Low-Cost Real-Time 2-D Motion Detection Based on Reconfigurable Computing

Kyprianos Papadimitriou, *Student Member, IEEE,* Apostolos Dollas, *Senior Member, IEEE,* and Stamatios N. Sotiropoulos

***Abstract*—This paper presents a method for real-time hand motion detection in two-dimensional (2-D) space. A new input device for kinetically challenged persons that uses this method is presented. The device consists of a solid-state accelerometer that senses 2-D motion, a microcontroller that samples the data in real time, and an embedded field-programmable gate array (FPGA) device that distinguishes the types of motion from programmable motion vocabularies. The system has a quadratic capability $O(n^2)$ in detecting motions, while the hardware used has a linear-complexity $O(n)$. The motion-detection computational model is presented, along with experimental results. The system adaptation to individual requirements and the cost versus quality tradeoff can be addressed through reconfiguration.**

***Index Terms*—Acceleration, kinetically challenged, real time, reconfigurable embedded system, two-dimensional (2-D) motion detection.**

I. Introduction

THE PROBLEM of motion detection and recognition has been considered from a number of perspectives, ranging from I/O for virtual reality environments [1] to gesture recognition systems [2]. Various alternatives, regarding hand-motion measurement, have been proposed. Among them, goniometric-, optoelectric-, and accelerometric-movement monitoring systems exist. Similarly, the problem of I/O assistive devices for kinetically challenged persons (i.e., persons who cannot control voluntary movement of more than one part of their body due to neurological disorders) has been addressed from a mechanical design perspective [3]–[5] to a brain-activity-detection perspective [6]. Many of the assistive devices are related to motion detection of disabled persons [7]. Specific movements of parts of the human body generate commands that control external devices (e.g., wheelchair, PC, and telephone).

A human–computer interface with emphasis on commands expressed as hand gestures is presented in [2]. A camera tracks hand motions and interprets them to user commands. Locomotive variables are extracted in [8] by a pair of accelerometers and a tilt sensor mounted in the sole of a shoe. A monitoring system for elderly persons used for detection of fall uses accelerometers, as well [9], in combination with physiological parameters, such as pulse, arterial pressure, breathing rate, or temperature.

More recent systems utilize the potential provided by reconfigurable computing [10] and target to assistance of kinetically challenged persons. Field-programmable-gate array (FPGA)-based systems are now widely used in almost any area of application. Being custom-computing machines, FPGA-based systems can offer high performance, outperforming any other programmable solution [11]. Moreover, only few applications deserve the expense of creating application-specific solutions. An FPGA-based system, which can be reprogrammed like a standard workstation, may offer the highest performance for many different applications.

In [12], a wearable accelerometric-motion analysis system was developed. The system consists of an Altera Flex10KE FPGA, digital accelerometers, and an embedded computer system with an adaptive-system Pentium-class processor module for further processing. The system is used to assess balance and mobility impairments for monitoring patients' progress.

A computer-interface device for handicapped people, which uses an FPGA, is presented in [13]. The movement of the head, on which an optical sensor is mounted, is used as a positioning signal of the computer cursor.

An adaptive integral system for assisted mobility has been developed using various sensors, FPGAs, and DSPs [14]. A wheelchair is controlled by various electronic-guidance alternatives, which include head movements, voice commands, electro-oculography commanding, digital joystick, and breath expulsion. FPGA and accelerometers are also used in a project called "3D Eye Tracking Device" [15], which aims to measure head and eye movements. The FPGA acquires real-time-sensor data, which are then processed by a computer for medical purposes.

In this paper, we present a low-cost FPGA-based embedded-I/O device for kinetically challenged persons. We target to shrink-wrapped hardware, which can be customized and retrained to individual needs without a recompilation of the design, but through reconfiguration, as needed. All of the above projects use large/multiple-FPGAs, PC-class fixed-computer resources, or expensive equipment to perform data acquisition and calculations. In our system, such approaches would not meet cost, size, and power consumption limits for an embedded application.

Previous publications on this project have focused on the reconfigurable computing for real-time motion detection and general system description [16], an improved model for motion

Manuscript received October 24, 2003; revised August 16, 2006. This work was supported in part by the Greek Secretariat of Research and Technology and the British Council, under the Britain Greece Joint Research Program and the EPET II European Union program under the Second Framework of Support to Greece.

The authors are with the Microprocessor and Hardware Laboratory, Electronic and Computer Engineering Department, Technical University of Crete, Chania, Greece (e-mail: kpapadim@mhl.tuc.gr; dollas@mhl.tuc.gr; sotirop@mhl.tuc.gr).

Digital Object Identifier 10.1109/TIM.2006.884280

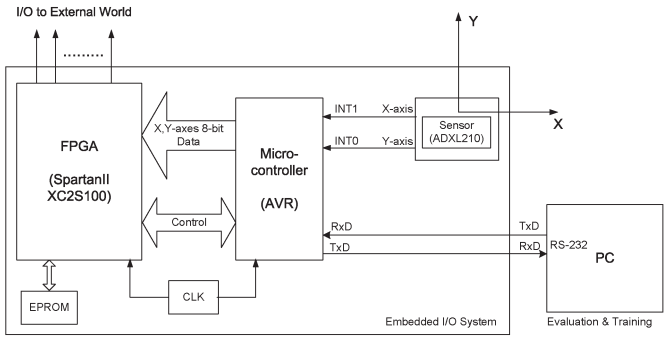

Fig. 1. Architecture of the embedded system. The system can be connected with a PC for evaluation and training.

detection [17], and an unsupervised training algorithm that makes the system adaptive to individual needs [18]. The latest version of the system as a whole improved in terms of cost, and potential from the instrumentation perspective, including issues of preprocessing and calibration, is presented in this paper. More specifically, the contributions of this paper are: 1) presentation of the preprocessing of sampled data in a way that optimizes the subsequent processing via reconfigurable computing; 2) system calibration issues, including sensor data variation and gravity compensation; 3) analysis of the two motion-detection models, along with hardware complexity/cost tradeoffs; and 4) the effect of system training in hardware resources, complexity, and cost.

Section II presents the architecture and applications of the system, and Section III has the real-time processing of the sensor data. Section IV presents the tradeoff-space evaluation. It also discusses the need of system adaptation to the user. Section V presents the real-time motion-detection model and compares two different versions of the model. Section VI includes model validation and performance results from persons with and without motor disabilities. Finally, Section VII presents the conclusion from this work as well as its future directions.

