

# Energy Efficient Software-Based Self-Test for Wireless Sensor Network Nodes

Rong Zhang, Zeljko Zilic  
McGill University, Dept. ECE  
Montreal, QC, Canada H3A 2A7  
{zhangr, zeljko}@macs.ece.mcgill.ca

Katarzyna Radecka  
Concordia University, Dept ECE  
Montreal, QC, Canada H3G 1M8  
kasiar@ece.concordia.ca

## Abstract

*We consider self-testing of complete wireless nodes in the field through a low-energy software-based self-test (SBST) method. Energy consumption is optimized both for individual components such as a CPU, embedded memories, and an RF module, as well as at the system level, considering the interplay between module tests. We first derive a scheme for software-based tests with the least amount of cycles and with operands of least Hamming distance and weight. Time interleaving of module tests at the system level further reduces the overall test energy consumption.*

## 1. Introduction

In recent years, wireless sensor networks (WSN) have become available for use in various industrial control, environmental and military applications. A WSN is a system composed of small, wireless nodes that cooperate on a common distributed application under strict energy, cost, noise and maintenance constraints. The ability to build reliable WSNs is essential to their acceptance in many applications. It is shown (see e.g.[1]) that the in-field self-test of individual nodes throughout their lifetime is needed to achieve sufficient WSN reliability and availability in the face of the overwhelming system-level constraints.

In-field test can be performed using a built-in self-test (BIST) infrastructure incorporated into wireless nodes. Hardware BIST, however, is often not possible. Dedicated test circuitry incurs a performance, area, and energy overhead [2], and in general is not preferable for low-cost WSN nodes, which are often built with commercial off-the-shelf (COTS) components. Therefore a software-based self-test (SBST) is considered as an effective alternative that can provide high quality at-speed testing with miniscule area and performance overhead. Such an SBST solution utilizes existing microcontroller instruction set to perform a self-test of all digital and mixed-signal components in a WSN node. In addition to the test quality, energy consumption is a major concern in testing WSN nodes.

Energy optimization of hardware, and even more so for software is a complex problem that is further hampered by lack of accurate power models, especially for purchased IP cores and COTS processors. Furthermore, energy consumption is a global phenomenon that depends on the precise interplay of all components in the system, including modern wireless protocols that dynamically adjust the transmission energy. Calculating the energy consumed during wireless node SBST based on an accurate model is then beyond our reach. Instead, for the SBST development without a comprehensive power model, we can rely on measuring the exact consumption profile. For this, we require a complete WSN node [1] outfitted with the accurate current sensing circuitry.

The SBST studied here considers the complete WSN node, including CPU, memories and an RF module, as its major components. For the processor core, we design SBST by exploiting its instruction set functionality and some knowledge of its structure (e.g., major buses). Instruction-level techniques select addressing modes, operands with minimal Hamming distance and weight and combine instructions through dynamic programming. The increasingly essential FLASH memory is tested by a March-type algorithm implemented in energy-efficient test software. An RF module characterization test is further devised. It uses our network test architecture to achieve the cooperation of several nodes in finding accurately whether the module meets the major parts of the RF specification. The test time and energy consumption are further reduced by the interleaving of module test codes, as a special case of test scheduling focused around prevalent FLASH test latencies. All the major design steps are based on the gathered energy profiling information, rather than simplistic models. The proposed techniques are flexible and cost-effective for a variety of networked embedded systems.

Section 3 presents the instruction-level SBST energy reduction method. The embedded FLASH tests, and time interleaving used for energy and time reduction are presented in Section 4. RF module validation

testing is outlined in Section 5. Section 6 demonstrates the efficiency of our methods by current measurements performed on a real wireless node.

## 2. Overview of the test methodology

A generic wireless node has at least an embedded microcontroller and an RF module, Figure 1. A microcontroller with its embedded memory performs data processing and control tasks. It can also be used to implement node self-testing, to interpret and communicate test results. An RF module combines the operations of an RF transceiver, balun circuitry and an antenna for seamless wireless transmission within a given specification. Modern RF modules support several low-power modes and provide encryption and media access control protocol support.

