International Journal of Innovative
Computing, Information and Control
Volume **9**, Number **4**, April **2013**

 ISSN 1349-4198
pp. 1715–1725

AN HW RECONFIGURABLE NODE WITH NOVEL SCHEDULING IN AN ENERGY-HARVESTING ENVIRONMENT

Yibin Li, Zhiping Jia, Shuai Xie and Fucai Liu

Department of Computer Science and Engineering
Shandong University
No. 1500, Shunhua Road, Jinan 250101, P. R. China
{ liyibing; zpj }@sdu.edu.cn; { xieshuai1210; liufucai }@mail.sdu.edu.cn

Received February 2012; revised June 2012

Abstract. *One fundamental constraint of wireless sensor network (WSN) is the ratio of the power consumption to the energy supply. Recent studies have shown that building WSN nodes with solar-energy harvesting capability is an effective approach to lengthening the lifetime of node. Meanwhile, the partial dynamic reconfiguration (PDR) is a productive approach in intensive applications such as video and encryption processes. For PDR, time and energy must be invested before an application can be running. Thus, this method is different from the software approach. Therefore, in an energy-harvesting system, the scheduling of tasks in the form of software or hardware is important. In this paper, a novel methodology that schedules dynamic reconfigurations for a WSN node with an energy-harvesting is presented. This method is based on statistical data on tasks and available energy. To demonstrate this approach, an HW reconfigurable WSN node is prototyped. Four typical applications are used as test cases and are divided into basic scheduling units. In the experiments, the efficiency of reconfigurable hardware is first demonstrated. The novel scheduling strategy is then used to identify the most valuable application for the reconfigurable hardware. Our experiments demonstrate that more than 50% energy cost can be saved compared with the software-based solution.*
Keywords: FPGA, WSN, Scheduling strategy, Energy efficiency, PDR

1. **Introduction.** Most current applications in outdoor environments run on batteries given their convenience and relatively low cost. However, battery-powered applications are not suitable for long-term use, despite the numerous studies on the various techniques that lengthen the nodal lifetime [1]. The capture of environmental energy such as solar, vibration and wind is referred to as energy harvesting. Systems that obtain their energy supply through energy harvesting have a regenerative energy source. These systems are fundamentally different from battery-based systems and thus have different design criteria. The power supply of these systems can be unlimited. Figure 1 shows the considerable variability of available energy. However, this energy may be predictable, depending on the characteristics of the energy sources.

For wireless sensor networks (WSN), energy harvesting is a novel approach to overcoming energy issues. Solar energy harvesting is a very dependable source of energy, particularly in an open environment. The perpetual operations can be supported by energy-harvesting functions [3,4]. Various prototype systems with different regenerative sources are available. A previous report [5] presented the history and survey of different types of energy-harvesting systems. In another study [6], a system with vibration energy-harvesting capability was presented. In other studies [7,8], the systems for solar power utilization were described and named as Prometheus and Heliomote in papers. To cope with the variability of available energy in energy-harvesting systems, advanced scheduling strategies have also been studied. A previous study [8] utilized dynamic voltage scaling to

