Journal of Computers Vol. 31 No. 3, 2020, pp. 266-274
doi:10.3966/199115992020063103021

Design of Embedded Sensor System with Parallel Reconfigurable Computing Platform

Chi-Chou Kao*

Department of Computer Science and Information Engineering, National University of Tainan, Tainan, Taiwan
cckao@mail.nutn.edu.tw

Received 26 October 2018; Revised 02 March 2019; Accepted 09 April 2019

Abstract. Embedded devices and sensor networks are becoming popular and cheap, enabling the design of an alternative pathology detection system to monitor structures based on these technologies. On the other hand, configurable computing machines are evolving rapidly. The parallel reconfigurable computing can connect to high-speed microcontroller and peripheral equipment of the embedded system. Based on those reasons, we use parallel reconfigurable computing as the signal processing core to integrate the different transmission interface and generate an embedded sensor system. Compared with other work, the embedded system design satisfies the ease of use, fault tolerance, scalability, low consumption, and flexibility requirements. The experimental results demonstrate the proposed system has great performance and practicability.

Keywords: configurable computing, embedded system, sensor, parallel, PRC

1 Introduction

If a device needs to communicate network and intelligent operations, the device must sense the environment. Thus, the device needs effective sensing element to sense physical environmental changes and convert into electrical signals such as changes in light, temperature, humidity, and gas into resistance, current, voltage, capacitors and other electronic signals. By sensing element signal, the operating system can understand the situation and maintain the best working condition. As a technology, the sensing system has been developed to be the embedded systems such that system complexity can be simplified. However, it needs a high performance platform to distinguish meaningful message or meaningless noise, and then is able to perceive changes in the environment to adjust the operation of the controlled device. This computing platform needs to have a complete computer configuration, including the processor, memory, input and output interfaces, peripheral devices, the necessary operating system, drivers and applications. In the paper, we will design a computing platform including hardware and software as an embedded sensor system.

Reconfigurable architectures, which use a limited number of logic units to set up coarse-grain parallelism, are powerful and flexible structures for multimedia applications and high-performance computing [1-3]. Reconfigurable systems are built around programmable hardware, which consists of programmable devices, such as field programmable gate arrays (FPGAs), with programmable interconnects [4-11]. The main difference compared to conventional microprocessors is the ability to change the hardware during run-time by loading a new circuit. Generally, a program can be accelerated via dynamic reconfiguration or run-time reconfiguration (RTR) hardware. RTR is similar to the concept of virtual memory as it allows additional customization during program execution [12-13]. The physical hardware is much smaller than the sum of the resources required by each of the configurations. Therefore, instead of reducing the number of configurations that are mapped, the configurations are swapped in and out of the actual hardware as they are needed. Because the reconfigurable hardware can be changed

* Corresponding Author

during run-time, more areas of an application can be mapped to the hardware. This increased usage of hardware improves performance.

Several memory configuration schemes have been proposed for dynamic reconfigurable systems. A single-context device, such as Dharma [14] or time-multiplexed FPGA (TM-FPGA) [15], is programmed using a serial stream of configuration information and requires a complete reconfiguration in order to change any of the programming bits. Dynamically programmable gate arrays (DPGAs) [16] are a multi-context device that has multiple layers of programming bits, each of which can be active at a different point in time. Multi-context devices can be considered as collections of single-context devices, each of which can be changed at any time. Extremely fast context switching is thus possible.

In typical multi-context FPGAs, contexts are executed sequentially so that the same resources are reused in different time slots. However, smaller contexts increase the logic and interconnect area and redundant contexts increase configuration memory [17]. Partial dynamic reconfiguration (often referred to as partial RTR) provides the framework to compensate for large reconfiguration times and enables customization of the limited logic resources to better satisfy application performance/power requirements [18]. Many computation intensive tasks, such as the discrete cosine transform (DCT), are completely data-parallel; i.e., the results of task execution on a block of data are independent of those from other blocks. With a partial RTR architecture, it is possible to improve the application execution time by dynamically adjusting the parallelism granularity of data-parallel tasks. The architecture can be reconfigured to instantiate multiple copies of the tasks during application execution; each instance uses an identical amount of hardware logic resources, but processes only part of the data. Due to complete pipelining, the execution time is directly proportional to the volume of data processed, and thus, reducing the data volume improves the application execution time. To benefit from on-demand computing capability, we need to develop methods for scheduling and allocating large circuit designs and applications under physical and architectural constraints for the partial RTR architecture.