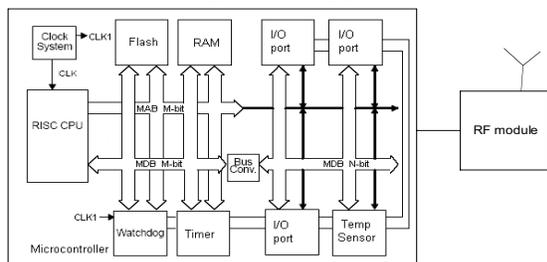


Figure 1 Generic node architecture

High system *availability* requires quick fault detection and its repair, hence nodes should test themselves [1], as a part of the broader in-field network test architecture. In such a scheme, a dedicated *Task Manager Node* (TMN) remotely activates and then coordinates a self-testing session of a *Node Under Test* (NUT) through a wireless channel.

Since wireless nodes are currently mostly made of IP cores and COTS, the possibility of adding self-test hardware is limited, and certain to induce additional cost. Hence, SBST is an appealing choice.

## 3. Energy reduced SBST of CPU

In early 1980's, Thatte, Abraham and Brahma [3],[4] proposed a *s-graph* model at a Register Transfer Level (RTL) to express a microprocessor self-test objective, and used functional fault models for instruction-level test generation. Further graph-based functional testing methods were proposed in [5], [6].

Modern SBST of microprocessors falls into two categories. The first group includes functional SBST approaches [7], [8] employed at a high level of abstraction. The second category represents a structural SBST [9] [10], [11], which requires a fault-driven test development. Fault coverage of a functional fault model is usually low, even with numerous manual interventions, while the test set can be lengthy [7], [8].

The structural SBST methodology in [10] is based on deterministic test generation targeting structural faults of individual processor components. The high coverage (>95%) SBST from [11] explores a divide-and-conquer approach targeting individual components for stuck-at faults and defines different test priorities for processor components. It combines desirable characteristics of functional testing (like test development at high level using the processor instruction set) with good use of RTL information.

### 3.1 Instruction level energy reduction methods

The testing of a WSN node processor core presented here augments the framework in [11] with a classical dynamic programming approach to code optimization, as in [12]. The optimality criterion is expressed here in terms of software energy consumption, rather than the program length, with instruction energy profile obtained from measurements.

The realistic model of the software energy consumption in a modern processor is hard to obtain, even when its full netlist is available. The energy consumed by a test code is equal to the sum of energies used by all the instructions executed. Furthermore, there is a dependency on the values of instruction parameters (operands, registers, addresses, etc.), referred to as *energy sensitive factors* [13], [14]. In a very simplistic model, energy consumption is proportional to the Hamming distance between consecutive instructions, and the weight in the current values of the energy sensitive factors.

In absence of good models, we ultimately rely on measuring the current during a test. By repeatedly executing short instruction sequences we obtain energy consumption profiles accurate to the instruction level, including accounting for energy sensitive factors and addressing modes. Obtained energy profiles let us employ instruction-level energy reduction based on the exact knowledge of the per-instruction energy consumption. Methods for energy reduction include selecting and combining instructions with the fewest CPU cycles, and selecting operands with least Hamming distance and weight.

**Instruction Selection and Combination:** Starting with SBST code such as in [11], our procedure performs a series of localized instruction replacements towards obtaining energy-optimized test code. To minimize SBST energy, an *instruction selection* step chooses instructions requiring the least amount of CPU cycles, while preserving the test coverage.

**Example 1:** Let *March X* algorithm test a register file. The operation set is  $O_{Reg} = \{\text{Write 0, Read 0, Write 1, Read 1}\}$ . WriteX can be implemented by instruction set  $I(\text{Reg, WriteX}) = \{\text{mov X, Rn}\}$ . We apply instruction selection to  $O_{Reg} = \{\text{Write 0}\}$ .

```

I1 mov 0, R1      I2 xor R1, R1
   mov 0, R2      mov R1, R2
   ...           ...
   mov 0, Rn      mov Rn-1, Rn

```

Software energy consumption in I2 is reduced because *mov* instruction with immediate addressing mode (I1) takes two CPU cycles, while *mov* instruction with register addressing mode (I2) takes one cycle. By this instruction selection, *n* CPU cycles are saved in I2 without compromising fault coverage.