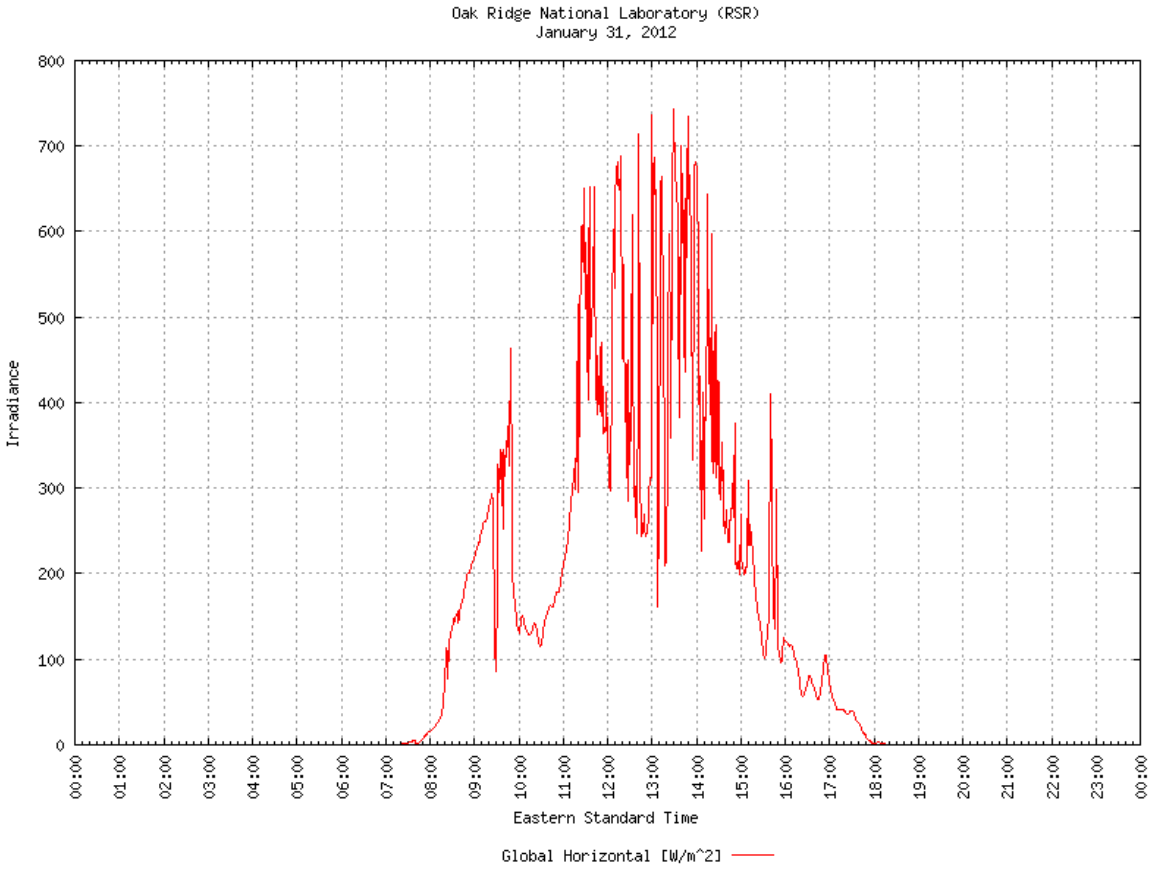


FIGURE 1. Typical global horizontal solar radiation in one day [2]

address the scheduling issue. An online scheduling approach was also developed to solve this problem [9].

Meanwhile, for WSNs, the use of a reconfigurable HW is an effective approach to improving the processing capacity of nodes. Reconfigurability can generally be classified as either static or dynamic. In the static configuration, reconfiguration occurs only when the field-programmable gate array (FPGA) chip is not running. Several studies have been conducted on the utilization and benefit of static reconfigurability through the development of ad-hoc reconfigurable devices [10,11]. To accommodate different networking protocols, reconfigurability was also utilized for the PicoRadio low-power sensor networks [12]. In a previous study [13], a dedicated elliptic curve cryptography (ECC) was developed for WSN nodes. Two systems were built to compare the software-based approach and the reconfigurable HW approach. The FPGA-based ECC implementation requires energy that is three orders of magnitude lower than that used in a low power micro controller implementation. Although these studies demonstrate the benefit of reconfigurable hardware, only static reconfiguration was investigated. In dynamic reconfiguration, when one part of the chip is under reconfiguration, the I/Os and remaining logic remain running. This characteristic allows the regular update of HWs at anytime and saves memory and configuration time compared with static reconfiguration. A number of studies have been conducted on utilizing the PDR approach. In previous papers [14,15], the use of the partial dynamic reconfiguration (PDR) function in the automotive industry and video processing was investigated. In another study [16], a scalable processor architecture central processing unit (SPARC CPU) was incorporated with the PDR function. A piece of remarkable work on WSNs was presented in a previous paper [17]. In this paper, an FPGA-based

WSN node was remotely configured while the node was running. Configuration files were transmitted to the nodes using the Zigbee protocol. The power consumption at different nodal working stages was determined and then analyzed. In this study, the power consumption of data transfers was clearly overwhelming. Therefore, the configuration file was in the node pre-stored. During reconfiguration, only the related control signals were transmitted to enable the reconfiguration process.

For a certain task, although dynamic reconfigurable HWs can deliver better performance with less energy consumption compared with software based solutions, this result may not be obtained when reconfigurations are frequently needed, given that reconfigurable HWs shave different cost models. In this case, an optimized scheduling strategy is needed for a PDR-based system. Therefore, under an energy-harvesting environment, the scheduling of tasks between the processor and the reconfigurable HW becomes vital. This topic was first examined in an energy harvesting WSN node [18], with the assumption that enough energy is always available to carry out the reconfigurations. Furthermore, a novel approach that is based on the statistical analysis of the tasks and the available energy was proposed to reduce missing task deadlines [19]. Although the method demonstrated a reduction in deadline misses by more than 57%, the scheduling algorithm requires large amounts of calculations, thereby making the method unsuitable for embedded applications. For WSN implementation, the proposed strategy aims at reducing the computing complexity and improving the efficiency.

