This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

Using Reconfigurable Hardware Devices in WSNs for Reducing the Energy Consumption of Routing and Security Tasks

Georgios-Grigorios Mplemenos, Konstantinos Papadopoulos and Ioannis Papaefstathiou
Department of Electronic and Computer Engineering
Technical University of Crete, Chania, Greece
Email: {mplemenos, papadopoulos, ygp}@mhl.tuc.gr

***Abstract*—As Wireless Sensor Networks (WSNs) expand their reach, their applications require lower energy consumption than those provided by current offerings. In this paper, a novel platform is introduced, which employs a reconfigurable device (CPLD) in order to enhance the processing power of typical sensor nodes and, more importantly, reduce the overall energy consumption in common tasks such as routing and security. Our real-world measurements, demonstrate that the proposed system can reduce the energy consumption of the Cost Estimation algorithm of the widely used XMesh routing protocol by 71.5%. Higher energy conservations (more than 90%) can be achieved at Blowfish Encryption, when this is implemented on our new platform.**

I. Introduction

Wireless Sensor Networks (WSN) have recently attracted great interest due to the diversity of their applications covering numerous areas such as home automation, health care, and environmental monitoring [1]. Each WSN consists of many interconnected nodes (namely motes), whose primary tasks are the sensing of the surroundings, the processing the of measurements and the wireless transmission of the sensed data to a central node (sink). Probably the most crucial characteristic of a mote is its energy consumption. Typically, nodes are powered by small batteries and replacing those energy sources on possibly thousands of nodes could become infeasible or even impossible in certain cases (e.g. environmental monitoring). Hence, it is well accepted that one of the key challenges in unlocking the potential of such data gathering sensor networks is conserving energy so as to maximize their post-deployment active sensing lifetime.

WSNs are mostly organized in multi-hop or ad-hoc topologies. Routing is challenging due to the inherent characteristics that distinguish these networks from other wireless ones like mobile ad-hoc networks. The task of finding and maintaining routes in WSNs is non-trivial since energy restrictions and sudden changes in node status (e.g. failure) cause frequent and unpredictable topological changes. To minimize energy consumption, routing techniques proposed in the literature, employ some well-known routing tactics as well as tactics special to WSNs, such as data aggregation and in-network processing, clustering, different node role assignment and data-centric methods [1].

At the same time, security is a critical factor in numerous WSN applications. However, these tiny, pervasive computing devices have extremely limited resources and computational capabilities. Thus, security engineers face the seemingly contradictory challenge of providing lightweight algorithms for strong authentication, encryption and other cryptographic services that can perform on a speck of dust.

Performing routing and security tasks fast and efficiently (in terms of energy consumption) is crucial for the overall performance of a wireless sensor network, as they represent some of the most frequently used functionality. We apply the general census (yet novel in the WSN field) that hardware implemented tasks can provide higher levels of performance with reduced energy consumption than their software equivalents and we propose a platform that uses reconfigurable hardware to accelerate the routing and security algorithms. Reconfigurable hardware offers the required flexibility to build a general-purpose node, which can support arbitrary algorithms for the aforementioned functionality. If the concept is generalized, then the same infrastructure could be used to accelerate other more demanding tasks according to application requirements. This paper evaluates the performance of the platform when implementing a widely used WSN routing protocol (XMesh) and a cipher encryption scheme (Blowfish). In order to measure the efficiency of our approach, we carried out real-world experiments with commonly used WSN nodes (Crossbow's IRIS motes) connected to Xilinx CoolRunner-II CPLDs. The proposed scheme reduces the measured energy of the Cost Estimation algorithm of the XMesh Routing Protocol by 71.5%, while the Blowfish encryption achieves even higher energy gains ($\sim$93%).

The main contributions of this paper come from the fact that it is the first time that such low cost reconfigurable devices (CPLDs) are utilized in the networking/processing tasks in a WSN environment. The results from our real-world experiments show the potential of such a design approach, since the ever important battery life of WSN nodes can be extended while the limited processing power is increased.

