Reconfigurable Hardware Architecture for Saving Power Consumption on a Sensor Node

Satoshi Tanaka*, Naotaka Fujita*, Yutaka Yanagisawa†, Tsutomu Terada* and Masahiko Tsukamoto*
*Graduate School of Engineering
Kobe University, 1-1 Rokkodaicho, Nadaku, Kobe 657-8501, JAPAN
Email: {tanaka, nfujita}@stu.kobe-u.ac.jp, tsutomu@eedept.kobe-u.ac.jp, tuka@kobe-u.ac.jp
†NTT Communication Science Laboratories
NTT Corporation, 2-4 Hikaridai, Seikacho, Sorakugun, Kyoto 619-0237, JAPAN
Email: yutaka@cslab.kecl.ntt.co.jp

***Abstract*—We propose the use of a reconfigurable hardware architecture to reduce the power consumption of small sensor node that has various sensors and wireless communication facilities, that were the result of an adaptive function specialization mechanism. Traditional sensor nodes must have had a powerful and multi functional Micro-Controller Unit (MCU) to satisfy the requirements for processing any kinds of application. However, most of these systems only use a part of the functions provided by an MCU. In other words, such a unit often consumes a great dead of power for unused circuits. To avoid this situation, we propose the use of a reconfigurable architecture based on a Field Programmable Gate Array (FPGA) instead of an MCU because this array dynamically changes the circuit to the optimal one that is just used for the calculation required by an application. Moreover, we implemented a prototype system to do a preliminary evaluation of our proposed mechanism. In this evaluation, we show the performance of our proposed reconfigurable architecture by comparison with traditional architecture that uses processing time and power consumption. The experimental result shows that our proposed mechanism reduces enough power of its sensor nodes to prolong the lifetime of nodes without decreasing the processing time.**

I. Introduction

Recently, many wireless sensor nodes have been proposed. Among them are MOTE [1] and Smart-Its [2]. Each node uses various types of sensors such as those for acceleration, geomagnetic, and temperature. Further, powerful middleware are provided for developers, such as an operating and a, database system, and a programming environment, which are useful in developing a variety of ubiquitous applications.

One of the most significant issues in developing sensor nodes to save the power consumption without decreasing functionalities. This is because its lifetime has a strong affect on the reliability of its service. Downsizing a device is also an important issue in designing hardware, but there has been a trade-off between downsizing and power saving. To do both at the same time, the hardware has needed to be optimized to a specific application.

In order to optimize hardware in existing architecture, however, a developer must carefully design it, and this is often a costly process. To reduce the cost of design, we propose the use of a new architecture.

Our basic idea is based on a *reconfigurable* hardware architecture with a mechanism to adapt functions depending on the requirement of an application. We use an FPGA-based hardware instead of MCU, where FPGA is used for signal processing. Our FPGA-based processor can dynamically change the circuit to process functions required by an application. For example, when an application requires the spectrum analysis of sensor data, the system reconfigures the circuit to process the Discrete Fourier Transform (DFT). In another case, when an application requires only a simple analysis, it dynamically configures itself to process a Finite Impulse Response filter (FIR), which consumes less power than the DFT does. To develop such a sensor node, a developer using a traditional MCU-based device must make a sensor board that is independent from other devices; however, a developer using our FPGA-based device can dynamically change the circuits on the sensor node with software and thus not need to design hardware for each devices. Thus, we consider that using our method can reduce the cost of developing various types of sensor nodes.

We show a prototype device as an implementation with an IGLOO FPGA device produced by Actel Corporation. Furthermore, we show the experiment to evaluate our architecture by comparing the prototype of our implementation with a traditional MCU-based device. In this experiment, we used a program and Fast Fourier Transform circuit (FFT) for testing a function processed on the sensor nodes. In the experiment, we measured power consumption (mW) and processing time (msec) to compare the performance of each method. The results show that our architecture has an advantage in most cases.

The rest of this paper is organized as follows. First, we mention related works in Section 2. We then explain our approach and the core idea of the dynamic reconfigurable hardware architecture with the function specialization mechanism in Section 3. A prototype system and an implementation based on our architecture is described in Section 4 and 5.

The evaluation results in our experiment are shown in Section 6. Finally, we conclude the paper in Section 7.

II. Related Works

In this section, we mention related works on reconfigurable hardware architecture for developing small sensor devices.

