TCAD).