The rest of the paper is organized as follows. Section 2 describes related work on the field of wireless nodes that employ reconfigurable devices. Section 3 provides a background on the applications implemented in our system, whose architecture is discussed in Section 4. Section 5 presents the

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

experimental process to evaluate the platform and its results. Finally, conclusions are drawn in Section 6.

II. Related Work

There are several groups working in developing efficient sensor nodes worldwide; however, only a limited number supports reconfigurable devices. In particular Tyndall Institute [2] and CEI-UPM [3] have developed two distinct platforms for wireless sensor networks which are stackable and modular. Those platforms can support a Field Programmable Gate Array (FPGA) module, which is connected in such a way that only fast DSP processing can be executed on the reconfigurable device. Using the same approach, Microsoft [4] has developed a modular platform called mPlatform. Its philosophy is modular, and they include reconfigurable hardware in order to reconfigure the interface protocols between the different layers of the modular platform as well as to adapt them to the requirements of the different layers. Those reconfigurable hardware resources are used so as to allow the plugging of different sensors/actuators to the main processing node; they can only execute different board-level intercommunication protocols and they cannot execute any data manipulation tasks neither any WSN communication protocols.

In comparison with all those approaches, the proposed system can execute various data manipulation algorithms and networking tasks on the reconfigurable hardware. It should be stressed that there does not exist any research work, to the best of the authors' knowledge, utilizing CPLDs for the network/data processing tasks involved in a WSN environment.

III. Background

A. Multihop Routing Protocols and XMesh Protocol Stack

Multi-hop or ad hoc, wireless networks use two or more wireless hops to convey information from a source to a destination. There are many Multihop Routing protocols for WSNs that try to adopt a different approach for routing, energy management and overall latency management. TinyOS [5], widely adopted OS for WSN nodes, allows users to wire-in different protocols with minimal effort. A Multihop protocol can be executed on each Mote, while each Mote can serve both as a data source and as a router. There are three different routing protocols supported by TinyOS 1.x which differ in terms of both the actual routing algorithm and the services they provide. In particular, *Route* seeks to minimize the number of hops that each packet traverses while *MINTRoute* and *ReliableRoute (XMesh)* route packets based on link-quality estimates that seek to maximize the probability of a packet being delivered [6], [7], [8], [9], [10], [11], [12]. The most widely used such routing scheme is the XMesh due mainly to its performance when applied in real-world WSNs.

B. Blowfish Encryption

Blowfish [13] is a secret-key block cipher proposed by Bruce Schneier. The block size is 64 bits, and the key can have any length up to 448 bits. Although there is a complex initialization phase required before any encryption can take place (i.e. key expansion), the actual encryption task is very efficient in terms of performance and energy consumption [13]. Key expansion converts a key of at most 448 bits into several subkey arrays totaling 4168 bytes. Data encryption is implemented via a 16-round Feistel network. Each round consists of a key-dependent permutation, and a key- and data-dependent substitution. All operations are XORs and additions on 32-bit words, while four indexed array data lookups are required per round.

There are also some smaller-block versions of Blowfish, such as Blowfish-16 which has a 16-bit block size. Its small block size makes this version of Blowfish perfectly suitable for WSNs while taking into account the limited resources of a CPLD, Blowfish-16 seems a good candidate for such CPLD-based network nodes.

IV. System Architecture

Before proceeding to the presentation of our platform, a reasoning behind the choice of the CPLD device should be given. To form a platform that uses hardware acceleration the primary choices are three: use a custom-designed ASIC device, a field-programmable gate array (FPGA) or a CPLD. The greatest performance and energy efficiency can no doubt be achieved with the first solution. However, it is the less flexible one, since it can be used for a single application/algorithm and, as a result, it is unsuitable for a general-purpose node. Furthermore, the cost associated with the design and production of such a device is prohibitive if the volumes are low.