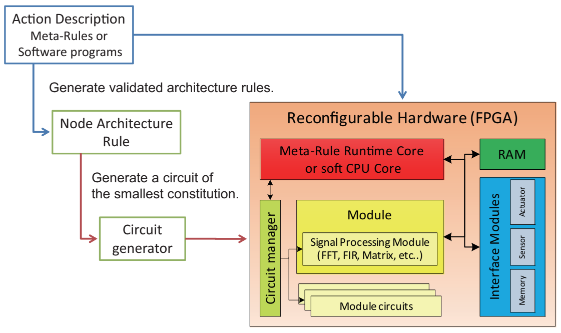
Action Description
Meta-Rules or
Software programs
Generate validated architecture rules.
Node Architecture Rule
Generate a circuit of the smallest constitution.
Circuit generator
Reconfigurable Hardware (FPGA)
Meta-Rule Runtime Core or soft CPU Core
RAM
Circuit manager
Module
Signal Processing Module (FFT, FIR, Matrix, etc..)
Module circuits
Interface Modules
Actuator
Sensor
Memory

Fig. 1. Goal of Hardware and Software Generator

Bellis, et al. proposed a function specialization that uses an FPGA device on a sensor node at a module to control the RF device for efficient routing on sensor network [3]. In these studies, using an FPGA enables us to downsize and process at a high-speed at the same time with an FPGA-based function specialization mechanism.

Most previous FPGA-based architecture brings us both the reduction of the device size and high-speed signal processing on a small sensor node. Moreover, in these architectures, the circuits cannot be dynamically changed to meet the requirements of applications because previous works focus on the case that a sensor device is only used in a specified application. In other words, a sensor node does not have to change functions dynamically because a developer of the application system fixes the required functions before developing the sensor node. Moreover a ubiquitous computing environment, many applications often require different functions to each sensor nodes to provide intelligent and convenient services to users. Because in this case the sensor node must calculate sensor data in a different way; the sensor needs to be dynamically reconfigured so it meets each requirement. Most of the previous sensor nodes do not support such dynamic reconfigurable mechanisms.

On the other hand, there are some reconfigurable devices [4] [5] [6] [7], which are aimed at specializing the function on the basis of both the requirements of the application systems and the place where a sensor node is put. These approaches are similar to ours. While works focused on quite primitive calculation operations, such as summation, and division, we implemented more powerful calculation functions such as FFT and FIR.

Thus, our contribution in this paper is summarized by the following two points. First, we show a dynamic reconfigurable hardware architecture with functions used in practical signal processing. Second, we show our implementation of a prototype system using an FPGA device, and further, we show the advantage of our mechanism over traditional mechanisms with an evaluation of the performance evaluation of our architecture.

III. Architecture

In this section, we describe the architecture of our system.

First, we describe the overall goals of our approach. The target of our approach is to reduce both the processing time and the power consumption of sensor nodes at the same time. For this purpose, we introduced a mechanism to configure the hardware dynamically to minimize the power consumption by specializing functions. To reconfigure the hardware, we used an FPGA instead of the MCU, which is a traditional device to process data on a sensor node. Our mechanism dynamically replaces circuits on the FPGA the set of functions required by application systems. That is, the FPGA only has just the minimum set of necessary circuits. By using this mechanism, a sensor node provides suitable a function set required by applications with the minimum power consumption without increasing the processing time.

In the rest of this section, we describe how our goal can be achieved with our proposing mechanism. Overview of our mechanism is shown in Fig. 1. The key device of our mechanism is the FPGA that can dynamically change circuits. The controller has also a signficant function, switching on the circuits on the FPGA. Our full implementation will use just an FPGA device as the controller but, to simplify the implementation, our prototype an MCU as the controller. The controller chooses a circuit set required by applications systems on the basis of previously described rules. A developer of sensor nodes describes available functions using the FPGA for application developers. When an application system needs to change the circuit sets, the controller dynamically replaces a set of circuits on the FPGA.

1) Hardware Architecture: Some of the traditional sensor nodes have a function that can change the software on altering demands of application [8]. However, the power consumption of the MCU changes little by changing the software in the main. In fact, although the traditional method cannot be used to reduce the power consumption but functions can be specialized it. It is necessary to replace circuits to reduce the power consumption. We can hold the power consumption to a minimum along the sensor nodes have that circuits to provide the function for required calculation. However, the hardware has a fixed architecture such an MCU, and it is impossible that the MCU cannot replace the circuits dynamically. We used the a that can reconfigure circuits as described above. Both the FPGA and the mechanism of dynamically replaceing the circuit replacement can process faster and lower power consumption than a versatile CPU with dynamical replacement of software does. For these reasons, we designed sensor nodes with FPGA.