2. **Problem Formulation.**

A good example that illustrates the necessity of this study is described in this section.

Two kinds of given resources are available: processors and FPGAs with reconfiguration costs of 0 and 5. The related energy cost is described in Table 1. Assume that two task lists, task_1 and task_2, are available, with each list containing four tasks. These two lists can be described as $\text{task}_1\{1,1,1,1\}$ and $\text{task}_2\{1,2,1,2\}$, depending on the task type. If every task is executed using a reconfigurable hardware, the total energy cost of the two task lists can be calculated as $\text{Cost}_1 = 5+2+2+2 = 11$ and $\text{Cost}_2 = 5+8+5+8 = 26$. Alternatively, if every task is executed using a software, the energy cost can be calculated as $\text{Cost}_1 = 5+5+5+5 = 25$ and $\text{Cost}_2 = 5+4+5+4 = 18$, respectively.

Table 1. Example cost

Task Type	*Energy Cost of Software Execution*	*Energy Cost of Hardware Execution*	*Reconfiguration Cost*
1	*5*	*2*	*5*
2	*4*	*3*	*5*

In this example, although the reconfigurable HW outperforms the software-based approach for every task type, the same result is not obtained when reconfiguration frequently occurs, given that the energy cost for reconfigurations is not negligible. Therefore, an optimized scheduling for the PDR approach should be adopted.

3. **Method.**

3.1. **Mathematical description of the problem.**

In this paper, the aim of optimization scheduling is to save on the total energy cost of the tasks. Therefore, the energy cost of the task must first be modeled.

Let $S = (t_1, t_2, \ldots, t_n)$ be a series of tasks that is mapped to a certain resource R. S can be divided into five suborders, $S = S_1S_2S_3S_4S_5$. Of these tasks, S_2 and S_4 are nonempty

and only contain task type i. S_1, S_3 and S_5 are also nonempty and contain task types other than i.

If we assume that the cost of the task series S is $C(s)$, then the cost can be divided into three parts: software-based, reconfiguration and HW implementation based costs. In this case, if task type i is implemented by a reconfigurable hardware, then let ρ_i be the energy cost of the reconfiguration and L_2 and L_4 be the hardware based cost for S_2 and S_4, respectively. The cost of the task series can then be described as follows:

$$\begin{aligned} C(S) &= \sum 1 + \sum 2 + \sum 3 + \sum 4 + \sum 5 \\ &= \sum 1 + \rho_i + L_2 + \sum 3 + \rho_i + L_4 + \sum 5 \\ &= 2\rho_i + \sum 1 + L_2 + \sum 3 + L_4 + \sum 5 \\ &= 2\rho_i + X, \end{aligned} \tag{1}$$

where $X = \sum 1 + L_2 + \sum 3 + L_4 + \sum 5$.

In a hardware reconfiguration system, real tasks of the same type do not require reconfigurability. To determine if reconfiguration is feasible, the available energy after a series of task executions should be estimated. The available energy in the future can be calculated using the following equation:

$$Exp(E) = E_{current} - R_j - H_{i,j} + Exp(E_A) \cdot F - (Exp(E_{type \neq j}) + Exp(E_{type=j})) \cdot F \tag{2}$$

In Equation (2), i is the task number, j is the task type number, $E_{current}$ is the current available energy, R_j is the reconfiguration cost associated with the task type j, $H_{i,j}$ is the running cost of task i of type j, $Exp(E_A)$ is the additional energy from energy harvesting, $Exp(E_{type \neq j})$ is the estimated energy cost of the next task that does not belong to type j and is running on software, and $Exp(E_{type=j})$ is the estimated cost of running the next task of type j on the hardware. Both $Exp(E_{type \neq j})$ and $Exp(E_{type=j})$ are scaled based on the likelihood of their occurrence. F is the number of tasks during the analyzed time interval.

$$\begin{aligned} Exp(E_{type=j}) &= \frac{N_j}{\sum_{k=1}^{TT} N_k} H_j \\ Exp(E_{type \neq j}) &= \sum_{l \neq j,\ l=1}^{TN} \frac{N_l}{\sum_{k=1}^{TN} N_k} S_l \end{aligned} \tag{3}$$

In Equation (3), TN is the number of task types, N_j is the occurring number of tasks of type j, S_l is the energy cost of the software implemented task l, and H_j is the energy cost of the hardware implemented task j. Equation (3) is used to calculate the estimated energy cost of future tasks. To estimate the future available energy, the probability of future task must first be obtained. The cost can then be calculated by multiplying the power cost of either the software or hardware implemented task.