On the other hand, among the reconfigurable devices (FPGAs and CPLDs), FPGAs can not only offer higher performance but also their increased resources can accommodate larger designs. These advantages come at the cost of higher energy consumption and, in most cases, higher prices. Another important advantage of the CPLDs over the FPGAs is the fact that they can retain their configuration when powered-off, which allows them to be programmed on lab or factory facility prior to being deployed, while the FPGAs need some form of non-volatile memory (i.e. flash memory) to hold their configuration data. The flash memory chips add to the Bill of Materials (BOM) of the sensor node and require precious PCB space which leads to larger final products. So, unless the requirements of the intended WSN application cannot be met by a CPLD device (and as it will be shown, in the applications we examine in this paper this is not true), they constitute the best choice.

A. Reference Node Design

In order to implement our prototype sensor platform, we connected a CPLD development board to a widely used sensor node. To be more specific, we used the Digilent X-Board [14], which hosts a Xilinx CoolRunner-II CPLD [15] along with the required circuitry to connect the reconfigurable device to peripherals and host systems. The CPLD on-board is a low-cost 256 macrocell CoolRunner-II CPLD (XC2C256) in a TQ-144 package.

The sensor node used is Crossbow's IRIS mote [16], which is

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

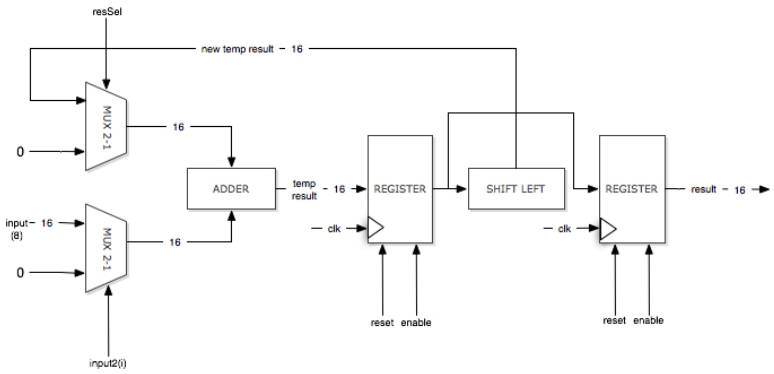

Fig. 1. Multiplication Block Diagram

a device designed around an ATmega 1281 micro-controller with an IEEE 802.15.4 compliant radio frequency transceiver to provide wireless connectivity. Through the MDA100 sensor and data acquisition board [17], which includes a number of sensors and a general prototype area, the mote's micro-controller is connected to the CPLD board. The development of custom sensor applications is enabled through Crossbow's MoteWorks software platform. The sensor devices have an embedded network stack (*XMesh* in our case), operating system (*TinyOS*) and transmission modules (802.15.4). For the programming of the sensor nodes we have utilized the nesC language. NesC (network embedded system C) is a component-based, event-driven programming language used to build applications for the TinyOS platform. On the hardware design side, we employed Xilinx's ISE 10.1 for synthesis and design implementation, while ModelSim SE 6.3 was used for logic and timing simulations. Finally, the CPLD was programmed using the Digilent ExPort toolset.

B. XMesh's Cost Functions Architecture

In order to form and maintain a Mesh network, the following two parallel processes are involved: **Link Estimation and Parent Selection**. These processes estimate the following metrics:
Receive Estimate (RE), Send Estimate (SE), Link Cost (LC), Neighbor's Cost (NC) and Overall Cost (OC). These metrics are calculated from the following equations and are described in [10]:

$$Est = 255 \times received/(received + missed)$$
$$RE = (1 - alpha) \times RE + alpha \times Est$$
$$LC = (1 \ll 18)/(SE \times RE)$$
$$OC = LC + NC$$