2) Controller: The controller, which is a significant module, has significant three functions: first, generating a set of circuits required by application systems using rules described by the circuit developers; second, ordering the replacement of the circuit set to the FPGA-based processor; and third, transferring data between an application system and the processor. Developers provide a set of circuits as a module and, moreover, describes the function of a module as a rule. The module includes the set of circuits, which provides a function to process simple summation, average, DFT, FIR, and so on.

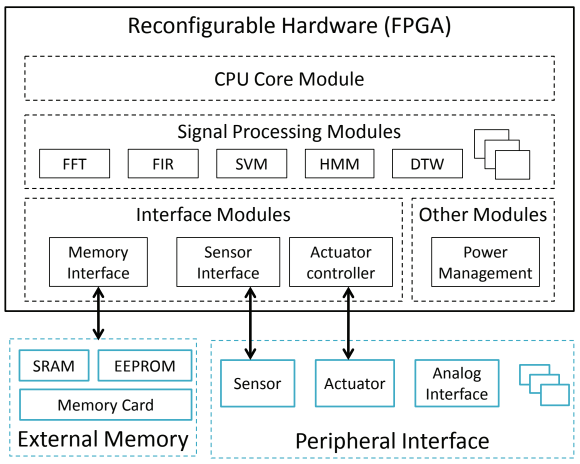

Fig. 2. Complete Design of Proposing Architecture

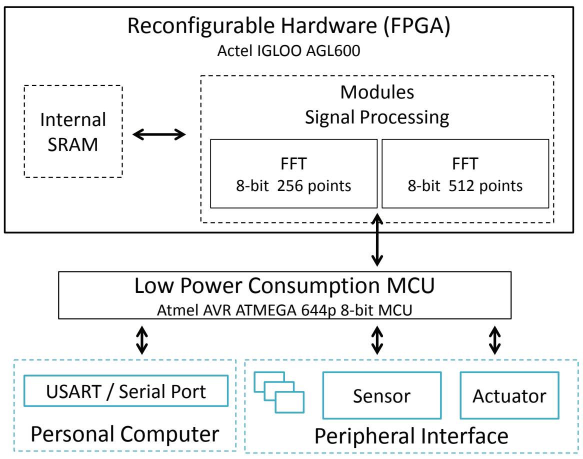

Fig. 3. Partial Design for Prototype System

Each module has the format both of input data and output data. The controller converts the format of data received from the application into a suitable format used in the module. Similarly, the controller returns the output data from the module with the format required by the application system. The mechanism allows many developers to make modules independently from each other, and they can also improve the module independently. We can add new functions to the sensor node by developing a module to provide the new function.

3) Software Layer: The reconfigurable hardware can calculate sensor data with dynamic changeable logical circuits. However, there is a trade-off when we design software on MCU or constitute a logical circuit because of scale of logical circuit have limit. The FPGA cannot process any tasks. It also does not well on sequential control. Note that both can be easily processed on an MCU. Therefore, we propose using a way to process complex calculations at a FPGA and others on software on an MCU.

The complete version of the proposing system is shown in Fig. 2. In this version, both the circuits on the FPGA and the software on the MCU are automatically built using both a library and rules written by developers. This library includes a logical circuit modules for the FPGA and software modules for the MCU, both of which have the same function. The controller automatically decides on the basis of the rule which modules must be used in the process at runtime.

IV. Prototype System

In the previous section, we showed the complete version of our architecture which achieves our goal, but the essential idea of our proposed architecture is made by using the reconfigurable hardware architecture. Therefore, we also have a simple version for the current implementation in order to evaluate the power consumed when reconfigurable hardware is used. Here, we explain the prototype system used in our experiment.