Instruction combination is another instruction-level reduction method exploiting collateral coverage for other not-targeted components. We may use the same instruction sequence for different component operation testing such as ALU test and data bus test. The duplicated instruction sequences can be combined with the similar dynamic programming approach in [12]. This instruction combination will decrease the instruction sequence length without harming fault coverage of each component.

**Operand Selection:** The weights of the instruction operands and the Hamming distance between successive instructions are of major concern in energy reduction addressed by *operand selection*.

**Example 2:** Consider one of the operations for ALU testing  $O_{ALU} = \{\text{add with carry}\}$ . We compare the instruction sequences I3 and I4 obtained before and after operand selection, respectively. The operand 0x8000 in I4 has the lowest Hamming distance and weight among operands that test the given ALU fault.

```

I3 mov 0xFFFF, Rn  I4 mov 0x8000, Rn
   mov 0xFFFF, Rm  mov Rn, Rm
   add Rn, Rm       add Rn, Rm

```

Based on the above instruction-level energy reduction methods and SBST from [11], our SBST method is shown in Algorithm 1. Steps 1-3 identify CPU components tests, as well as instruction sequences that test components fully. Step 4 uses the information extracted in the previous steps; the components that appear in a CPU core are sorted by test priorities. Instructions selection and combination (Step 6) are followed by operand selection with least Hamming distance and weight (Step 7).

```

1. C = {C1, C2...CN}; // set of CPU components
2. Oc = {O1, O2...Om}; // set of ops for component C
3. l(C, O) = {l1, l2...lp}; // instructions causing C to perform O
4. Sort C in decreasing test priority
5. for (i=1, i<N+1, i++) // for untested component Ci
   for (j=1, j<M+1, j++){ // for operation Oj from set Oci ;
6. Find a least energy test sequence lk from l(Ci, Oj) by
   dynamic programming // inst. selection and combination;
7. Select lk operands of least Hamming distance and weight;
8. Apply lk to Ci and propagate the result to outputs.
   }}

```

Algorithm 1: Low-Energy SBST Generation

## 4. Efficient SBST of memory

The trend of incorporating growing amounts of FLASH will make the FLASH test predominant in a wireless node SBST. FLASH is a non-volatile memory that allows erasing the memory data in blocks. The conventional RAM testing methods are not applicable because FLASH cannot perform random access erase. In [15], an efficient March-type algorithm (March FT) was proposed for conventional and memory disturb faults [15]. We use the March FT algorithm as a starting point, to which we first apply the instruction selection and combination methods from Section 3.1. Next, we consider bus switching activity reduction.

### 4.1 Efficiency of SBC addressing

A method that minimizes the Hamming distance between the consecutive addresses during March-type tests was introduced in [16]. Authors replace the usual *binary* (consecutive) address sequence with the Single Bit Convert (SBC) addressing by which the address bus transitions are reduced by 50%. Total energy reduction claimed was between 18% and 77% for different sizes of standalone RAM memories.

In absence of detailed energy models, energy profiling is indispensable in devising energy-efficient memory SBST. We establish by measurements that the SBC method might actually *increase* energy consumption. Figure 2 compares measured currents for SBC and the binary addressing for on-chip FLASH testing on TI MSP430 processor. Although the SBC addressing draws less average current on the bus, the overall energy consumption is higher. For embedded memories (such as in MSP430 processor), the energy overhead (proportional to *time \* current*) in instructions needed to implement conversion from binary to SBC (e.g. *shift*, *xor* and *mov*) is more costly than the amount saved by SBC encoding.

We conclude that the energy reduction by switching activity minimization on memory bus requires energy profiling to infer the relation between the parts.



Figure 2 Comparison of Binary and SBC tests

### 4.2 Time interleaving FLASH and other tests

FLASH memory can perform only block or chip erase, with the erase operation being much slower than

read and write. Any erase cycle can be initiated from within FLASH memory or from RAM. When a FLASH segment erase operation is initiated from within FLASH memory, all timing is controlled by the FLASH controller, and the CPU is held while the erase cycle completes. After the erase cycle completes, the CPU resumes code execution.