II. Architecture and Applications of the System

The architecture of the system is shown in Fig. 1. Briefly, the system consists of

1) solid-state dual-axis accelerometer placed on the user's hand that sense motion in space;
2) an 8-bit microcontroller that samples the sensor data in real time and converts them to acceleration values;
3) an embedded FPGA that receives accelerations from the microcontroller and processes them to detect a motion from a programmable vocabulary of motions;
4) a Programmable Read-Only Memory (PROM) for on-board programming of the FPGA.

A key decision for the system was the usage of solid-state accelerometers. The choice of accelerometers as input devices opened a slew of design considerations. We chose two axes accelerometer instead of three axes as we determined that the inclusion of a third axis complicated unnecessarily the computational requirements [16]. Moreover, the digital nature of the sensor output signal provided us the ability to handle it in an easy and efficient way. Although other sensors were considered (e.g., Hall-effect sensors), accelerometers were considered sufficiently small in size, reliable, and low cost. However, the system can be modified to accept different types of sensors.

Regarding the computational model, we showed in [16] that a model of independently operating finite-state machines (FSM) offers a good design tradeoff versus the usage of the microcontrollers alone for free-space motion detection. Furthermore, we showed that the sampling and conditioning of real-time data are best performed by microcontrollers, and that reconfigurable logic is advantageous as compared to very large-scale integration.

The first generation of the embedded system was implemented with an Atmel AVR 90S8515 microcontroller, an Analog Devices ADXL210 2-axes accelerometer [19], and a Xilinx XC4010 low-cost FPGA [16]. The next versions of the embedded system were implemented with larger FPGAs [17], [18]. The second version was implemented with a Xilinx XC4028 FPGA, whereas the last version uses a Xilinx XC2S100 SPARTANII FPGA. The final embedded system has the ability of onboard programming. An external PROM downloads the configuration file to the FPGA on power-up. The cost of the system is less than $70, which is an arbitrary limit that nonetheless precluded certain types of solutions to the motion-detection problem.

The dual-axis accelerometer is positioned on user's hand and responds to hand's accelerations. The microcontroller and the FPGA were placed on a prototype board, and they communicate through an 8-bit data bus and few control bits. All interfaces were implemented at the TTL level (0–5 V). The SPARTANII FPGA operates at the LVTTL level (0–3.3 V); however, its I/O pins are 5-V tolerant. The few pins of the FPGA that are connected to the microcontroller were pulled up in order to operate at the latter's voltage level. The ability to connect with the PC through the RS-232 serial port was employed as a rapid prototyping tool for algorithm evaluation and as a user interface for system training (to adapt to individual needs), but not as a necessary component in field deployment. In the standalone operation of the system, the personal computer (PC) is not connected.

In terms of applications, the original requirement for direct I/O was incorporated into the architecture with I/O lines, which can take any desirable logical values. Through optocouplers, direct interfacing to devices can then be accomplished (e.g., wheelchair control). Moreover, these I/O lines can be easily converted to infrared (IR) ports, so that remote devices may be accessed (e.g., IR-controlled door locks, IR-controlled telephone-device answer/dial, IR-controlled air conditioner). Lastly, the I/O lines can be either decoded, leaving one line per device, or encoded, allowing for larger numbers of devices to be accessed at the expense of more complex per-device decoding of multiple signals.

III. Real-Time Processing of Sensor Data

This section presents in detail 1) the calibration process; 2) the decoding of the X- and Y-axes outputs; 3) the conversion

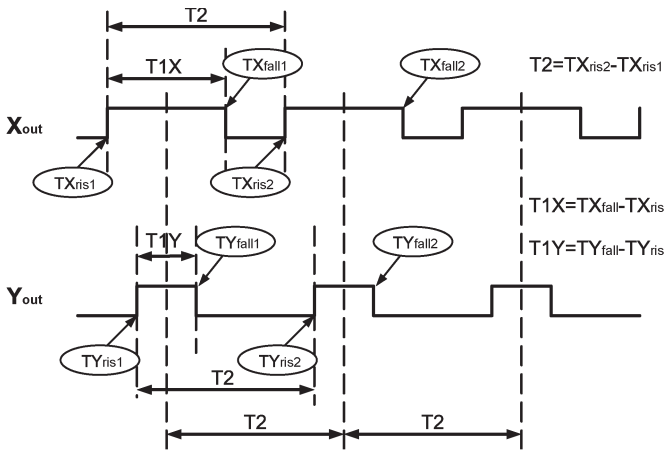

Fig. 2. Decoding technique for the ADXL210 accelerometer.

from duty cycle to acceleration, all performed by the microcontroller; and 4) the transfer of the acceleration data from the microcontroller to the FPGA.

The accelerometers used in this system have dual pulse width modulated (PWM) output [19], with the duty cycle being proportional to the acceleration in each of the two sensitive axes. The X- and Y-axes output signals are connected to the microcontroller's external interrupts INT1 and INT0, respectively. The use of interrupts provides the ability of immediate response, which is superior to the polling method. The PWM signals are measured with the microcontroller's counter. A system calibration process is performed by the microcontroller and sets the level of zero acceleration.

Acceleration in g experienced by the ADXL210 may be calculated by the following formula:

$$a = \frac{\text{DutyCycle} - \text{DutyCycle at } 0 \text{ g}}{\text{DutyCycle change per g}}. \tag{1}$$

By replacing with the nominal value of scale factor [19], the equation takes the form

$$a = \frac{T1/T2 - T1 \text{ at } 0g/T2}{0.04} \tag{2}$$

where $T1$ is the pulsewidth, $T2$ is the period of the PWM output, and $T1$ at $0\ g$ is the pulsewidth at 0 g.

The $T2$ period may drift due to temperature, and 0-g output varies slightly from device to device. The easiest way to calibrate the ADXL210 is by using the earth's gravity as a reference input. A reading of the 0-g point can be determined by orienting the device parallel to the earth's surface. An easy way to calibrate the accelerometer and decode the duty-cycle output is shown in Fig. 2.

Since the duty-cycle modulation uses the same triangle wave reference for the X- and Y-channels, the midpoints of the $T1$ of each period must be coincident. This is illustrated in Fig. 2. A fast PWM-decoding technique that allows the best data-acquisition time can be, therefore, used and meet the real-time decoding requirements. In the following paragraphs, we present in detail the calibration and the decoding processes.