The most important characteristic of an energy harvesting system is that the energy can be added over time. To model the added energy, as described in Equation (4), the time gap between two tasks (D) and the estimated available power ($P_{expected}$) are used.

$$Exp(E_A) = P_{expected} \cdot Exp(D) \tag{4}$$

To determine the validity of reconfiguration, the $Exp(E)$ in Equation (2) is used. If $Exp(E)$ is below a threshold or negative, then the associated reconfiguration is considered as unvaluable.

To model the energy source and work load, a nonnegative, continuous and bounded function is designed as a (ρ σ_1 σ_2) function in Equations (5) and (6), only if any of its values are finite positive real numbers τ and T.

These equations can be used for the energy modeling of the workload and the additional energy.

For example, if the energy cost of the workload profile $Pc(t)$ is a (ρ_c σ_1 σ_2) function, then the average rate at which the energy cost is ρ_c and the maximum and minimum are bounded by σ_1 and σ_2, respectively. In a similar manner, the additional energy can be calculated using the same equation.

$$\int_{\tau}^{\tau+T} P(t)dt \leq \rho T + \sigma_1 \tag{5}$$

$$\int_{\tau}^{\tau+T} P(t)dt \geq \rho T - \sigma_2 \tag{6}$$

3.2. **Proposed strategy.**

Most-used-in-the-future (MUF) strategy

Our scheduling strategy primarily consists of two steps. First, a task window is built based on the proposed model. If $Exp(E)$ is non-negative or exceeds the threshold value, then the task can be added into the task window. Second, a most used in the future (MUF) strategy is adopted to select the most valuable task for reconfigurable hardware implementation; this strategy can be described by Equation (7).

In the task window, if task type j is implemented, the use of the reconfigurable HW depends on

$$Max\left(\frac{\sum_{l=j,k=1}^{I} Exp(E_{k,l})}{\sum_{l=1}^{TN}\sum_{k=1}^{NL} Exp(E_{k,l})}\right) \tag{7}$$

where I is the number of tasks that belong to type j, TN is the number of task type, and NL is the number of tasks of type l.

3.3. **Other strategies for evaluation.** In this paper, extensive simulations have been performed to demonstrate the effectiveness of our approach. For comparison, four strategies are used.

All Software

In this case, all test cases are run on software when the energy is sufficient.

All Configurable Hardware

When the energy is sufficient, the all configurable hardware approach always runs the tasks on hardware.

Random

In the random approach, the tasks have a 50% chance to run on configurable HW. If the random approach does not run the task on the hardware, then it is run in the form of software. If the energy needed to perform the reconfiguration is insufficient, then the approach attempts to run the task on software.

MUF Strategy

As previously illustrated, in this method, a task window is built based on whether the energy consumption of all tasks in the window can keep the future energy below a threshold value. Given the number of tasks to be executed, these tasks are then queued. Only the task with the highest probability of being executed is performed using reconfigurable HW. The rest of the tasks are executed on software.

3.4. **Test case and task.** To evaluate the applicability of FPGA in a WSN, proper applications need to be considered. Although WSN environments are dynamic in nature, the basic tasks performed in a WSN node are quite similar. Typically, they include sensing a certain phenomenon, gathering the relevant data and transferring pre- or post-processed information to a base-station or sink node. Several attempts were made to profile the workload of the generic WSN node [20,21]. For instance, the application of the WSN node was categorized into six classes [21]. In this paper, considering the nature of the applications and their availability, finite impulse response (FIR) secure hash algorithm 2 (SHA-2), fast fourier transform (FFT) and advanced encryption standard (AES) have been selected to evaluate the applicability of FPGA for the WSN node and the proposed approach.

Of these applications, FIR is a relatively small application. 16-bit input precision and 8-bit coefficients are used in this design. The security of WSN is a critical issue in WSN application [22] because the WSN nodes are deployed in open areas. AES is widely used to protect sensitive information over transmissions. In the AES core, a 128-bit key is chosen. SHA-2, which has better security than MD5, is used to check the integrity. A hardcore for SHA-2 is used in accordance with a previous study. FFT is a well-known algorithm in the DSP field and is used to transform time domains to frequency ones. In this paper, a 16-bit hard core is developed for the 1024-point FFT implementation using a commercial electronic system level (ESL) tool.

In the four test cases, the input data can be divided into bit-levels. Therefore, the input data is split into 256-bit blocks as the basic scheduling unit for each application. A Gaussian distribution is proposed to simulate the harvested energy.