During a FLASH erase cycle, CPU can be utilized, provided that we test other components with code executed in RAM, which is the premise of time interleaving, illustrated in Figure 3. The efficiency of time interleaving depends on the size of the FLASH, test code, timing and energy consumption relation between FLASH (erase/program/read) and other components. There is also a possible overhead in transferring the test code to RAM.

Figure 3 (a) shows a normal test routine sequence and the average power. By applying time interleaving, Figure 3 (b), tests are rescheduled by interleaving the FLASH erase cycle with other component tests, causing a reduction in overall test time and energy:

$$\text{Time reduction (\%)} = (T_{FE} + T_{Other} - T_{Interleave}) / T_{Total}$$

Where  $T_{Total} = T_{CPU} + T_{FE} + T_{FP} + T_{FR} + T_{Other}$ ,  $T_{CPU}$ ,  $T_{FE}$ ,  $T_{FP}$ ,  $T_{FR}$ ,  $T_{Other}$  and  $T_{Interleave}$  are in the Figure 3.

$$\text{Energy reduction (\%)} = (E_{Before} - E_{After}) / E_{Before}$$

Where  $E_{Before}$  and  $E_{After}$  are the total software energy consumed before and after the time interleaving.

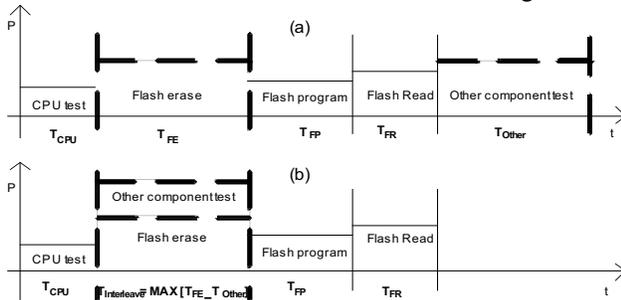


Figure 3 The concept of time interleaving

From above equations, the amount of energy and time reduction due to interleaving alone increases with the size of the FLASH and the proportional disparity between the FLASH and CPU speeds, both of which increase with advances in technology.

## 5. Characterization testing of RF module

Failures such as broken/dislodged antenna and RF circuit parameter drifting can prevent the RF module from meeting the specification. For the same reasons as in previous sections, we concentrate again on in-field test scenario, using our network test architecture.

Instead of loopback mode tests, we characterize the complete RF module (including antenna). The key specifications of IEEE 802.15.4 [18] are in Table 1.

Table 1 Key specifications in IEEE 802.15.4

Parameter	Test configuration/ (Specifications)
Transmitted Power	Nominal output power: 0dBm / (> -3dBm).
Receiver Sensitivity	The threshold input signal power yielding (<1%) PER / (<-85dBm)
Adjacent/Alternate channel rejection	Adjacent/Alternate channel interference level for <1% PER/ (>0 dB/30 dB)

The test methodology, presented in subsections to follow, relies on the Friis transmission equation [19]:

$$P_R (dBm) = P_T (dBm) + G_T (dB) + G_R (dB) - 32.44 - 20 \log f (MHz) - 20 \log d (km) \quad (1)$$

where  $P_R$ ,  $P_T$  are the receiving and transmitting signal power;  $G_T$ ,  $G_R$  are the antenna gain for transmitter and receiver;  $f$  is a working frequency and  $d$  is a distance.

### 5.1 Transmitted power and receiver sensitivity

Figure 4 shows the test setup for first two specifications. Received power for test packets that are continuously sent from NUT is read at TMN from the registers of the receiver IC [17]. The *transmitted power* is then calculated from the received power, frequency and a distance using Eqn. (1). Similarly, *receiver sensitivity* of NUT is obtained by sending test packets from TMN to NUT, and searching for the transmit level at which the 1% PER is observed. Finally, Eqn. (1) directly determines the receiver sensitivity as the received power at which PER is smaller than 1%.