From the previous cost metric functions, the multiplications were chosen to be implemented on the reconfigurable device, in order to accelerate the performance of the routing protocol and to reduce the energy consumption. At this point, it should be mentioned that implementing all the cost metric functions on the CPLD, even though it decreases the energy consumption, it also increases the whole execution time of the system, since after experimentation, it was found that implementing a divider in a CPLD is very slow; obviously if a larger CPLD is utilized, in which a high-speed divider can be realized, the complete cost function will be mapped to the CPLD resulting in greater energy savings and performance improvements. Besides that, the reconfigurable multiplication unit is utilized in other tasks of the protocol as well, since there are a number of related metrics that are calculated using only multiplication. In Figure 1, the block diagram of the multiplier that was implemented on the CPLD, is demonstrated. In Table I the resource utilization of the overall CPLD design is reported.
For the implementation of this design, a multiplication is realized through a shift and add scheme. Based on the multiplication algorithm implemented, a multiplication is completed in nine cycles, one for each bit of the second multiplicand plus an additional cycle that is needed in order to check and fetch the result from the last output register.

C. Blowfish Encryption

The main reason behind using a mini version of Blowfish encryption is the fact that the data exchanged among the nodes of a WSN usually have a smaller size than the block size of the original algorithm being 64 bit. The majority of the sensors included in the most widely-used sensor and data acquisition boards such as the Crossbow MDA100 used for our platform export a 16-bit digital output that corresponds to the value of the specific measurement (i.e., temperature, light, e.t.c.). This is why we prefer having a cipher encryption scheme with 16-bit block size to exchanging 64-bit quantities with many unused data bits and, consequently, useless information that would burden the communication channel.
Our system implements only the encryption task since (a) it is a very CPU intensive task, (b) the procedure of the encryption is more popular than the decryption one in a sensor network (i.e. each node encrypts the collected data, whereas they are decrypted only in their destination) and (c) the block cipher encryption tasks consume, based on our measurements, the majority of the overall *sending message* energy.

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

TABLE I
CPLD RESOURCE UTILIZATION

Resources	XMesh Routing	Blowfish Encryption
Macrocells	94/256 (37%)	118/256 (47%)
Pterms	451/896 (51%)	613/896 (69%)
Registers	67/256 (27%)	71/256 (28%)
Pins	18/118 (16%)	28/118 (24%)
Function Block Inputs	248/640 (40%)	271/640 (43%)

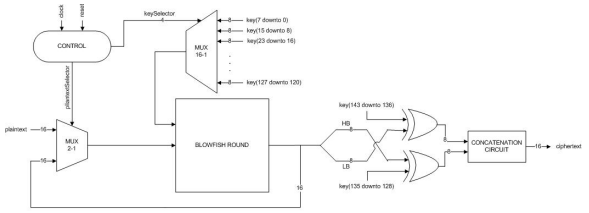

Fig. 2. Blowfish encryption block diagram

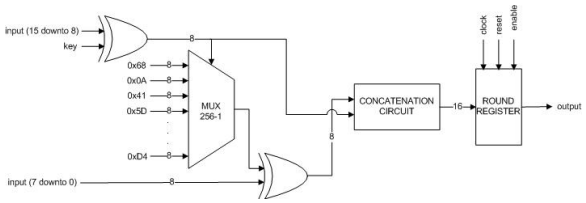

Fig. 3. Blowfish round block diagram

(Round1) 1st round of Blowfish encryption – enable round register, plaintext selection, 1st key byte selection

(Round2) 2nd round of Blowfish encryption – enable round register, previous result selection, 2nd key byte selection

(Round3) 3rd round of Blowfish encryption – enable round register, previous result selection, 3rd key byte selection

...

(Round16) 16th round of Blowfish encryption – enable round register, previous result selection, 16th key byte selection, enable output

Fig. 4. Blowfish procedure

As mentioned above the encryption process of Blowfish consists of 16 rounds and its input is a 16-bit data element, X.