4. Prototype Architecture and Design Flow.

4.1. **Proposed node architecture.** A sensor node prototype is built to facilitate the novel scheduling strategy of the reconfigurable HW (Figure 2). Virtex 4 from Xilinx is selected for the reconfigurable HW because it can facilitate the PDR function and has the capacity to accommodate large designs. To carry out the experiment all tasks can be run in using hardware or software. Therefore, a relatively powerful CPU is needed to execute large applications. In our prototype, an open-source SPARC V8-based processor

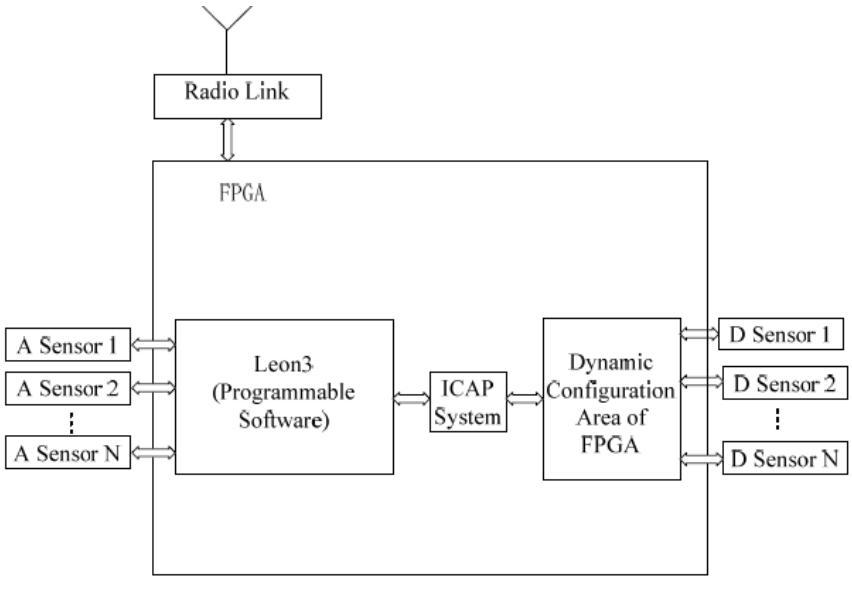


FIGURE 2. Proposed node architecture

is used with Virtex 4. Although the Leon processor is not initially intended for low-power implementations, it is chosen for two reasons. First, it is supplied with soft core and related tools which make it suitable for custom designs. Second, it has the necessary ability to process the proposed test cases. The configuration memory of the Virtex-4 family consists of frames which are the basic fabric units. The adopted FPGA chip of the node contains a total of 11070 frames, including 10410 configurable frames and 660 non-configurable frames.

The internal configuration access port (ICAP) carries out the reconfiguration. Through ICAP, the internal registers and the configuration memory can be read and written. The ICAP wrapper and the ICAP itself are contained in a module called the ICAP hardware system. To work with the Leon core, ICAP is integrated with the rest of the WSN node as a slave to the APB. The ICAP wrapper is the module containing the hardware controllers that allow the interaction between the software and ICAP.

To enable the ZigBee prototype, the ETRX2 module from Telegesis which is based on ZigBee and a low-power implementation, is adopted. The module is connected to the processor through a universal asynchronous receiver transmitter (UART) port.

4.2. **Partial design flow.** To develop a PDR system, the FPGA chip is first divided into reconfigurable and fixed regions. This process is referred to as FPGA structuring, resource arrangement or partitioning [23]. The PDR design and work flow are briefly introduced in Figure 3. In this design flow, the supported granularity for modeling and reconfiguration is the first to be considered. The selected reconfiguration granularity is coarse, given that a full-IP core is loaded into the FPGA when the reconfiguration occurs. During FPGA reconfiguration, these cores are loaded in the form of placed and routed designs (partial configuration files) also called hard cores [24]. Virtual architecture (VA) models can be one-dimensional (1D) and two-dimensional (2D). In this paper, 1D-based VAs can satisfy the node requirements. When all VA design-related issues have been addressed, the hard cores are transferred into a set of files which include a user constraint file (.ucf file) and a communication file (.nmc file). In the constraint file the slot positions and boundaries are defined. Chip communication is defined as placed macros.