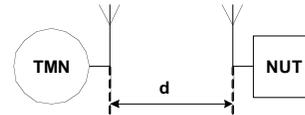


Figure 4 Transmit and receiver sensitivity test

In both cases, we ensure that the PER is < 1% by performing sweeps in test SW through the transmit power levels until reaching the 1% threshold.

A multi-channel physical layer specification requires good interference rejection between channels. The specification distinguishes *adjacent* and *alternate* channels. For instance, channel 13 and 14 as adjacent, while channels 11 and 15 are alternate.

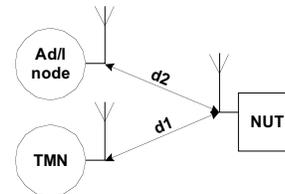


Figure 5 Testing adjacent channel rejection

The test setup for adjacent/alternate channel rejection uses three nodes, Figure 5. TMN is the transmitter, NUT is the receiver, and the third node is the interference source. The signal level from TMN is set to values required in [18]. By sweeping through the interference levels, for the PER crossing 1%, we then apply twice Friis equation. From frequencies, distances  $d_1$  and  $d_2$  (Figure 5), and the received power levels,

we calculate the emitted power levels. Channel rejection is the difference in two power levels in dB.

## 5.2 Energy considerations for RF testing

The instruction-level energy reduction method and time interleaving are applicable throughout the RF test. Additional optimization steps include the sweeping in the order of increasing power level in Section 5.1. In this way, the transmit energy level is increased only when needed; transmit energy increase step can also be adjusted dynamically, based on the power differential between steps and again by Eqn (1).

Modern wireless protocols incorporate several energy reduction techniques, including the use of beacon signals. Testing of WSN node is periodically activated by the beacon signal, and the NUT mostly enters the sleep mode between beacons. Figure 6 shows current measurement under the different operation modes of NUT node. Modes 1-3 send test packets at different transmission power levels. Sleep modes 4-5 are with and without network beacons.

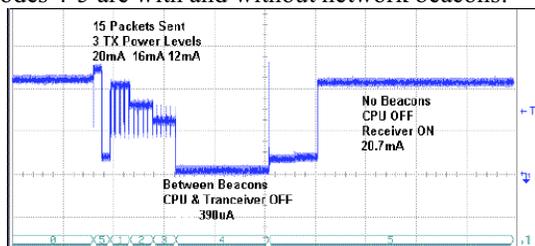


Figure 6 Current consumption profile

Considering least test software energy consumption, the cycling beacon RF communication scheme should be used and the TX power level should be set as low as possible with the required PER.

## 6. Experimental results

Our SBST and energy reduction methods were implemented, and results were measured for a baseline node in [1] (photo omitted for space reason). The node contains a TI MSP430 microcontroller, a CC2420 RF transceiver and a printed antenna integrated directly on a PCB, as detailed out in [20].

The current measurement setup added to the board is outlined in Figure 7. It measures the instantaneous current drawn by the processor. The voltage drop measured across a small resistor is amplified by the Burr-Brown INA145 programmable-gain amplifier, and the output voltage is recorded by the Agilent 54830D oscilloscope, capable of synchronizing recording with digital signals  $D_{0-3}$ , as with logic analyzers. These signals help identify different *modes* in test routines. We display these modes on the bottom of oscilloscope screen captures. Energy consumed by test routines is calculated by integrating the product of

instantaneous currents with node power supply voltage  $V_{dd}$ . Current integration is done in Agilent 54830D.

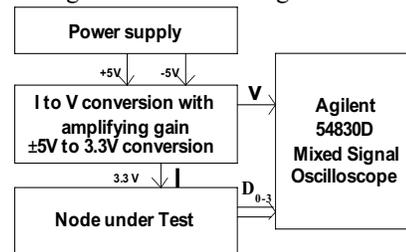


Figure 7 Current measurement block diagram

Energy consumption for MSP430 CPU testing method in Section 3.1 is recorded in Table 2, together with the baseline SBST before applying energy reduction method in Algorithm 1. We observe a 21.2% energy reduction and a 20.1% time reduction by using all instruction selection and combination techniques. We also show the energy reduction achieved by using only the operand selection technique, to indicate the relative savings by different components.