The data encryption algorithm is shown below:

Divide X into two 8-bit halves: X_L, X_R
For i = 1 to 16:
X_L = $X_L \oplus P_i$
X_R = $F(X_L) \oplus X_R$
Swap X_L and X_R
Next i
Swap X_L and X_R (Undo the last swap)
X_R = $X_R \oplus P_{17}$
X_L = $X_L \oplus P_{18}$
Recombine X_L and X_R

Figure 2 presents the block diagram of the presented Blowfish encryption implementation.
The presented implementation executes only one of the rounds of the encryption algorithm, which is represented by the *Round* box and implemented as shown in Figure 3, and the last four steps of the algorithm (one swap, two XOR calculations and the concatenation of the two halves in order to form the final result of encryption). So the use of a finite state machine (FSM) is necessary for securing the right operation of the system and executing the remaining rounds of the algorithm by re-utilizing the existing blocks. This FSM implements the Blowfish procedure as described in Figure 4.

V. PERFORMANCE

The system was evaluated based on three major metrics: execution time, energy consumption and maximum power draw. All these are critical parameters in WSNs, since it is certainly desirable to increase the limited processing power of the node while also increasing the battery lifetime of the wireless mote by lowering its energy consumption and maximum power draw.

Our results are based on real world experiments in which a mixed signal oscilloscope has been used in order to take the speed, energy and power measurements. An extra signal has been utilized in both the software and the hardware schemes of both implemented applications in order to measure the execution time; this signal transits to high when the execution of the specific process starts and then toggles back to low when the process ends. Furthermore, the energy consumption is calculated using the integral of the measured voltage V_m for the measured execution time period $\Delta\tau$. The result is divided with the reference resistance R_{ref}, which is equal to 0.5 Ω in the experimental circuits used, in order to calculate the reference current I_{ref}. Multiplying I_{ref} with the reference voltage V_{ref}, which is equal to 3.0 V for the Mote and 3.3 V for the CPLD, the overall energy consumption is calculated, based on formula (1).

$$E = I_{ref} V_{ref} \quad \text{where} \quad I_{ref} = \frac{\Sigma_i V_{m,i} \Delta\tau}{R} \tag{1}$$

The overall measured values for both applications, are presented in Table II.
Based on our experiments (the CPLD was clocked at 48Mhz), the measured energy consumption when the aforementioned function of the first described application is executed on our platform, is reduced by 71.49%, compared to the software-based approach. Simultaneously, there is a decrease of the measured maximum power draw by 49.2%. At this point it should be mentioned that, due to the limited resources of the reconfigurable device, only the multiplication is executed on the CPLD, while the rest of the cost function is executed on the mote. As Table II clearly demonstrates, the execution time remains almost the same in both cases due to the bottleneck that is introduced by the communication protocol between the Mote and the CPLD device. It was measured that the Cost Estimation Function when this executed on the CPLD device, without the Mote connection, takes about $0.206us$ while the

TABLE II
PERFORMANCE RESULTS

Application	System	Execution Time (us)	Energy Consumption (uJ)	Maximum Power Draw (mW)
XMesh Routing	IRIS Mote	125.000	85.600	1010.000
	IRIS Mote plus CoolRunner-II	123.000	24.400	513.000
			Reduction: 71.49%	**Reduction**: 49.20%
Blowfish Encryption	IRIS Mote	75.000	320.000	446.900
	IRIS Mote plus CoolRunner-II	79.800	21.600	36.100
			Reduction: 93.3%	**Reduction**: 91.9%