We extracted essential two mechanisms from our proposed architecture; that is the FPGA-based reconfigurable node that can dynamically turn on/off the power provided to a circuit on the FPGA. In other words, this device cannot dynamically change the circuit but using the device can limit it, and doing twis consumes the power. This prototype is quite simple as it has only the core mechanism of our proposed architecture. In our experiment, we measure both the power consumption and processing time with this prototype. If we show our prototype reduces the power consumption without decreasing the processing time, we will say that our complete version also has the capability to reduce the power consumption because each system has the same mechanism used to turn on/off the circuit on a sensor node. To compare our mechanism with the traditional fixed hardware architecture, we also implemented an MCU-based device. An overview of the prototype is shown in Fig. 3.

Here, we explain the design of our prototype using an FPGA and an MCU.

The purpose of implementing a prototype is to compare performance of our proposed architecture with the traditional method in a real situation. For comparing austerely, we combined the greater part of our architecture with the proposed FPGA system and the MCU based one. In particular, we used the independent module as a controller in the MCU. The module has the function to make both composed a simple MCU processor and composed am MCU with a FPGA processor to run a task. The module can also be used to measure time. We got the processing time and the pure power consumption in processing by combined driving and time measuring.

The prototype does not make use of sensor devices. We used random numbers or fixed number as calculating data as a substitute for sensor data.

Our prototype system has the following two features. First, only FFT is implemented as a logical circuit. Second, it supports a 256 and 512 byte result of FFT by writing code to the MCU in advance.

V. Implementation

Here, we show our implementation of the prototype system with reconfigurable hardware based on the architecture mentioned in the previous section.

A. Hardware

Our implemented sensor node based on reconfigurable hardware architecture with a low power microcontroller is shown in Fig. 4. Our prototype sensor board uses the "M1AGL-DEV-KIT" evaluation board that has the Actel IGLOO M1AGL600V2 FPGA chip, one of the lowest power flash-based FPGA chips. To control this processor, we also used the Atmel AVR ATmega644p as an MCU that is a low power microcontroller.

Our implemented board works on 3.3 V batteries. The board supplies 1.2 V on both the core and the I/O, and also supplies 3.3 V to the FPGA and the MCU. This implementation has a communication problem between the FPGA and the MCU because the processing time of FPGA is much shorter than that of the MCU. To avoid this, we set the clock of the FPGA to its 1/4 (4 MHz). The FPGA has two 8-bit parallel communication ports which are connected to the MCU. This is because parallel communication is one of the fastest methods for communication between chips.

B. Software

We implemented the software of logical circuit modules for signal processing in the FPGA as well as a software program for the MCU.

We used the FFT function as described in the previous section. The logical circuit module of the FFT is based on the CoreFFT IP that provided by the Actel corporation. The CoreFFT module can process a value series both of 256 points and 512 points with an 8-bit data width. Moreover, we added the circuit module to the CoreFFT for to save power. This additional module supports the function of an external way to control the FPGA for changing two modes: *active* and *sleep*. We developed these logical circuit modules using the Actel Libero8.3 with Verilog-HDL on WindowsXP.

The MCU processes the following three: First, generating sample data for calculating the FPGA. Second, sending data to the FPGA. Third, measuring the time needed to calculate the FPGA. Sample data are calculated on the FPGA instead of using real sensor data because it is difficult to maintain experimental environment. The data is 256 points and 512 points with 8-bit data width. The data generator generates sample data in two way: by using random numbers and by using a fixed value given by a user. We implemented a program code to communicate data between the FPGA and the MCU. Moreover, we also implemented a program code to communicate data between the MCU and the PC in order to send the measurement of the calculation time. The MCU rejects any requirements to communicate data from FPGA while the FPGA is working, and the FPGA sends the data to the MCU because it may be intercepted by I/O interruption signals in processing. We implemented a program to use a 16-bit hardware timer to measure the time. The measurement error of the processing time is within 0.01 msec. This was because, we allowed a 0.01 msec error that increases the capability of the intercept by reducing the measurement error for timer overflow. We develop this MCU program using C-Language and gcc/AVR-Libc1.6.2 on WindowsXP.