Table 2 Energy consumption of CPU tests

	Original test	Operand Sel.	Instruction Sel.
Energy [J]	2.558E-6	2.43E-6	2.014E-6
CPU cycles	940	940	751

### 6.1 Current measurement for time interleaving

We implemented the proposed time interleaving, Section 4.2, and present the results for two cases: time interleaving between FLASH and RAM testing, and time interleaving between FLASH and RF module testing. Table 3 gives the description of different test routines in these two cases.

Table 3 Test routines in Figure 8 and 9

Mode	Description
5	FLASH erase (one block erase-512bytes)
6	FLASH R1, W0, R0 (512bytes)
7	FLASH R0 (512bytes)
8	RAM W0 (ten blocks -5kBytes)
9	RAM R0 (ten blocks - 5kBytes)
A	RF Initialization
B	RF packets sending (4 packets)

Figure 8 shows the current measurement result before and after time interleaving between FLASH erase and RAM testing within embedded memories in the MSP430 microcontroller (TI). Since one block (512 bytes) is the minimum unit for FLASH erases in the MSP430, we will take one block of FLASH testing as an example to show the efficiency of time interleaving. Modes 5-7 are the first three elements of the March FT algorithm for FLASH and modes 8-9 are the main parts of the March X algorithm for RAM.

Table 4 Energy for FLASH and RAM testing

	Before Interleave	After Interleave
Energy [J]	715.78E-6	619.57E-6
Time [ms]	91.24	72.3

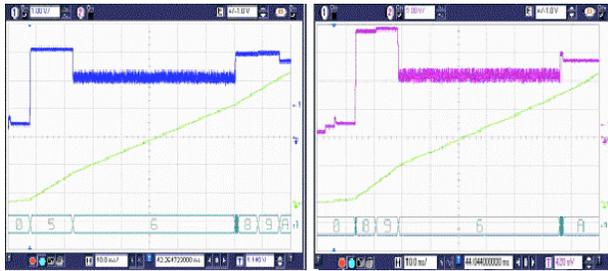


Figure 8 **FLASH/RAM test interleave**

Time interleaving can also be used between embedded FLASH and the RF module, two components at the system level. One FLASH block erase and 4 RF packets transmission take roughly the same amount of time, so we interleave them.

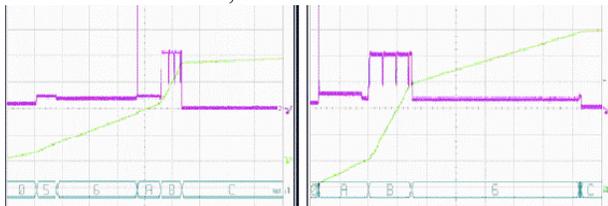


Figure 9 **FLASH/RF test interleave**

Table 5 **FLASH and RF test energy**

	Before Interleave	After Interleave
Energy [J]	2.236E-3	1.996E-3
Time [ms]	105.2	90.24

## 6.2 Network testing for RF module

The RF module consists of the RF transceiver, a balun circuit and a printed antenna, the latter two detailed out in [20]. A 2.4GHz IEEE 802.15.4/Zigbee RF transceiver [17] has 8 programmable transmission power levels and a built-in received signal strength indicator (RSSI). The gains of the printed antenna,  $G_T$  and  $G_R$  are both 8dB [20]. The results in Table 6 show that our RF module meets the specification [18].

Table 6 **RF module characterization**

Features	Specification	Test
Transmit power	0dB (MIN -3dBm)	-1dBm
Receiver sensitivity	Maximum -85dBm	-88dBm
Adjacent rejection	Minimum 0dB	7dB
Alternate rejection	Minimum 30dB	43dB

## 7 Conclusions

In this paper, we presented a comprehensive system-level low power SBST method for wireless nodes. The test scheme is primarily aimed at in-field testing, but can be applied to manufacturing test as well. For CPU SBST, instructions with fewest cycles and operands with least Hamming distance and weight are selected via dynamic programming approach. The CPU testing energy reduction of 21.2% is observed by current measurement on a prototype node. Time interleaving of the embedded FLASH tests is a major system-level technique used to reduce the energy consumption and

the test time. In addition to the above energy reduction techniques, RF module test can benefit from the transmit energy optimization and the use of low-power modes native to the modern protocols.