To further optimize system performance, the parallel processing technique usually utilizes multiple instructions, multiple data (MIMD) architecture. The MIMD organization with multiple processors and I/O processors can access one or more memory module via bus. However, this kind of organization has a problem. Since all memory references pass through a common bus, the speed of the system is reduced by the bus cycle time. One way to improve performance would be to equip each processor that has memory with several contexts as a dynamically reconfigurable computing machine. That would reduce the number of bus accesses dramatically. A reconfigurable computing machine combined with the parallel processing technique is called a parallel reconfigurable computing (PRC) machine. An architecture consisting of reconfigurable processing units (RPUs) can further optimize system performance [19-21]. In this architecture, the RPU is a dynamically reconfigurable computing core. Figure 1 is a diagram depicting parallel reconfigurable computing architecture. The PRC architecture is capable of handling high computation and data intensive applications such as MPEG-4 or H.264 video encoder [20].

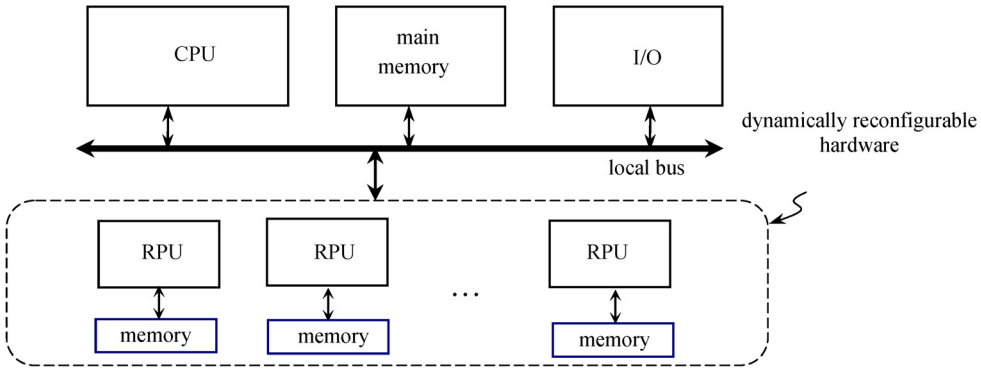


Fig. 1. Parallel reconfigurable computing architecture

The PRC can connect to high speed microcontroller and peripheral equipment of the embedded system. Based on the reasons, we use parallel reconfigurable computing as the signal processing core to integrate the different transmission interface and generate an embedded sensor system. Compared with other work, the embedded system design satisfies the following requirements.

- Ease of Use: The system can configure itself, without human intervention, reducing maintenance costs.

- Fault tolerance: The system can be designed so as not to leave any node isolated.
- Scalability: The system can be as extensible as possible, to place in areas potentially far away from the building.
- Low consumption: The system can be equipped with a battery so the consumption is very low.
- Flexibility: The system is an independently configurable parameter in every single node.

The remainder of this paper is organized as follows. The proposed embedded sensor system is described in Section 2. Section 3 presents the solutions of some important issues for the embedded sensor system. Experimental results are provided in Section 4. Conclusions can be found in Section 5.

2 Proposed Embedded Sensor System

The proposed embedded sensor system is divided into four main blocks including: (1) sensing signal acquisition, (2) signal processing, (3) transmission interface, and (4) system application platform as shown in Fig. 2. In the system, different users first capture the physical demand signals from the outside and convert into electrical signals through the sensor and converter in the sensing signal acquisition block. Next, the signal processing block uses the parallel reconfigurable computing (PRC) machine to do the filter circuit and analog/digital conversion to transmission interface. Finally, the transmission interface block transmits through different interfaces to system application platform. We will describe the four blocks in detail as follows.