On power up, the FPGA signals the microcontroller to enter calibration mode. The accelerometer X- and Y-axes should be parallel to the horizontal plane, so that both axes experience 0 g. The accelerometer should also be still for a short period of time (less than 100 ms). The operation is performed with the following steps.

1) INT1 is set to be triggered at the rising edge; INT0 is disabled.
2) The counter starts at the rising edge of the X output $(\text{TX}_{\text{ris1}})$.
3) The counter value at the next rising edge of the X output $(\text{TX}_{\text{ris2}})$ is recorded.
4) The counter is cleared.
5) The recorded values of the counter are subtracted in order to produce the $T2$ value.
6) The process is repeated ten times for producing the average value of $T2$.

Then, decoding of the X- and Y-axes outputs is performed (we examine the scenario in which the rising edge of the X channel occurs before the rising edge of the Y channel).

1) INT1 and INT0 are set to be triggered at the rising edge.
2) The counter starts at the rising edge of the X output (TX_{ris}).
3) INT1 is set to be triggered at the falling edge.
4) The counter value at the rising edge of the Y output (TY_{ris}) is recorded.
5) INT0 is set to be triggered at the falling edge.
6) The counter value at the falling edge of the Y output $(\text{TY}_{\text{fall}})$ is recorded.
7) INT0 is set to be triggered at the rising edge.
8) The counter value at the falling edge of the X output $(\text{TX}_{\text{fall}})$ is recorded.
9) INT1 is set to be triggered at the rising edge.
10) The counter is cleared.
11) The recorded values of the counter for each of the two channels are subtracted to produce $T1X$ and $T1Y$ values.
12) The process is repeated for every new sample.

The first ten samples of $T1$ from each channel are used for calibration purposes. These are used to produce the average value of the pulsewidth at 0 g ($T1$ at $0g$) for each axis. Experiments suggested that averaging ten samples during calibration is sufficient to avoid noise effects.

The conversion from duty cycle to acceleration data is performed with (2). A more convenient form of (2) for the implementation into the microcontroller's code is

$$a = (T1 - T1 \text{ at } 0 \text{ g}) \times \frac{25}{T2} \tag{3}$$

with $T1$ being measured for every channel with the microcontroller's counter, $T2$ calculated as the average of ten successive samples, and $T1$ at $0g$ calculated for every channel as the average of ten successive samples. The values $25/T2$, $T1X$ at $0g$, and $T1Y$ at $0g$ are calculated during calibration process. They are stored into internal registers as calibration constants and retained for use in calculating the accelerations after calibration.

Calculation of accelerations is followed by transferring acceleration data to the FPGA. Data transfer is performed synchronously. The information is sent to the FPGA via an 8-bit port. Every X- and Y-acceleration value is interpreted as one 16-bit number. The first 8 bits transmitted to the FPGA represent the LSB part of the acceleration value, and the remaining 8 bits represent the MSB part.

The result of (3) would be a number in the range of ± 10 g for the ADXL210 device. The accelerations are represented as signed fixed point numbers with an integer and a fractional part. In general, it is inconvenient to use this format of numbers in calculations with a microcontroller or an FPGA [20]. In the first implementation of the input subsystem, the microcontroller converted the accelerometer outputs to the fixed-point format described above [16]. In the latest implementation of the system, in order to reduce the design complexity, all primary inputs are level-shifted by a half range to result in unsigned 16-bit numbers and simple unsigned integer arithmetic. Given that the motion-detection model (described in the following section) works equally well with unsigned numbers, there is no compromise on the system accuracy or speed.

To illustrate why such a change, trivial as it may seem, can have an effect on the system design, we present some results from the actual system implementation: Many comparators are needed in order to compare the values of incoming acceleration data with predefined values/ranges of X–Y data. The comparators are implemented with programmable resources of the FPGA called configurable logic blocks (CLBs). For detection of six motions (identical motions in both cases), the initial system required 180 CLBs for the comparators, whereas after the change is number representation only 120 CLBs were needed (out of the 400 CLBs of the XC4010). This reduction illustrates how a low-cost embedded system can save on resources with alternative approaches for some aspect of the design.

IV. Tradeoff Space Evaluation

After sampling the input data, several strategies were considered for their processing, including direct processing of accelerations or using indirect data calculated by integrating accelerations (i.e., velocity and position). After considerable experimentation, we came up to the same conclusion with previous studies that suggest that the latter approaches lead to excessive error [21]. The main reason for this error is that, since integration is performed by multiplying the output of an accelerometer by t (velocity) and t^2 (position), any errors in the output of the accelerometer are also multiplied by these factors. Thus, we abandoned such approaches in favor of processing the original data.

Two generations of the motion-detection model have been developed, which are described below.

A. First Motion-Detection Model

The computational model of the first generation of the system [16] is that of parallel FSMs, each of which consists of states for detection of values/ranges of X–Y data, followed by states to wait for a predefined period of time (including zero time). This way, each motion was represented in terms of thresholds, that need to be exceeded for the state to be active, followed by periods of "not examining the input," which were useful in avoiding local minima (from irregular motion or noise). The model could detect simple motions, i.e., "forward," "back," "right," or "left" hand motions, as well as more complex motions, such as circular ones. Circular motions were represented by more thresholds of accelerations than simple motions and they required more samples in order to be detected. The six motions detected by the first-generation system are shown in Fig. 3. In that system, detection of each motion was independent of the detection of the remaining motions.

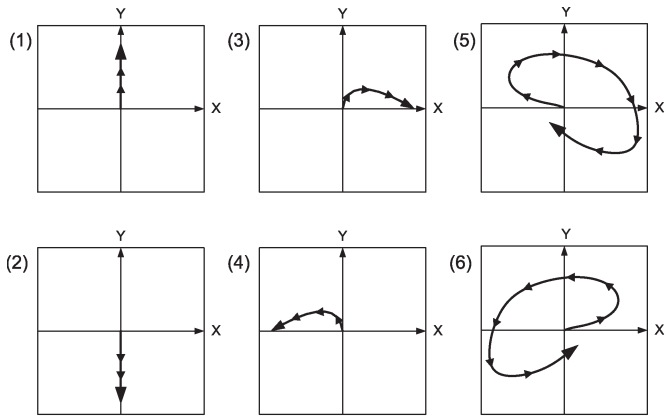

Fig. 3. Initial vocabulary of motions, including four simple (forward, back, left, and right) and two complex (circular) motions. One FSM is implemented per detectable motion. N FSMs recognize N motions.