energy consumption is significantly lower ($0.216uJ$).
Regarding the second application, the Blowfish encryption process on the CPLD was about 179 times faster than when executed on the micro-controller of the IRIS Mote. Especially, the time used for calculating the ciphertext is 0.42 us when the encryption is implemented on hardware instead of 75 us used by the software implementation. On top of that, the overall energy consumption for the calculation of the ciphertext is reduced by 99.9% when the hardware scheme is utilized ($0.1uJ$ are consumed by the hardware implementation of Blowfish instead of $320uJ$ consumed by the software one).
Connecting the reconfigurable device with the IRIS mote causes an insignificant increase in the execution time due to the overhead triggered by the communication of the two devices (the same applies to the previous application), but even in this case there is also a significant decrease in the values of the overall energy and power consumption. To be more specific, energy consumption is decreased by 93.3% and maximum power draw by 91.9%, when compared with the conventional CPU-based approach. These results prove that the utilization of a CPLD in a WSN mote is an efficient approach for this cipher encryption application.

VI. CONCLUSIONS

In this paper, a novel WSN node architecture is proposed which combines a low-cost, low-power reconfigurable hardware device (CPLD) with a typical sensor node, in order to reduce the energy consumption of certain routing and security tasks. Through our real-world experiments we achieved a significant energy consumption reduction of 71.49% and 93.3%, for XMEsh Cost Estimation task and Blowfish encryption, respectively. For a WSN infrastructure to be successful, it has to meet an application's performance requirements with as little energy consumption as possible, so that its battery operated nodes can remain functional for longer periods of time. The prototype presented in this paper achieves the same levels of performance with typical current offerings, while being much more energy efficient. However, this is a limitation posed by the prototype nature of our components and it can be addressed in a final production system. In this case, significantly better performance can be achieved, while we expect the energy consumption numbers to drop even more.

ACKNOWLEDGMENTS

This work is part of the Ad-hoc PAN and WIreless Sensor SEcure NETwork (AWISSENET) research project, implemented within the Seventh Framework Programme and financed by Community Funds (Information Society Technologies of European Union).

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.
[2] B. O'Flynn, S. Bellis, K. Delaney, J. Barton, S. O'Mathuna, A. Barroso, J. Benson, U. Roedig, and C. Sreenan, "The development of a novel miniaturized modular platform for wireless sensor networks," *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pp. 370–375, April 2005.
[3] J. Portilla, A. de Castro, E. de la Torre, and T. Riesgo, "A modular architecture for nodes in wireless sensor networks," *in journal of universal computer science*, vol. 12, no. 3, pp. 328–339, 2006 2006.
[4] D. Lymberopoulos, N. Priyantha, and F. Zhao, "mplatform: A reconfigurable architecture and efficient data sharing mechanism for modular sensor nodes," *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pp. 128–137, April 2007.
[5] *Tinyos Online Tutorial*, http://www.tinyos.net, September 2008.
[6] *XMesh Users's Manual*, Crossbow Technologies, March 2007, revision C.
[7] A. Teo, G. Singh, and J. McEachen, "Evaluation of the xmesh routing protocol in wireless sensor networks," *Circuits and Systems, 2006. MWSCAS '06. 49th IEEE International Midwest Symposium on*, vol. 2, pp. 113–117, Aug. 2006.
[8] U. Malesci and S. Madden, "A measurement-based analysis of the interaction between network layers in tinyos," in *EWSN*, 2006.
[9] P. Buonadonna, D. Gay, J. Hellerstein, W. Hong, and S. Madden, "Task: sensor network in a box," *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pp. 133–144, Jan.-2 Feb. 2005.
[10] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2003, pp. 14–27.
[11] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2003, pp. 134–146.
[12] M. D. Yarvis, W. S. Conner, L. Krishnamurthy, and A. Mainwaring, "Intel labs."
[13] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (blowfish)," pp. 191–204, 1994.
[14] *X-Board Reference Manual*, Digilent, January 2007.
[15] *CoolRunner-II CPLD Family*, Xilinx, March 2007.
[16] *MPR-MIB Users Manual*, Crossbow, June 2007, revision A.
[17] *MTS/ MDA Sensor Board Users Manual*, Crossbow, June 2007, revision A.