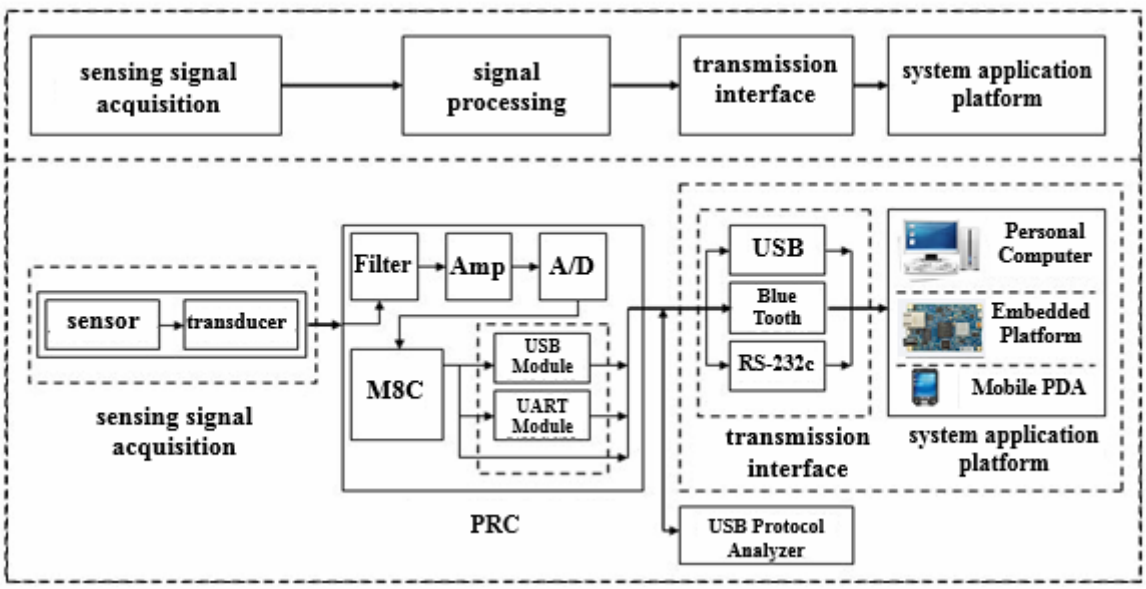


Fig. 2. System architecture block diagram

2.1 Sensing Signal Acquisition Block

The block includes the sensor and transducer. The sensor is to convert the physical external signal to the measuring signal and the transducer is to convert physical quantities into electrical signals. The transducer must meet the difference between drive and measurement circuit as shown in Fig. 3. If different signal patterns are sensed by the system at the same time, the design approach is difficult. The reason is because system designs require a complex circuit to measure the signal. It will generate high cost and the noise in the hardware is more complex. Hence, we will measure the physical quantity sensing signal collection circuit and signal processing to do integration and modular. Different functions and applications of the sensor module will be separated from the system circuit. The design will make the embedded system more flexible because sensing circuit applications can be removable. In addition, because the measurement signal is being digitized in the module so the system is more anti-noise capability.

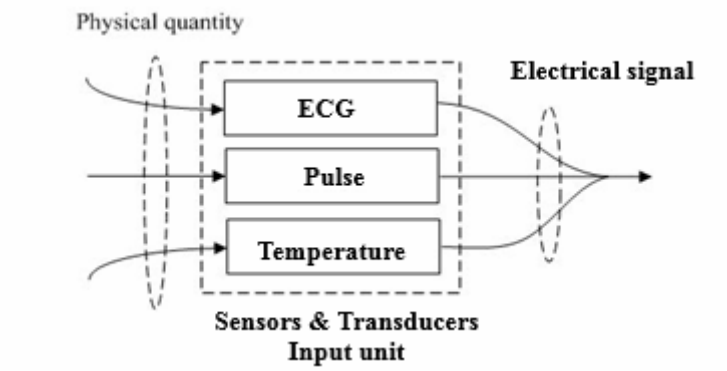


Fig. 3. The sensing signal transducer schematic

2.2 Signal Processing Block

In the block, the various external physical digital will be converted to signal processor. In general, the signals captured by sensors require specific signal processing technique. However, many measuring instruments are not available for all kinds of signal processing functions. For example, physiological signal need to produce a weak voltage signal so an amplifier filter, analog-to-digital converter and other processing operations are necessary. The temperature control sensor requires impedance conversion, filter, amplifier, comparator circuit, and signal conversion. However, no instruments can satisfy all measurement system requirements. Therefore, in the signal processing block, we combine sensors with parallel reconfigurable computing (PRC) machine to generate signal processing technology such that various signal types can be transfer to the system as shown in Fig. 4. The maximum differences between PRC and traditional system are in the development environment. The PRC can support the software and hardware debugging functions, such as user module selection view, interconnect view, and debugger. In addition, users can configure hardware circuit to target design. It makes users more &rapid development in the design process for hardware circuit and signal processing.