Even if the first generation of the system could detect fairly complex motions (e.g., circular hand motions), preliminary clinical evaluation showed that these motions were undesirable to the user, regardless of system capability [17]. Circular, being more complicated than simple motions, require more effort by the user in order to be executed. Thus, a simpler vocabulary of motions that can be used in succession can lead into more options for the user, as described below.

B. Second Motion-Detection Model

The second model [17], which we eventually used, is based on the concept that sequences of the four simple motions (forward, back, left, and right) are used to produce a complex vocabulary. Thus, a sequence of two simple motions with n possibilities each produces n^2 distinct motions versus n (more complex) motions of the first model. In both cases, n is the number of FSMs. The hardware cost is O(n) because the same FSMs are reused for each segment of the motion. In the new model, the parallel operation of independent FSMs as well as the modularity of the design is retained from the first model. The FSMs that are kept in the design are only those that led to the detection of the four simple motions. The complex motions are segmented into two simpler motions, and the four FSMs are reused for each segment as is shown in the vocabulary of Fig. 4.

In order to avoid false positives by successions of the same motion (e.g., "forward–forward"), these cases are not

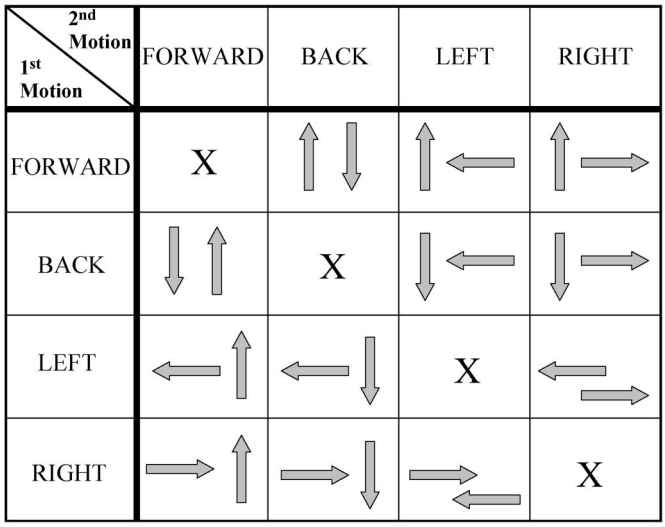

Fig. 4. Second vocabulary of motions, including four simple (forward, back, left, and right)—denoted by "X"—and 12 complex (sequences of simple) motions. N FSMs recognize N^2 motions.

considered as complex motions, leading to $n^2 - n$ well-defined complex motions. The maximum number of complex motions that the system can detect is $4^2 - 4 = 12$. In addition, it recognizes the four simple motions leading to 16 well-defined motions.

C. System Adaptation to Individual Needs

It can be said that the general form of each simple motion is the same, regardless of the specific user. For example, a forward movement consists of a set of accelerations, followed by samples of stable velocity, which are then followed by a set of decelerations (hand starts moving, moves instantaneously without accelerating, and then stops). However, even if this procedure is similar for all users, it is not exactly the same. Clinical tests (of limited scope) have shown that the motions of a kinetically challenged person are not as even as the respective motions of a healthy one. The speed of execution may vary even during the same motion. Thereby, a motion can be too fast in the beginning of the execution and too slow at the end and vice versa. The above observations led to the conclusion that the system must have the capability to adapt to the different executions of the same motion by different users.

The idea of adaptation primarily affects the FSMs that are responsible for motion recognition. FSMs operation is based on the excess of thresholds by acceleration samples collected during motion execution. Thus, adaptation concerns determination of thresholds that must change according to each user. The procedure that has been implemented for threshold calculation is an unsupervised-learning method and is presented in detail in [18].

Briefly, by using a PC as the user interface, we collect data for each simple motion from individuals. These data are the patterns on which the adaptation procedure is based. After calculating the appropriate thresholds for each simple motion, we download them to a nonvolatile memory, i.e., ROM, in order to be available for further use. The microcontroller loads the thresholds from the ROM on power-up and sends them to the FPGA, where they are stored in registers. The FSMs have access to these registers and can read the respective thresholds. This way, they can recognize the motions that are similar to the initial patterns, which the user provided. This procedure leads to a reconfigurable system with capability to adapt to individual needs, but without recompilation of the design.

V. Motion-Detection Models

The flowcharts for the two models are shown in Fig. 5. The first implementation contains six FSMs, whereas the second one contains four FSMs. The difference in the control logic between the two models is represented with bold lines. As shown in the figure, operation flow up to the first dotted line, where the "MotionCounter" value is recorded, is identical for both schemes. In the first model, "MotionCounter" is then checked to determine whether it exceeds the "Recognition value" threshold, whereas in the second model, an intermediate transition state takes place. "MotionCounter" is used to measure the elapsed time after the trigger of an FSM, whereas "Recognition value" is a predefined value that corresponds to the maximum time interval that is allowed for the completion of a motion. Once an FSM detects a motion within that time frame, the value of "MotionCounter" is recorded. A register per FSM is available in order to record the elapsed time that is required for the recognition of the corresponding motion.

The difference between the two models after the update of "MotionCounter" is that in the second one, the completion of an FSM resets all other FSMs. This is necessary due to the partial overlap of different motions which is illustrated in Fig. 6. A portion of acceleration values that represents the "forward" motion is similar to a portion of acceleration values that represents the "back" motion. Thus, detection of "forward" includes partial detection of "back" motion. In the first model, if, e.g., during the execution of "forward" motion, "MotionCounter" has not exceeded "Recognition value," and both FSMs are activated, a small hand deviation during motion detection could lead to completion of "back" motion. The second model avoids this undesirable result with a state that resets all other FSMs.

Another difference is that in the first model, the motion, which was recognized first, terminated the processing. This assumption was based on the fact that there is one FSM per detectable motion. This is not the case in the second model because two different motions that are detected sequentially constitute a complex motion. Thus, the combination of completion of two FSMs leads to the detection of a complex motion. This can, however, cause other problems. For example, the system could potentially detect the complex "forward–back" motion, instead of the simple "forward" motion. A solution to this is to initialize the FSMs every time an FSM sequence is successfully completed.