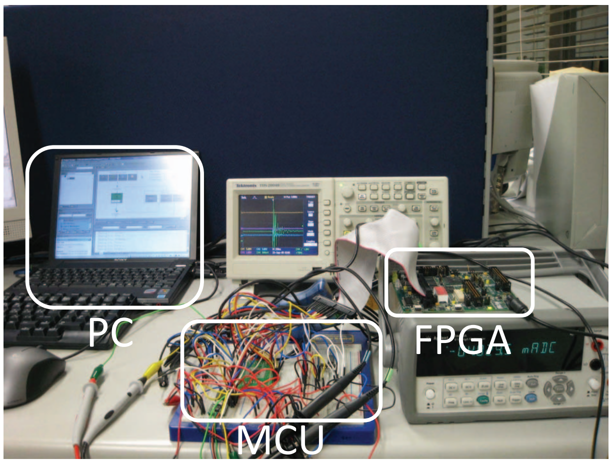

Fig. 4. Implementated Test Board

VI. Evaluation

We evaluated the power consumption and the computing time of reconfigurable hardware for analyzing sensor data. We show experimental results of experiments to reduce power consumption and specialize function.

We compare our implemented system with the MCU based method. In particular, we evaluated power consumption and processing time for processing the FFT in these systems. The evaluation program on the MCU-based method unit uses a program published by the third edition of "Numerical Recipes" [9].

A. Processing time

First, we show the result on processing time. We wrote a code for generating *dummy* sensor data of 8-bit, 256-points in the MCU-based method unit, then measured the time needed to acquire the calculated result. We compared the processing time of our reconfigurable unit and that of the MCU-based method unit. The reconfigurable unit transforms sensor data, and processes the FFT in FPGA. To reduce the possiblity of errors occurring in the measurement time as much as possible, the time is measured it by a hardware counter in MCU and we used the mean duration of 100 times calculation.

The result is shown in Table I. As a result, the reconfigurable hardware can calculate 97.06% and 97.29% faster. In addition, we can expect faster processing because we used 1 / 4 clock speed of the FPGA capability to make the communication between the MCU and the FPGA smooth. Furthermore, by solving the communication problem with the MCU, we will make it much faster.

TABLE I
EVALUATION OF FFT CALCULATING TIME

	8bit 256 points	8bit 512 points
Traditional Hardware	49.33 [ms]	106.46 [ms]
Reconfigurable Hardware	1.45 [ms]	2.89 [ms]

TABLE II
EVALUATION OF POWER CONSUMPTION

	Active	Sleep
Measured Power Consumption		
Traditional Hardware	35.97 [mW]	13.27 [mW]
Reconfigurable Hardware	42.82 [mW]	18.28 [mW]
Theoretical Power Consumption		
Traditional Hardware	25.08 [mW]	0.99 [mW]
Reconfigurable Hardware	28.00 [mW]	1.43 [mW]

TABLE III
EVALUATION OF FFT CALCURATION TOTAL POWER CONSUMPTION

	8bit 256 points	8bit 512 points
Measured Power Consumption		
Traditional Hardware	176.50 [mW·s]	382.94 [mW·s]
Reconfigurable Hardware	6.21 [mW·s]	12.37 [mW·s]
Reduction rate	96.48 [%]	96.77 [%]
Theoretical Power Consumption		
Traditional Hardware	123.72 [mW·s]	267.00 [mW·s]
Reconfigurable Hardware	4.06 [mW·s]	8.09 [mW·s]
Reduction rate	96.72 [%]	96.97 [%]

B. Evaluate of power consumption

Next, we show our experiment to measure the power consumption. In this experiment, we compared the amount of power consumption that depends on sensor data analysis in the active and the sleep mode of MCU and FPGA. We examined the power consumption by measuring the power supply to the MCU and the FPGA. We measured in current and supply voltage using a highly accurate 34410A multimeter made by Agilent Technologies. Power consumption is calculated by multiplying by current and voltage.

1) Power Consumption on Each Mode: We measured the power consumption in the active and the sleep mode of the MCU and the FPGA when calculating and sleeping in traditional hardware and reconfigurable hardware in FPGA at the calculating and sleeping. We also show in Table II the power consumption specifications provided by MCU and FPGA venders, as additional information.

The experiment results show that the power consumption of our proposed architecture based on the FPGA is higher than that in the traditional architecture using only the MCU. The measured power consumption is higher than the theory. We consider this is because of the charge of the test board.

2) Power Consumption for Calculating: We demanded the power consumption by multiplying the processing time that was shown in the evaluation of the processing time by power consumption of processing time. We show the results of the experiment on power consumption in Table III. Therefore, we catch on the method of the proposed system that could reduce the power consumption to 96.72 % and to 96.97 %.

As a result, we were able to reduce the power consumption for calculations.