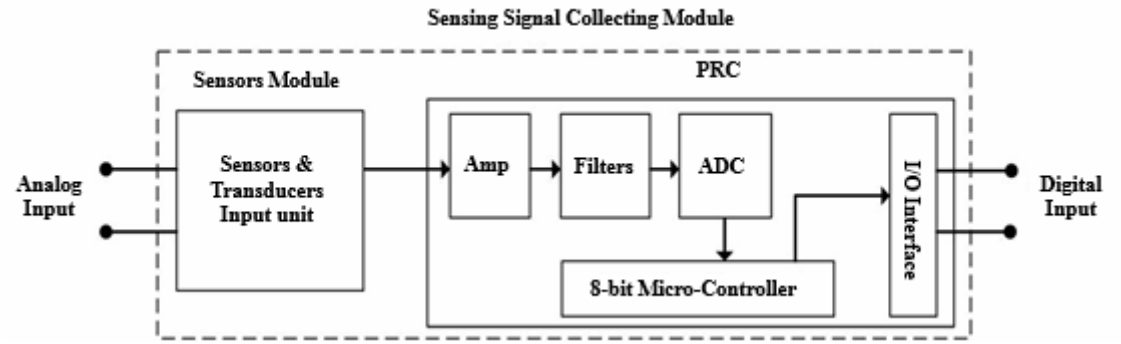


Fig. 4. Sensing signal block

2.3 Transmission Interface

General measurement system can only rely on a single channel to transmit signals and most of the transmission interfaces are RS-232C interface. However, using a single interface to transfer data is insufficient because each transmission interface has limitations of transmission. In the proposed system, we adds Bluetooth wireless transmission and Universal Serial Bus (USB) interfaces to allow designers to decide what transmission interface to connect to the system application platform according data size and transmission speed. It enhances the usefulness of the overall system architecture.

2.4 System Application Platform Block

There are two different system application platforms in the proposed system. One is Windows CE operating system embedded single board computer and the other is the Windows XP operating system for

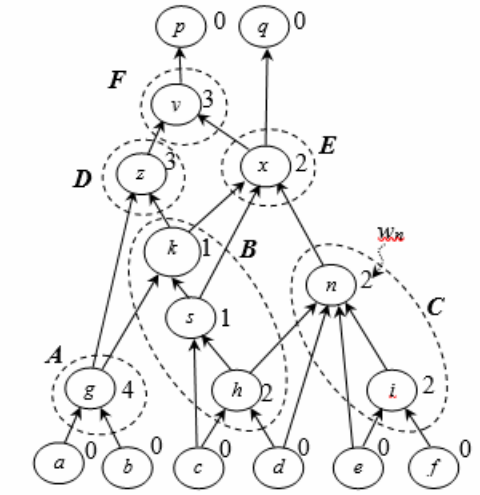
personal computers. We use Embedded Visual C ++ in Windows CE operating system and Visual Basic in Windows XP operating system to develop applications. We develop program in the Windows CE operating system. The program is mainly the C or C ++ and can be applied mobile development tools. The advantages of the program user interface are small, fast, and high portability. For application design, we use Visual Basic language to develop.

3 The Scheduling, Block Allocation and Consistency Methods for the Embedded Sensor System

In the above system, hardware restructuring for increasing power energy, memory space and time consumption costs cannot be ignored. Therefore, we need some methodology to obtain an optimal approximate solution in the proposed system.