The next two states are identical for both models. If the "MotionCounter" exceeds "Recognition value," the FSMs are led to an "inactive" state (which is not the initial state), otherwise, the above process is repeated.

When the FSMs enter an inactive state, the system checks whether a complex motion has been detected by comparing

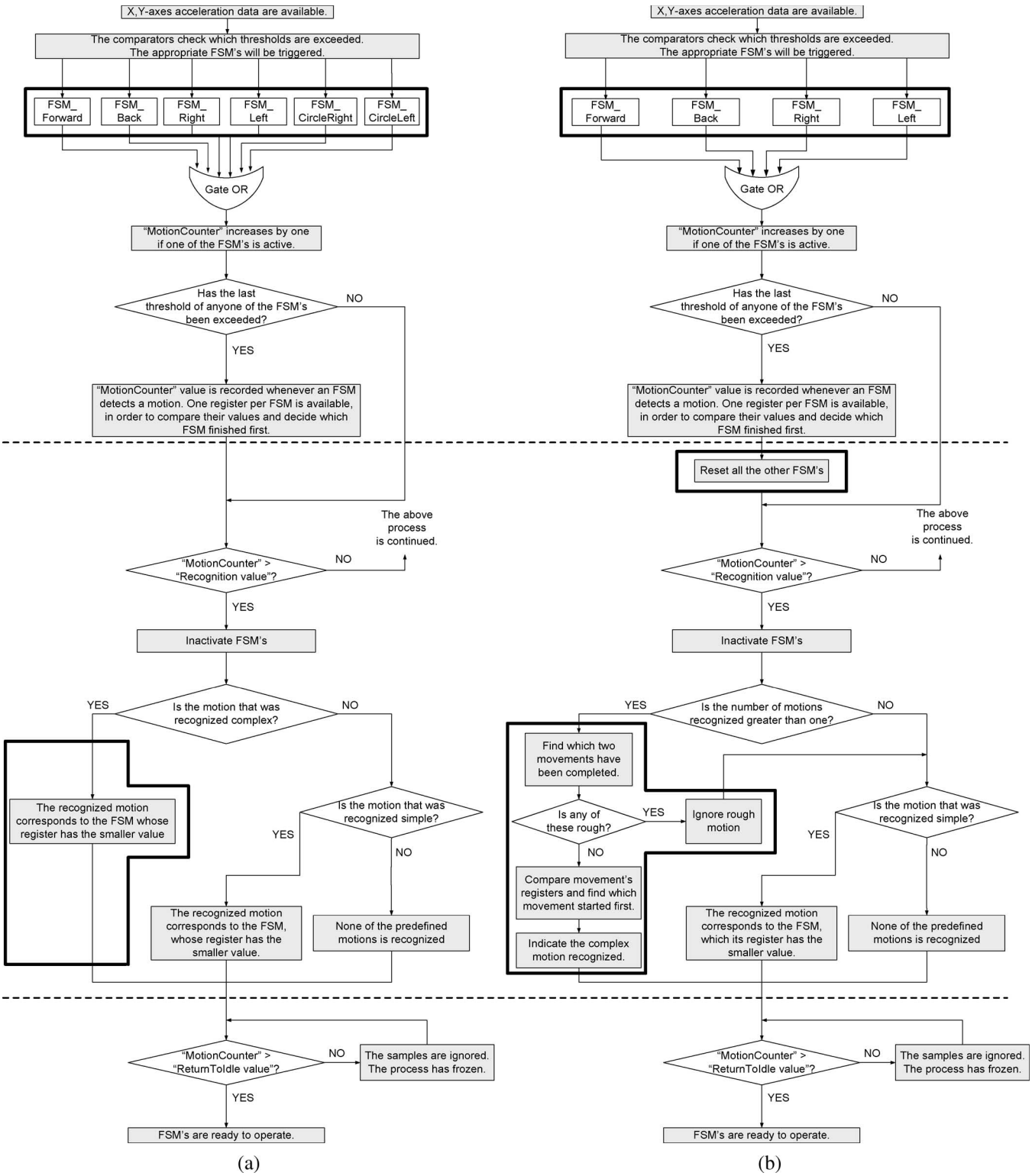

Fig. 5. (a) Initial model of parallel FSMs. (b) Optimized model of parallel FSMs.

the register values. If none of the complex motions has been recognized, the process is identical for both models.

1) The controller checks if the motion is a simple one.
2) If this is true, it checks which of the four registers carries the smallest value, and the algorithm outputs the type of motion, which was recognized in less time.
3) If this is false, the output of the system will not correspond to any of the vocabulary motions, i.e., nonrecognizable motion.

If one of the complex motions has been recognized, the process is different for each model, as described below.

1) The first-generation system [Fig. 6(a)] is as follows.

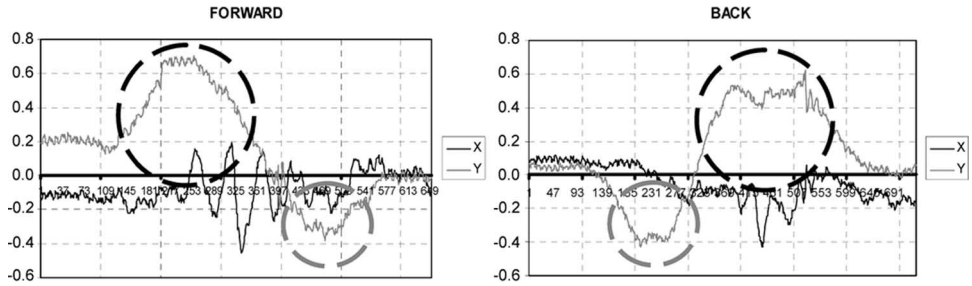

Fig. 6. Partial overlaps between "forward" and "back" motions, indicated by black and gray circles. The vertical axis represents the value of the X and Y acceleration samples, expressed in g, whereas the horizontal axis represents the sample number.

a) The controller checks which of the two registers carries the smallest value, and the algorithm outputs the type of the corresponding motion that was successfully recognized first.