VII. CONCLUSION

We have proposed the use of reconfigurable hardware architecture for a small sensor node. We evaluated our proposed mechanism with our implemented prototype system. In our experiment, the implemented system can reduce 96.72% and 96.97% power consumption than traditional hardware architecture. Moreover our mechanism also improves the processing time by 97.06% and 97.29% . We believe that the result of our experiment indicates a high likelihood of both saving power consumption and specializing functions. Thus, we consider that fully implementing our mechanism will lead to a smaller sensor node with a higher lead of processing power than existing sensor ones.

The current implementation does not have wireless communication functionality. However, we have another implementation of a small wireless device [10]. In general, existing sensor nodes, consume more power consumption for transmission wireless devices is much more than those used for calculation processing [11]. This is because, in most traditional sensor node architectures, all the necessary sensor data must sent to a server. One way to reduce the amount of data sent to the server from a node is to compress and analyze them with signal processing functions on the node. Our method possibly enables nodes to process such calculation with both low power consumption and in a short processing time. We are now extending the scope of our architecture to include the RF-part and to thus reduce the power consumption of the sensor node even more.

In the future, we will implement several other calculation circuits such as the Support Vector Machine (SVM), the Hidden Markov Model (HMM), and the Dynamic Time Warping (DTW), and thus make analyzing sensor data more powerful. Moreover, we will archive other additional experiments on our system in realistic environments.

ACKNOWLEDGMENTS

This research was supported in part by a Grant-in-Aid for Scientific Research (A)(20240007, 20240009) of the Japanese Ministry of Education, Culture, Sports, Science and Technology.

REFERENCES

[1] J. M. Kahn and R. H. Katz and K. S. J. Pister, “Next century challenges: mobile networking for “Smart Dust””, *Proc. of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 271-278, USA, 1999.

[2] Hans W. Gellersen and Albercht Schmidt and Michael Beigl, Kafil M. Razeeb, Cian O’Mathuna, “Multi-sensor context-awareness in mobile devices and smart artifacts”, *Mob. Netw. Appl.*, Vol. 7, pp. 341-351, 2002.

[3] Stephen J. Bellis, Kieran Delaney, Brendan O’Flynn, John Barton, Kafil M. Razeeb, Cian O’Mathuna, “Development of field programmable modular wireless sensor network nodes for ambient systems”, *Computer Communications*, Vol. 28, pp. 1531-1544, 2005.

[4] Ali El Kateeb, L. Azzawi, “Hardware Reconfiguration Capability for Third Generation Sensor Nodes: Design and Challenges”, *Proc. of Workshops conjunction with 22nd International Conference on Advanced Information Networking and Applications*, Japan, pp. 675-680, 2008.

[5] Ali El Kateeb, Aiyappa Ramesh, L. Azzawi, “Wireless Sensor Nodes Processor Architecture and Design”, *Proc. of Workshops conjunction with 22nd International Conference on Advanced Information Networking and Applications*, pp. 892-897, Japan, 2008.

[6] H. Hinkelmann, P. Zipf, M. Glesner, “Design Concepts for a Dynamically ReconfigurableWireless Sensor Node”, *Proc. of First NASA/ESA Conference on Adaptive Hardware and Systems*, pp. 436-441, USA, 2006.

[7] H. Hinkelmann, P. Zipf, M. Glesner, “A Domain-Specific Dynamically Reconfigurable Hardware Platform for Wireless Sensor Networks”, *Proc. of International Conference on Field-Programmable Technology 2007*, pp. 313-316, Japan, 2007.

[8] Ting Liu, Margaret Martonosi, “Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems”, *Proc. of the 9th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 107-118 USA, 2003.

[9] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *“Numerical Recipes 3rd Edition: The Art of Scientific Computing”*, 3-rd edition, Cambridge University Press, pp. 608-639, USA, 2007.

[10] Kenji Kodama, Naotaka Fujita, Yutaka Yanagisawa, Masahiko Tsukamoto, Tomoki Yoshihisa, “A Rule Engine to Process Acceleration Data on Small Sensor Nodes”, *Proc. of the 5th International Conference on Pervasive Services, Poster Session*, Italy, 2008.

[11] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, M. Welsh, “Simulating the Power Consumption of Large-Scale Sensor Network Applications”, *Proc. of the 2nd ACM Conference on Embedded Networked Sensor Systems 2004*, pp. 188-200, USA, 2004.