3.1 Scheduling Method

The scheduling method is to determine the execution time and sequence of each task in the PRC. In the proposed system, we set a data acyclic graph (DAG), $G = (V, E, W)$, represents can be cut into n work model of parallel reconfigurable computing units calculations, where V is the set of nodes, $|V| = n$, each node $v_i \in V$ represents a reconfigurable computing unit, there exists a weight $w_i \in W$ that represents the size of v_i, $1 \leq i \leq n$, E is set of edges, if there is a directed edge $e_{ij} = \langle v_i, v_j \rangle \in E$, represents has reconfigurable computing units of output function v_i is other reconfigurable computing units of input function v_j. An algorithm [22] that divides the given circuit into subsystems such that each subsystem cannot exceed its reconfigurable capacity. The union of the solutions of these subsystems is the solution of the whole system. Then, a greedy approach is proposed to find the solution for each subsystem. The time complexity of this algorithm is proven to be bounded by $O(n^3)$, where n is the number of the functional operations for the given task. A greedy algorithm is used that maximizes data parallelism for task scheduling of parallel reconfigurable architectures [22]. For example, the DAG in Fig. 5.a is partitioned into six blocks, namely A, B, C, D, E, and F to generate a feasible partitioning solution. Fig. 5(b) shows the execution sequence. In this example, the area of the reconfigure unit is 4. It can be seen that blocks A, B, and C are implemented simultaneously in the first cycle and that blocks D and E are implemented simultaneously in the second cycle. Finally, block F is implemented in the third cycle. The depth of the feasible partitioning solution is 3. Using the algorithm, if the subgraphs have inherent parallelism, we can organize the properly partitioned subgraphs so that they can be executed during the same cycle to improve performance.



(a)

execution sequence	
1	*A*, *B*, *C*
2	*D*, *E*
3	*F*

(b)

Fig. 5. A corresponding DAG of a task

3.2 Block Allocation Method

An application has a lot of work. Each task has its own operation types. However, the operation types can be the same patterns. It is to say that the task can do similar operation but part the input is not the same. Therefore, we can implement of the same the same operations on a block and then reuse the block. Moreover, if a task is executed by a block, and then the task is performed again in a short time, we should do not to re-configure the block. When a task cannot find a suitable block to use, we will choose the least used blocks to execute the task. These methods can obviously reduce the number of reconfigurations and will therefore reduce time and energy. In our design, we use the pre-fetch technique to reduce time of restructuring block. If we have known a task must be performed at which block and the block is not be used now, we will construct in advance the block such that the task can immediately is executed on the block to reducing the number of reconfigurations.

3.3 Block Consistency Method

Our pre-fetch technique can increase the complexity of the task configuration. To reduce complexity, we select the largest block to accommodate the largest of task. Moreover, when we select which block to perform a task, we begin from the smallest blocks and try to use the most suitable for the size of the space such that all reconfigurable logic blocks are used fully.

4 Experimental Results

This section describes the experiments carried out to verify the proposed system. We first describe the experimental environment. Next, several tests dealing with coverage and power consumption where done in order to evaluate the system performance.

4.1 Experimental Environment

In order to verify the effectiveness of the proposed system, we must establish a model of parallel reconfigurable computing system. The model will be established in a hardware/software co-execution environment. The Xilinx 4000 series field programmable Array (FPGAs) is used as the main unit. Each field programmable array allocated to local memory. The transmission control throughout the system hardware and software interface controllers to perform. It links coprocessors platform and SUN workstations. The SUN workstation is used to perform the algorithms and some of the pre-processing (such as Boolean algebraic simplification) algorithms. Using the model, we can test many real-life applications to understand the actual efficiency of the proposed system.

4.2 Comparison Results

We measured the energy and time spent probing the medium, starting a transmission, sending one packet, stopping a transmission after a successful or failed transmission, and receiving a packet. Every scenario was reconstructed with Matlab by adding the energy expended during each performance. Measurements indicate that it takes a little under 32 μs to send one byte, a source node sends 40Bs (bytes) packets at a rate of 1/2 ptk.s^{-1}, and the time separating two frames in a stream is 1.351 *m*s. The power model presented in [23] is assumed. The amount of power consumed over a distance is given by:

$$P_{Tx}(l,d) = lP_{elec}^{Tx} + l\varepsilon_{amp}d^n$$
$$P_{Rx}(l,d) = lP_{elec}^{Rx}$$

where P_{elec}^{Tx} and P_{elec}^{Rx} are the amounts of power being dissipated to run the transmitter's or and receiver's circuitry, respectively, ε_{amp} is the power consumed by the transmission amplifier for a receiver with a distance *d*. *L* is the length of the transmitted/received message in bits, and *n* is the path loss exponent which is limited by the radio propagation model.

Based on the above explanations, Table 1 gives the average power consumption and packet transmission delay (the delay) for the proposed system, SoC [29], and WiseMAC [24]. Compared to the existing

system, the proposed system consumes 12.5~62.5% less power. The delivery delay of regular packets for the proposed system is also less than the other systems.