2) The second-generation system [Fig. 6(b)] is as follows.

a) The controller checks which two motions have been detected.
b) The controller checks if some motion was "rough" in order to avoid false positives from tremble. This check is performed in order to avoid local minima (from irregular motions, tremble, etc.), and it is based on the comparison of the value of the corresponding register with a constant that represents the minimum value that is allowed for a motion to be detected (correspond to a time interval of $\sim$220 ms).
c) If it is true (i.e., the motion was "rough"), the motion is ignored, and the next step of the algorithm is to execute the process that is identical for the two models and has been reported above.
d) If it is false (i.e., the motion is valid), the values of registers are compared to find which simple motion was triggered first. The register that carries the smallest value indicates the motion that was successfully completed first.
e) The algorithm outputs the type of motion, which is synthesized by the two simple motions that have been recognized.

The model also allows for priorities to be set as needed for multiple detection (leading to predefined resolution of conflicts). The priority order can change with respect to the user's demands. Furthermore, following successful detection of a motion, the process will freeze with the insertion of a delay. Thus, the user has in his/her service a little time to stabilize his/her hand to a rest state.

VI. Experimental Results and Model Validation

Many experiments have been performed in order to evaluate the behavior and performance of the system. Over 400 experiments were performed with healthy and kinetically challenged subjects. Four healthy subjects tested both generations of the system by performing the respective motions (simple and complex ones). The accelerometer was mounted on the subjects' right hand, and the system's ability to detect the performed motion was assessed. Furthermore, the acceleration traces acquired during these experiments were used to determine average user behavior as well as thresholds and delays for the motion-detection model. The implementation of FSMs in reconfigurable software allowed for substantial experimentation with computational models, thresholds, and delays. All experiments were recorded in a benchmark, which can be reused every time a design change is made in the system. The graphs in Figs. 6 and 7 have been generated from the collected data. The vertical axis represents the acceleration value along the X- and Y-axes, expressed in g, whereas the horizontal axis represents the number of samples. The graphs in Fig. 7 show why the change from the first generation of the system to the second generation was needed. As illustrated, the sequence of acceleration values of the four simple motions is similar to the acceleration values that represent a circular motion (movement no. 5 of the vocabulary in Fig. 3). This shows not only the complexity of the circular motion in terms of user effort, but also the increased complexity in terms of hardware control needed to detect the motion and distinguish it from the simpler ones.

After designing and implementing the model, based on experiments with healthy subjects, we tested it with a kinetically challenged subject with diagnosed paraplegia. One of the significant results of these experiments was that the four simple motions were the preferred motions for this subject. These results led to the second-generation computational model, which uses only the simple motions and their sequential combinations. The results of the experiments were also valuable in more ways. The latest motion-detection model was verified to be independent for each motion. Besides, it worked very well for the motions that the kinetically challenged subject could easily perform.

Table I presents the effectiveness of the two computational models in detecting a motion. These results were determined through experiments with healthy subjects. No motion detection or wrong motion detection, when a hand movement was executed, was considered unsuccessful system performance. We can observe in Table I that both models can recognize simple motions with better than 95% success. The circular motions, being the most complex, are the less identifiable motions. The complex motions of the second model are not as complex as the circular ones and therefore have much higher recognition rates. This material improvement corresponds to better system performance. We should point out that slow motions do not produce any response in this version of the system due to the chosen sensitivity of the accelerometer. A slow motion is perceived by the system as zero-acceleration activity. If more