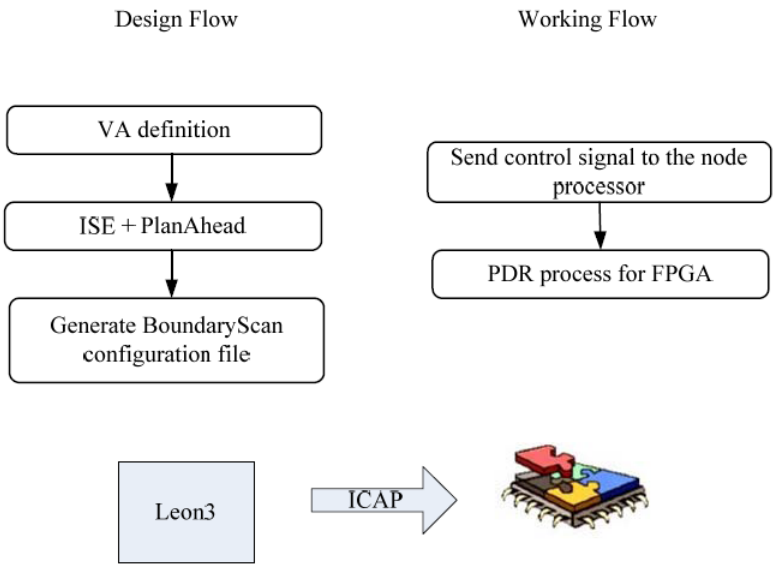


FIGURE 3. Design and work flow for the wireless sensor network (WSN) node with a partial dynamic reconfiguration (PDR) function

After all related files are defined, the hard cores are then developed. In our work, the design flow is based on ISE and the Plan Ahead tool. The advantage of this design is a more efficient routing in most cases as well as the direct production of partial configuration files.

5. **Results and Discussion.** To evaluate the time and energy cost of reconfiguration, the power and timing information are collected during the experiment. The time cost then is measured using the cycle count of the system. The cycle counts for each hard core configuration are first collected. The time information is then converted into seconds. The power consumption of the configuration process is then multiplied to obtain the energy cost. Table 2 shows that for the HW reconfiguration system, the power and time costs of reconfiguration are not trivial compared with the task execution. Therefore, the "all hardware"approach is clearly not the best strategy.

Table 2. PDR energy consumption

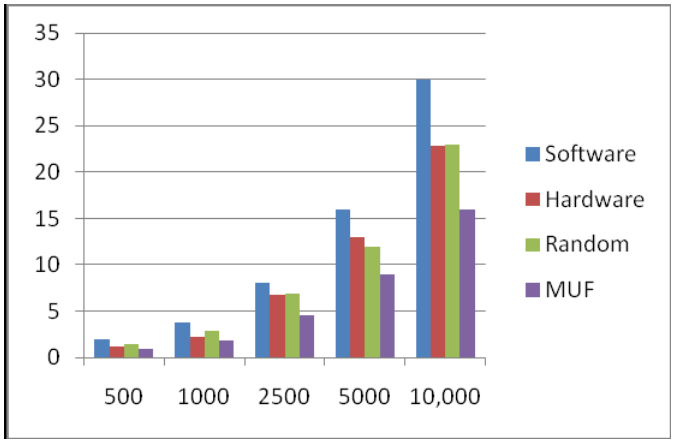
	Clock cycles to PDR	*Average Power During PDR (W)*	*PDR duration (sec)*	*PDR Energy (J)*
AES	*612,891,596*	*0.715*	*12.21*	*8.73*
FIR	*26,195,440*	*0.715*	*0.52*	*0.372*
FFT	*778,036,147*	*0.715*	*15.50*	*11.08*
SHA-2	*471,378,000*	*0.715*	*9.39*	*6.71*

Table 3. Power analysis (mW)

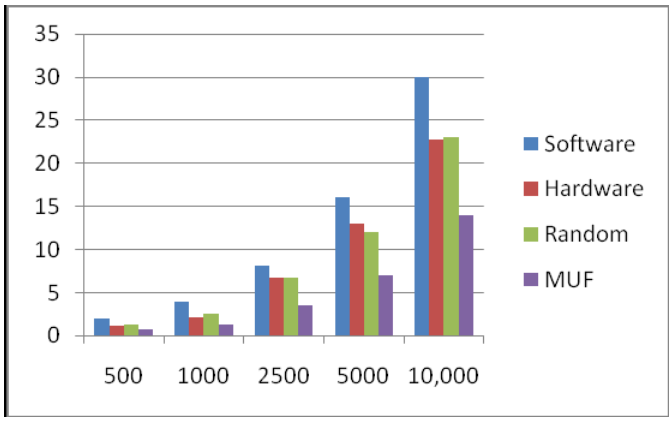
	Static Power	*Average Runtime Power (Software)*	*Average Runtime Power (Reconfigurable Hardware)*	*Application Power Reduction*
Leon3 Static&Blank	*620*			
Leon3&AES	*629*	*1267*	*932*	*53%*
Leon3&FIR	*623*	*681*	*660*	*36%*
Leon3&FFT	*633*	*2033*	*1204*	*60%*
Leon3&SHA-2	*628*	*831*	*723*	*53%*

Table 3 shows the experimental power consumption results. The static power of the Leon core is approximately 620 mW. When the test cases (except for the FIR case, which is a relatively small application) are run in the software form, the power consumption is significantly increased. Compared with the software implementation, the reconfigurable HW can save 31 mW to 829 mW power for each case. Thus, runtime energy consumption savings are demonstrated after the adoption of a reconfigurable HW. Considering the shorter running time of the HW implementation compared with that of the software implementation, the reconfigurable HW exhibits both time and power advantages over the software approach.