Table 1. Power consumption and packet delay

proposed		SoC		WiseMAC	
Power (J)	Delay (s)	Power (J)	Delay (s)	Power (J)	Delay (s)
1.6	1.36	1.8	1.74	2.6	1.52
1	1	+12.5	+27.9%	+62.5%	+11.8%

We compared the power consumption of the proposed system with that of a previously published scheme [26], which uses Bluetooth™ technology and point-to-point architecture to connect a personal digital assistant (PDA) and general wireless sensor nodes, for wireless ECG sensors. The requirements of ECGs are set as follows:

- sampling rate: 400 samples/sec
- compression rate: 1.92
- sensor data bits per sample: 12-bits
- average transfer time per sample: 6.25 μs
- setup and control overhead time per sample: 12.5 μs (the setup and control overhead time is the time taken to initialize the RF transceiver, modulator, and demodulator, and executing the control signals.)
- switch overhead time per sample: 1.56 μs (the switch overhead time is the time required to change the transmitter or receiver in a transceiver.)
- total transfer time: 1.525 ms
- sleep time: 0.995 s
- duty cycle: 0.488% (duty cycle = $\frac{T_A}{T_S}(1+N_R)$, where the T_A is active time, T_S is allocating sleep time, and N_R is the number of retransmissions [27-28].)

The power consumption of the ECG sensor system [29] is 765.6 μW and that of the proposed system is 658.7 μW for 400 ECG values per second. It shows that the power consumption of the proposed work is 16% lower.

This coverage test has consisted in detecting the mean RSSI (Received Signal Strength Indication) value, obtained from the reception of 100 messages. Each measure is averaged from 5 repetitions of the same experiment, to counteract the signal fluctuations caused by indoor fading [30]. Both modules have transmitted with a power of 3 dB. The obtained results are shown in Table 2 and Table 3.

Table 2. Signal loss in free space

Free Space	Mean Attenuation (dB)
50 cm	0.00
1 m	7.12
2 m	10.25
4 m	16.65
8 m	20.86
12 m	27.45

Table 3. Signal loss with obstacles

Obstacle	Mean Attenuation (dB)
Window (Open Metallic Blinds)	0.95
Window (Closed Metallic Blinds)	3.62
Wall with Open Door	0.25
Wall with Closed Door	1.07
Brick Wall	1.39
Between Floors	11.65

5 Conclusions

In the paper, we propose an embedded sensor system with parallel reconfigurable computing platform. The system can improve significantly performance and reduce power. Moreover, we also propose some scheduling and allocating methods under physical and architectural constraints for the parallel reconfigurable computing architecture. We have simulated for some experiments. The measured results have validated the effective operation of the design.

Acknowledgements

This work was supported in part by Ministry of Science and Technology, Taiwan, under Grant MOST 103-2622-E-024-006-CC3

References

[1] I. Skliarova, A multimedia tool for teaching reconfigurable computing, in: Proc. International Conference on Computer and Electrical Engineering, 2009.

[2] R. Sangireddy, H. Kim, A.K. Somani, Low-power high performance reconfigurable computing cache architectures, IEEE Transactions on Computers 53(10)(2004) 1274-1290.

[3] P. Dai, X.A. Wang, X. Zhang, A novel reconfigurable operator based IC design methodology for multimedia processing, in: Proc. IEEE TENCON, 2009.

[4] Z. Ruan, Y.Z. Han, H.B. Cai, S.Z. Jin, J. Han, A dynamically partial-reconfigurable FPGA-based architecture for data processing on space solar telescope, in: Proc. International Symposium on Industrial Embedded Systems, 2007.

[5] M. Awad, FPGA supercomputing platforms: a survey, in: Proc. International Conference on Field Programmable Logic and Applications, 2009.

[6] J.M. Emmert, C.E. Stroud, M. Abramovici, Online fault tolerance for FPGA logic blocks, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 1(2)(2007) 216-226.

[7] I. Lebedev, S. Cheng, A. Doupnik, J. Martin, C. Fletcher, D. Burke, Mingjie Lin, J. Wawrzynek, MARC, A many-core approach to reconfigurable computing, in: Proc. International Conference on Reconfigurable Computing and FPGAs (ReConFig), 2010.

[8] T.J. Todman, G.A. Constantinides, S.J.E. Wilton, O. Mencer, W. Luk, P.Y.K. Cheung, Reconfigurable computing: architectures and design methods, IEE Proceedings Computers and Digital Techniques 152(2)(2005) 193-207.