**Acknowledgements** go to J-S. Chenard for current sensing circuit and B. Mihajlovic for editing help.

## 8 References

- [1] M.W. Chiang, Z. Zilic, K. Radecka and J. Chenard, "Architectures of increased availability wireless sensor network nodes", *Proc. Intl. Test Conf.* 2004, pp. 1232-1241.
- [2] A. Krstic, W.C. Lai, K.T. Cheng, S. Dey, "Embedded software-based self-test for core-based designs", *IEEE Design & Test of Computers*, July-Aug. 2002, pp.18-27
- [3] S. Thatte and J. Abraham, "Test Generation for processors," *IEEE Trans. Computers*, 1980, pp. 429-441.
- [4] D. Rahme and J. Abraham. "Functional Testing of processors", *IEEE Trans. Computers*, 1984, pp. 475 – 485.
- [5] A. J. van de Goor and Th.J.W. Verhallen, "Functional testing of current microprocessors (applied to the Intel i860)", *Proc. Intl. Test Conference*, 1992, pp. 684-695.
- [6] B.S. Joshi and S.H. Hosseini, "Efficient algorithms for processor Test" *Proc. Reliability Symp.*, 1998, pp. 100-104.
- [7] J. Shen and J. Abraham, "Native Mode Functional Test Generation for Microprocessors with Applications to Self-Test and Design Validation", *Proc. ITC*, 1998, pp. 990-999
- [8] K.Batcher, C.Papachristou, "Instruction Randomization Self Test for Processor Cores," *Proc. VTS*, 1999, pp. 34-40.
- [9] L. Chen and S. Dey, "Software-Based Self-Testing Methodology for Processor Cores", *IEEE Trans. CAD*, March 2001, pp. 369-380.
- [10] N. Kranitis, A. Paschalis, D. Gizopoulos and Y. Zorian, "Effective software self-test methodology for processor cores", *Proc. DATE*, 2002, pp. 592 – 597.
- [11] N. Kranitis, A. Paschalis, D. Gizopoulos and G. Xenoulis, "Software-Based Self-Testing of Embedded Processors", *IEEE Trans. Computers*, 2005, pp. 461 – 475.
- [12] A.V. Aho and S.C. Johnson, "Optimal Code Generation for Expression Trees", *Journal of ACM*, 1976, pp 488-501.
- [13] S. Nikolaidis and T. Laopoulos, "Instruction-level power consumption estimation for low-power applications", *Proc. Workshop IDAACS*, Jul. 2001, pp.139 – 142.
- [14] N. Chang, K.H. Kim and H.G. Lee, "Cycle-accurate energy consumption measurement and analysis: Case study of ARM7TDMI", *Proc. ISPLED*, 2000, pp. 185-190.
- [15] J-C. Yeh, C-F. Wu, K-L. Cheng and Y-F. Chou "FLASH memory built-in self-test using March-type algorithms", *Proc. Workshop EDTA*, 2002, pp. 137-141.
- [16] H. Cheung and S.K. Gupta, "A BIST methodology for comprehensive testing of RAM with reduced heat dissipation", *Proc. Intl. Test Conference*, 1996, pp.386 – 395.
- [17] Chipcon SmartRF CC2420 preliminary datasheet 2004.
- [18] *IEEE std. 802.15.4/D18 – 2003: MAC And Physical Layer Specification for Low Rate Wireless Networks*.
- [19] W.L. Stutzman and G.A. Thiele. "Antenna Theory and Design", Wiley, second edition, 1997.
- [20] J. Chenard, C.Y. Chu, Z. Zilic and M. Popovic, "Design methodology for wireless nodes with printed antennas." *Proc. DAC*, 2004, pp. 291-296.