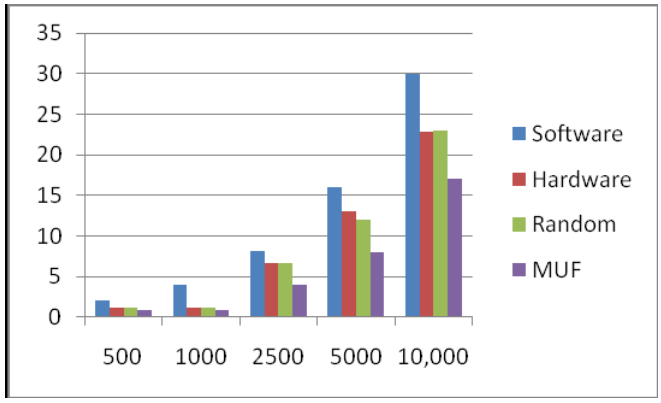
The energy costs of the different strategies are shown in Figures 4(a)-4(c). The window sizes are 10, 50, and 100 across all strategies. In these figures, the Y axis represents the energy cost of the different strategies in J. In the different figures, the hardware and software energy costs between different task numbers remain the same. The reconfigurable hardware shows an advantage over all software strategies in terms of the energy cost across all test cases. With the scheduling strategy, the energy cost can be further reduced. The



(a) Energy consumption for window size 100



(b) Energy consumption for window size 50



(c) Energy consumption for window size 10

FIGURE 4. Energy costs of the experiments

MUF strategy shows the highest energy efficiency compared with other strategies; the energy cost for window 50 is minimal.

6. **Conclusions.** We have presented a novel scheduling strategy for heterogeneous WSN nodes under an energy harvesting environment. Theoretical models for an energy harvesting node and for heterogeneous power consumption are built and then analyzed. A novel reconfiguration strategy is then proposed. In this method, a task is first proposed according to the available energy. The most frequently used task inside a task window is then implemented using a reconfigurable HW. This method enables the utilization of

environmental energy with harvesting awareness. The task allocation can be scheduled according to energy availability. In this approach, energy can be saved because only the most frequently used tasks are executed using the hardware. To demonstrate the proposed scheduling strategy, a PDR functioned prototype is proposed with the associated design flow. Four typical applications are used as test cases for evaluation. The benefits of the reconfigurable HW are demonstrated in our work. By adopting our strategy, the energy cost of the application can be reduced by up to 50% compared with all software strategies. In addition, the proposed method can deliver a performance comparable with those of existing strategies but with less computing complexity.

Energy capture from the environment is a competitive, highly practical approach to addressing energy constraints. The reconfigurable HW-based heterogeneous system is an effective method of increasing the processing ability of systems but at lower energy costs. Further studies on energy-effective communication mechanisms in energy-harvesting environments will be conducted in the future.

Acknowledgments. We want to thank the helpful comments and suggestions from the anonymous reviewers. This research is financially supported by International Science & Technology Cooperation Program of China (2011DFR20090), Public Science and Technology Research Funds Projects of Ocean (201205036-04), Shandong Province Research Program (2010GcG20101, 2011GGH10132), IT Development Fund of Shandong Province (2010R1501) and Independent Innovation Foundation of Shandong University (IIFSDU 2010TB020).

REFERENCES

[1] Y. Obashi, H. Chen, H. Mineno and T. Mizuno, An energy-aware routing scheme with node replay willingness in wireless sensor networks, *International Journal of Innovative Computing, Information and Control*, vol.3, no.3, pp.565-574, 2007.

[2] *Oak Ridge National Laboratory (ORNL) RSR web site*, http://www.nrel.gov/midc/ornl_rsr/, 2012.

[3] A. Kansal, J. Hsu, S. Zahedi and M. B. Srivastava, Power management in energy harvesting sensor networks, *ACM Transactions on Embedded Computing Systems*, vol.6, no.4, 2007.

[4] X. Jiang, J. Polastre and D. Culler, Perpetual environmentally powered sensor networks, *Proc. of the 4th International Symposium on Information Processing in Sensor Networks*, Los Angeles, CA, USA, pp.463-468, 2005.