[9] S.C. Goldstein, H. Schmit, M. Budiu, S. Cadambi, M. Moe, R.R. Taylor, PipeRench: a reconfigurable architecture and compiler, Computer 33(4)(2000) 70-77.

[10] M.A. Gora, A. Maiti, P. Schaumont, A flexible design flow for software IP binding in FPGA, IEEE Transactions on Industrial Informatics 6(4)(2010) 719-728.

[11] X. Shao, D. Sun, Development of a new robot controller architecture with FPGA-based IC design for improved high-speed performance, IEEE Transactions on Industrial Informatics 3(4)(2007) 312-321.

[12] T. Heng, R.F. DeMara, A multilayer framework supporting autonomous run-time partial reconfiguration, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 16(5)(2008) 504-516.

[13] S. Wray, W. Luk, P. Pietzuch, Run-time reconfiguration for a reconfigurable algorithmic trading engine, in: Proc. International Conference on Field Programmable Logic and Applications, 2010.

[14] C.H. Lin, C.Y. Chen, A.Y. Wu, Area-efficient scalable MAP processor design for high-throughput multistandard convolutional turbo decoding, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 19(2)(2011) 305-318.

[15] G.M. Wu, J.M. Lin, Y.W. Chang, Generic ILP-based approaches for time-multiplexed FPGA partitioning, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 20(10)(2001) 1266-1274.

[16] S.J. Stone, R. Porter, Y.C. Kim, J.V. Paul, A dynamically reconfigurable field programmable gate array hardware foundation for security applications, in: Proc. International Conference on ICECE Technology, 2008.

[17] H.M. Waidyasooriya, M. Hariyama, M. Kameyama, Implementation of a partially reconfigurable multi-context FPGA based on asynchronous architecture, IEICE Transactions on Electronics 92-c(4)(2009) 539-549.

[18] S. Banerjee, E. Bozorgzadeh, J. Noguera, N. Dutt, Selective band width and resource management in scheduling for dynamically reconfigurable architectures, in: Proc. the 44th Design Automation Conference, DAC 2007, 2007.

[19] M.A. Bayoumi, Parallel Algorithms and Architectures for DSP Applications, Kluwer Academic, Norwell, MA, 1991.

[20] L.-F. Chen, Y.-K. Lai, VLSI architecture of the reconfigurable computing engine for digital signal processing applications, in: Proc. IEEE Circuits and Systems Conference, ISCAS ’04, 2004.

[21] K.A. Vissers, Parallel processing architectures for reconfigurable systems, in: Proc. Design, Automation and Test in Europe Conference and Exhibition, 2003.

[22] C.-C. Kao, Performance-oriented partitioning for task scheduling of parallel reconfigurable architectures, IEEE Transactions on Parallel and Distributed Systems 26(3)(2015) 858-867.

[23] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, An application specific protocol architecture for wireless microsensor networks, IEEE Transactions on Wireless Communalizations l(4)(2002) 660-670.

[24] A. El-Hoiydi, J. Decotignie, WiseMAC: an ultra low power MAC protocol for multi-hop wireless sensor networks, in: Proc. First Int’l Workshop Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS), 2004.

[25] Toumaz Technology Ltd, Oxfordshire, U.K., Ultra low power intelligent sensor interface and transceiver platform. http:// <http://www.toumaz.com/public/page.php?page=sensium_intro>.

[26] M. Ekstrom, Small Wireless ECG Bluetooth Communication to a PDA, [dissertation] Vasteras, Sweden: Dept. Comput. Sci. Electron., Malardalen Univ., 2006.

[27] P. Hamilton, Open source ECG analysis, in: Proc. IEEE Conf. Comput. Cardiol., 2002.

[28] O. Omeni, A.C.W. Wong, A.J. Burdett, and . Toumazou, Energy efficient medium access protocol for wireless medical body area sensor networks, IEEE Trans. Biomed. Circuits Syst. 2(4)(2008) 251-259.

[29] C.-C. Kao, W.-L. Yang, Energy efficient system-on-chip design for wireless body area sensor network, Electric Power Components and Systems 42(7)(2014) 737-745.

[30] A. Goldsmith, Wireless Communications, Cambridge University Press, New York, NY, 2005.