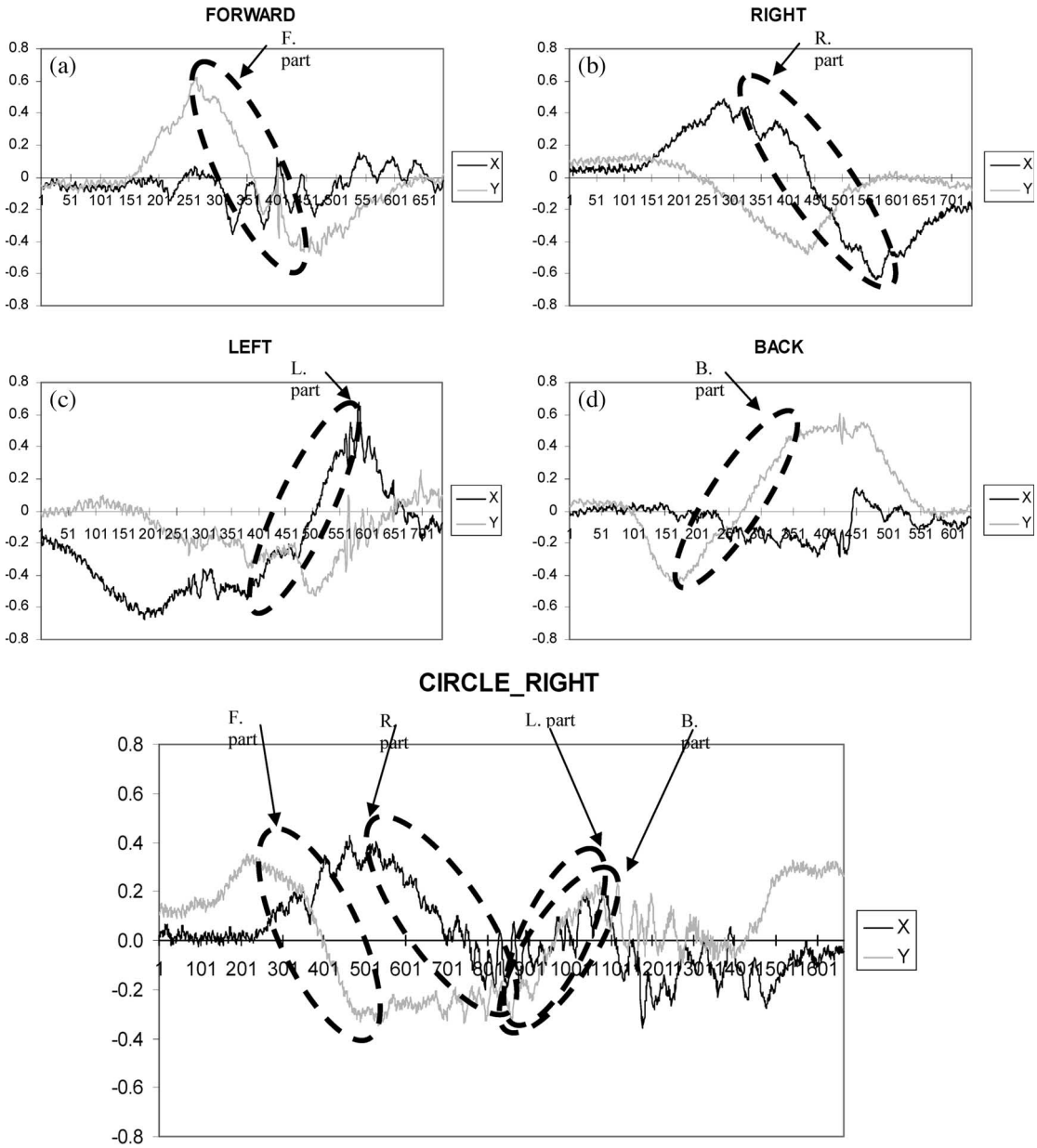

Fig. 7. Sets of acceleration samples that represent simple motions can be synthesized to produce a circular motion. The movement "circle_right" comprises (a) F. part of movement "forward," (b) R. part of movement "right," (c) L. part of movement "left," and (d) B. part of movement "back." The vertical axis represents the value of the X and Y acceleration samples, expressed in g, whereas the horizontal axis represents the sample number.

TABLE I
RECOGNITION EFFECTIVENESS (PERCENTAGE OF CORRECTLY IDENTIFIED MOTIONS)

	1st model	*2nd model*
Simple motions	>95%	>95%
Circular motions	75%	-
Forward + Back/Right/Left	-	95%
Back + Forward	-	95%
Right + Left	-	95%
Left + Right	-	95%
Back + Right/Left	-	80%
Right + Forward/Back	-	80%
Left + Forward/Back	-	80%

TABLE II
COMPARISON BETWEEN THE FEATURES OF THE TWO HARDWARE IMPLEMENTATION IN XC4010 FPGA (FIRST AND SECOND MODELS WITHOUT ADAPTATION)

	1st model	*2nd model*
FSMs implemented	6	4
Motions recognized	6	16
FPGA CLB usage	327/400	375/400
FPGA total usage	81.75%	93.75%
Max Operating Frequency	30 MHz	28 MHz

sensitive accelerometers are used, a system that recognizes slower motions and even minor variations in the accelerations can be implemented. Nevertheless, there are tradeoffs between choosing the smallest detectable acceleration, the highest

TABLE III
COMPARISON BETWEEN THE FEATURES OF THE TWO HARDWARE IMPLEMENTATION IN SPARTANII XC2S100 FPGA (SECOND MODEL WITH AND WITHOUT ADAPTATION)

	1st system	*2nd system*
FSMs implemented	4	4
Motions recognized	16	16
Adaptation to user	NO	YES
FPGA slice usage	500/1200	980/1200
FPGA total usage	41.60%	81.00%

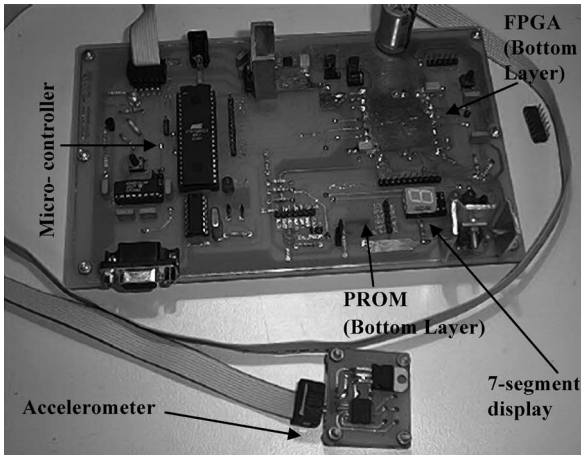

Fig. 8. PCB of the system.

detectable motion frequency, an acceptable noise level, and the sampling rate that need to be considered [19].

Table II compares hardware-system complexity for the two motion-detection models mapped in the first FPGA that was used. The FPGA CLB usage is the number of CLBs that the design occupies when downloaded to the FPGA as a ratio of the total available CLBs to the FPGA. The FPGA total usage gives the same number expressed as a percentage. For both models, the maximum operating frequency is substantially beyond what may be needed in practice. However, the identical FPGA usage and the increased number of detected motions for the second model directly correspond to an overall better system performance for the second model.

Table III compares hardware system complexity for the second model, with and without adaptation of the system to the user, as described in [18]. The device used was a SPARTANII FPGA. The FPGA usage is much bigger when the adaptation process is integrated into the design.

VII. PRESENT STATUS AND FUTURE WORK

In conclusion, we have presented a two-dimensional (2-D) motion-detection model, which can be readily implemented in reconfigurable hardware for a low-cost solution. The improved detection model decreases processing complexity. Therefore, the system has diminished demands in reconfigurable resources, which contributes to the low cost of the whole design. Furthermore, a simple vocabulary of motions can be used to form more complex motions. The system can be trained to individual users' needs, and the vocabulary of the motions can be easily extended.

A photograph of the printed-circuit board (PCB) of the system is shown in Fig. 8. The small board with the accelerometer is mounted on the user's hand. However, if a person does not prefer to use his/her hand, the accelerometer can be placed on another limb of the body. This selection highly depends on the accuracy with which the person controls the corresponding limb to execute the desirable motion. The accelerometer is calibrated when it is powered on. This way, the system can be adjusted to different orientations and to different accelerometers. Moreover, the system can operate in different temperature conditions.

Clinical trials have to be expanded, and as this technology matures and does prove to be valuable, we are considering commercial development of this system. Certain aspects of this system have been filed for a U.S. patent.

ACKNOWLEDGMENT

The authors would like to thank Dr. T. Kean for the contributions and the valuable advice throughout this project. They would also like to thank the Xilinx Corporation, for the generous donations to their educational and research activities, and Analog Devices for the electronics parts. Last, they would also like to thank Mr. Koutsorinakis for the willingness to test their system in various stages during its development.