[5] J. A. Paradiso and T. Starner, Energy scavenging for mobile and wireless electronics, *IEEE Pervasive Computing*, vol.4, no.1, pp.18-27, 2005.

[6] Y. Ammar, A. Buhrig, M. Marzencki, B. Charlot, S. Basrour and M. Renaudin, Wireless sensor network node with asynchronous architecture and vibration harvesting micro power generator, *Proc. of sOc-EUSAI*, Grenoble, France, pp.21-28, 2005.

[7] *HelioMote Project*, http://research.cens.ucla.edu/portal/page?_pageid=56,55124,56_55125&_dad=portal&_schema=PORTAL.

[8] B. A. Allavena, A. Allavena and D. Mossé, Scheduling of frame-based embedded systems with rechargeable batteries, *Proc. of IEEE Workshop on Power Management for Real-Time and Embedded Systems*, Taipei, Taiwan, pp.12-18, 2001.

[9] C. Rusu, R. Melhem and D. Mossé, Multi-version scheduling in rechargeable energy-aware real-time systems, *Proc. of IEEE Euromicro Conference on Real-Time Systems*, Porto, Portugal, pp.12-24, 2003.

[10] H. Hinkelmann, P. Zipf and M. Glesner, Design concepts fora dynamically reconfigurable wireless sensor node, *Proc. of the 1st NASA/ESA Conf. Adaptive Hardware and Systems*, Istanbul, Turkey, pp.436-441, 2006.

[11] E. Susu, M. Magno, A. Acquaviva and D. Atienza, Reconfiguration strategies for environmentally powered devices: Theoretical analysis and experimental validation, *Transactions on High-Performance Embedded Architectures and Compilers I*, 2007.

[12] T. Tuan, S. F. Li and J. Rabaey, Reconfigurable platform design for wireless protocol processors, *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City, UT, pp.893-896, 2001.

[13] J. Portilla, A. Otero, E. de la Torre et al., Adaptable security in wireless sensor networks by using reconfigurable ECC hardware coprocessors, *International Journal of Distributed Sensor Networks*, vol.2010, 2010.

[14] J. Becker et al., Dynamic and partial FPGA exploitation, *Proc. of IEEE*, vol.95, no.2, pp.438-452, 2007.

[15] P. Sedcole, Y. K. Peter, C. George, A. Constantinides and W. Luk, Run-time integration of reconfigurable video processing systems, *IEEE Transactions on VLSI Syst.*, vol.15, no.9, pp.1003-1016, 2007.

[16] I. Zaidi, A. Nabina, C. N. Canagarajah and J. Nunez-Yanez, Evaluating dynamic partial reconfiguration in the integer pipeline of a FPGA-based opensource processor, *Proc. of FPL*, Heidelberg, Germany, pp.547-550, 2008.

[17] Y. E. Krasteva, J. Portilla, E. de la Torre and T. Riesgo, Embedded runtime reconfigurable nodes for wireless sensor networks applications krasteva, *IEEE Sensors Journal*, vol.11, no.9, pp.1800-1810, 2011.

[18] I. Folcarelli, A. Susu, T. Kluter, G. De Micheli and A. Acquaviva, An opportunistic reconfiguration strategy for environmentally powered devices, *Proc. of the 3rd Conference on Computing Frontiers*, Urbino, pp.1200-1208, 2006.

[19] A. Nahapetian, P. Lombardo, A. Acquaviva, L. Benini and M. Sarrafzadeh, Dynamic reconfiguration in sensor networks with regenerative energy sources, *Proc. of DATE Nice*, France, pp.1-6, 2007.

[20] L. Nazhandali, M. Minuth and T. Austin, SenseBench: Toward an accurate evaluation of sensor network processors, *Proc. of IISWC*, Austin, TX, pp.197-203, 2005.

[21] S. Mysore, B. Agrawal, F. Chong and T. Sherwood, Exploring the processor and ISA design for wireless sensor network applications, *Proc. of VLSI*, Hyderabad, India, pp.59-64, 2008.

[22] R. C. Chen and S. P. Chen, Intrusion detection using a hybrid support vector machine based on entropy and TF-IDF, *International Journal of Innovative Computing, Information and Control*, vol.4, no.2, pp.413-424, 2008.

[23] P. Sedcole, B. Blodget, T. Becker, J. Anderson and P. Lysaght, Modular dynamic reconfiguration in Virtex FPGAs, *IEE Proc. of Comput. Digit. Tech.*, vol.153, no.3, pp.157-164, 2006.

[24] E. L. Horta and J. W. Lockwood, Automated method to generate bit stream intellectual property cores for Virtex FPGAs, *Proc. of FPL*, Leuven, Belgium, pp.975-979, 2004.