REFERENCES

[1] T. Kuhlen and C. Dohle, "Virtual reality for physically disabled people," *Comput. Biol. Med.*, vol. 25, no. 2, pp. 205–211, 1995.
[2] L. Bretzner, I. Laptev, T. Lindeberg, S. Lenman, and Y. Sundblad, "A prototype system for computer vision based human computer interaction," Royal Inst. Technol. (KTH), Stockholm, Sweden, Tech. Rep. ISRN KTHNA/P-01/09-SE, Apr. 2001.
[3] V. Kumar, T. Rahman, and V. Krovi, "Assistive devices for people with motor disabilities," in *Wiley Encyclopedia of Electrical and Electronics Engineering*, J. G. Webster, Ed. New York: Wiley, 1999.
[4] C. Lau and S. O'Leary, "Comparison of computer input devices for persons with severe physical disabilities," *Amer. J. Occup. Ther.*, vol. 47, no. 11, pp. 1022–1030, 1993.
[5] J. D. Bronzino, "Rehabilitation engineering technologies: Principles of application," in *The Biomedical Engineering Handbook*, 2nd ed, vol. 2, 2000, pp. 146-1–146-9.
[6] V. Stanford, "Biosignals offer potential for direct interfaces and health monitoring," *Pervasive Comput.*, vol. 3, no. 1, pp. 99–103, Jan.–Mar. 2004.
[7] L. M. Bergasa, M. Mazo, A. Gardel, J. C. García, A. Ortuno, and A. E. Mendez, "Guidance of a wheelchair for handicapped people by head movements," in *Proc. 7th Int. Conf. ETFA*, Barcelona, Spain, 1999, pp. 105–111.
[8] "Motion Analysis System," US6301964 Patent, Oct. 16, 2001. DYHASTREAM INNOVATIONS INC CA.
[9] "Monitoring System for Elderly Person Includes Activity Sensor and Physiological Sensor Combined to Initiate Warning of Fall," FR2808609 Patent, Nov. 9, 2001. UNIV RENNES (FR).
[10] K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," *ACM Comput. Surv.*, vol. 34, no. 2, pp. 171–210, 2002.
[11] S. Hauck, "The roles of FPGAs in reprogrammable systems," *Proc. IEEE*, vol. 86, no. 4, pp. 615–639, Apr. 1998.
[12] K. Nakahara, E. E. Sabelman, and D. L. Jaffe, "Development of a second generation wearable accelerometric motion analysis system," in *Proc. Annu. Fall Meeting BMES*, Atlanta, GA, 1999, p. 630.
[13] T. Ishimatsu, N. Irie, and O. Takami, "Computer interface device for handicapped people using head movement," in *Proc. IEEE Pacific Rim Conf. Commun., Comput. and Signal Process.*, Victoria, BC, Canada, Aug. 1997, vol. 1, pp. 273–276.
[14] M. Mazo *et al.*, "An integral system for assisted mobility," *IEEE Robot. Autom. Mag.*, vol. 8, no. 1, pp. 46–56, Mar. 2001. (SIAMO Project Group).
[15] Kayser-Threde GMBH, *3D Eye Tracking Device*, Nov. 2000, Munchen, Germany. Technical Description.

[16] A. Dollas, K. Papademetriou, N. Aslanides, and T. Kean, "A reconfigurable embedded input device for kinetically challenged persons," in *Proc. Eur. Conf. FPL*. Belfast, Northern Ireland, Aug. 2001, p. 319.
[17] K. Papademetriou, A. Dollas, and S. Sotiropoulos, "A second generation embedded reconfigurable input device for kinetically challenged persons," in *Proc. 11th IEEE Symp. FPGAs for Custom Computing Machines (FCCM)*, Napa, CA, Apr. 2003, pp. 294–295.
[18] A. Dollas, S. Sotiropoulos, and K. Papademetriou, "A 2D motion detection model for low cost embedded reconfigurable IO devices," *IEEE Trans. Biomed. Eng.*, vol. 52, no. 8, pp. 1443–1449, Aug. 2005.
[19] *Low Cost ±2 g/±10 g Dual Axis iMEMS Accelerometers With Digital Output*, ADXL202/210 Device Datasheet, Analog Devices, Norwood, MA.
[20] H. Weinberg, *Using the ADXL202 Duty Cycle Output*, Application note, rev A., Analog Devices, Norwood, MA.
[21] C. Verplaetse, "Inertial proprioceptive devices: Self-motion-sensing toys and tools," *IBM Syst. J.*, vol. 35, no. 3/4, pp. 639–650, 1996.

Kyprianos Papadimitriou (S'06) received the Diploma and M.Sc. degrees in electronic and computer engineering from the Technical University of Crete, Chania, Greece, in 1998 and 2003, respectively, where he is currently working toward the Ph.D. degree in the Department of Electronic and Computer Engineering.

During 1998–1999, he was with the R&D Department of Atmel Corporation, where he worked on hardware implementation of wireless protocols. His research interests are in reconfigurable computing, with emphasis on techniques for scheduling and allocation in partially reconfigurable systems. He is currently with the Microprocessor and Hardware Laboratory, Electronic and Computer Engineering Department, Technical University of Crete.

Apostolos Dollas (S'85–M'87–SM'93) received the B.S., M.S., and Ph.D. degrees in computer science from the University of Illinois at Urbana—Champaign, in 1982, 1984, and 1987, respectively.

He has been a Professor with the Electronic and Computer Engineering Department at the Technical University of Crete, Chania, Greece, where he has also served one term as Department Chairman. He has also been on the faculty of the Electrical Engineering and of the Computer Science at Duke University. He is currently the Director with the Microprocessor and Hardware Laboratory, Technical University of Crete. He conducts research and teaches in the areas of reconfigurable computing, rapid system prototyping, embedded systems, and application-specific high-performance digital systems. In all of these areas, he places emphasis on the development of fully functional prototypes.

Prof. Dollas is a member of the IEEE Computer Society. He belongs to the Eta Kappa Nu and Tau Beta Pi. He is a recipient of the IEEE Computer Society Golden Core Member Award, the IEEE Computer Society Meritorious Service Award, and twice the Department of Computer Science Award for Outstanding Teaching at the University of Illinois at Urbana—Champaign. He has served on the program committees of several IEEE conferences and workshops, including Field-Programmable Custom Computing Machines (since 1993), Rapid System Prototyping (since 1990, Program Cochairman in 2001), and Tools for Artificial Intelligence (1989–1991).

Stamatios N. Sotiropoulos received the Diploma in electronic and computer engineering from the Technical University of Crete, Chania, Greece, in 2003, and the M.Sc. degree in biomedical engineering from the University of Minnesota, Twin Cities, in 2005. He is currently working toward the Ph.D. degree as a Marie Curie Research Fellow at the University of Nottingham, Nottingham, U.K.

His research interests are in central nervous system electrophysiology, brain electrical stimulation, and brain MR imaging.

Mr. Sotiropoulos has been a Student Member of the Society for Neuroscience since 2003.