Energy Reduced Software-Based Self-Testing for Wireless Sensor Network Nodes

Rong Zhang

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

October 2005

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering.

Abstract

Reliable operation of networks consisting of many embedded wireless nodes operating under strict cost and energy constraint is a major requirement for their widespread acceptance. Towards this goal, we consider self-testing of complete wireless nodes in the field through a low-energy software-based self-test (SBST) method. The energy consumption of SBST 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 derive an SBST scheme that utilizes existing CPU instructions efficiently by taking the least amount of cycles and selecting operands with least Hamming distance and weight to minimize the overall energy consumption of the test code. Time interleaving of module tests at the system level is used to further reduce the overall test energy consumption. The efficacy of the proposed methods is evaluated experimentally, using current measurement circuitry integrated in a wireless sensor network node. Finally, we provide a significant amount of data from wireless network performance testing that can be efficiently utilized to build statistical models for protocol, algorithm and hardware design of wireless sensor network.

Résumé

Une opération fiable des réseaux de capteurs sans fil soumis à des conditions strictes de coût et d'énergie est un des facteurs déterminants pour une utilisation étendue de cette nouvelle technologie. Suivant cet objectif, nous considérons un autotest complet du noeud sans fil en utilisant un test intégré à faible consommation énergétique. Les économies d'énergie sont optimisées pour chaque composante telle l'unité centrale de traitement, les mémoires intégrées ainsi que le module de transmission radio. Nous dérivons un système utilisant les instructions de l'unité centrale de traitement de façon efficace en utilisant le moins de cycles possible. En sélectionnant les opérations afin que leur distance de Hamming soit minimisée, la consommation d'énergie globale du logiciel de test intégré est améliorée. Un entrelacement temporel de l'exécution des tests permet d'économiser encore davantage d'énergie. L'efficacité de la méthode proposée est évaluée de façon expérimentale en utilisant un circuit mesurant dynamiquement la consommation de courant d'un noeud de réseau de capteurs sans fil. Finalement, nous présentons une quantité appréciable de données receuillies à partir de tests de performance et pouvant être utilisées pour construire un modèle statistique, utile pour la validation de protocoles et la conception de matériel relié aux réseaux de capteurs.

Acknowledgment

I would like to take this opportunity to show my sincere appreciation to my supervisors: Dr. Zeljko Zilic and Dr. Katarzyna Radecka. It has been an extremely rewarding experience for me to do my master research under their seasoned guidance. I am thankful for their time and effort to instill in me the many sides of research.

I would like to extend my appreciation to my peers in MACS lab. I thank Jean-Samuel Chenard for his help in design of current measurement board and for all the fruitful discussions that sparked so many great ideas. I want to thank my colleagues Milos Prokic, Usman Khalid and Kahn-Li Lim for their help and suggestions on my master research. I also wish to thank Bojan Mihajlovic for his proofreading of my thesis.

I am grateful towards the ReSMiQ and towards my supervisors for their financial assistance during my master's studies.

During the years of my studies, I have always been enjoying the dedicated support from my parents. I am grateful for their sacrifice and endurance, which ensure that I can receive the finest education. I also deeply thank my husband, Bo Xu, for his support and encouragement throughout my master's studies.

Contents

List of Figures

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In recent years, Wireless Sensor Networks (WSNs) have become available for use in various industrial controls, environment monitoring and military applications. According to a statistical survey [1], over 53 percent of 200 industrial end-users and systems integrators are considering deploying a WSN in 2006. By comparison, a similar survey in January found 45 percent of respondents planning to deploy a WSN in 2005. Interest in WSNs specifically for industrial monitoring has also increased as 73 percent of respondents are researching wireless networks for use in these environments, compared to 64 percent of respondents surveyed in January, the study found. Reliability of wireless sensor networks were the main reason respondents were delaying deployment, according to 33 percent of those surveyed. 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

To achieve sufficient WSN reliability and availability, the periodic in-field test is needed to pinpoint and repair (or bypass) the failed WSN node that might be physically unreachable [2]. The environment in which WSNs operate can speed up failure mechanisms through, for example, cosmic radiation and extreme temperatures. Therefore, the periodic testing of WSN node should be controlled remotely. A testing session might result in processing a large volume of vectors that makes the completely remote test vector generation unrealistic. In addition to bandwidth limitations (most WSNs use low-bandwidth channels), it is not guaranteed that the sent vectors will reach the destination node, unless their reception is explicitly confirmed, which is prohibitively energy- and time-consuming. Therefore, the rational solution is that each WSN node has a build-in self-test architecture. Then, the communication with a tested WSN node happens only during the initialization of a test procedure and reporting of the outcome of test sessions. The on-board self-test capability of a WSN node is essential for the availability of WSN.

In-field test can be performed using a Built-In Self-Test (BIST) infrastructure incorporated into the wireless nodes. Hardware BIST uses embedded hardware test generators and test response analyzers to generate and apply test patterns on-chip at the speed of the circuit under test. BIST moves the testing task from external Automated Test Equipment (ATE) to internal hardware, thereby eliminating the need for an external tester and reducing the overall test cost. Hardware BIST [3, 4, 5, 6], however, faces many challenges. The most widely used logic BIST relies on the generation and application of pseudorandom test patterns. The fault coverage achieved by pseudorandom testing may be low for some random-pattern resistant circuits, such as microprocessor. The insertion of the BIST circuitry used for generating and applying pseudorandom patterns may result in a significant area and performance overhead. Furthermore, the low cost WSN node is often built with commercial off-the-shelf (COTS) components that make extra circuitry insertion is not feasible.

Recently, the use of low-cost Software-Based Self-Test (SBST) methodologies has been proposed as an effective alternative to hardware BIST, especially for core-based architecture. An SBST has a non-intrusive nature as it utilizes the existing on-chip programmable controller and related instruction set for both test pattern generation and output data evaluation. Therefore, SBST can provide high quality at-speed testing in normal operational mode without any performance, area and power overhead. Self-testing approaches for microprocessor have been proposed in the literature [7, 8, 9, 10, 11] and a review of some of them is given by A. Krstic et al. [12]. An outline of the embedded SBST concept is shown in Figure 1. Self-test code and data are downloaded into instruction and data memories from an on-chip memory dedicated for the task of periodic in-field testing or from a low-speed, low-cost external ATE. Subsequently, these self-test codes are executed at the processor actual speed (at-speed testing) and test responses are stored back in the on-chip memory.

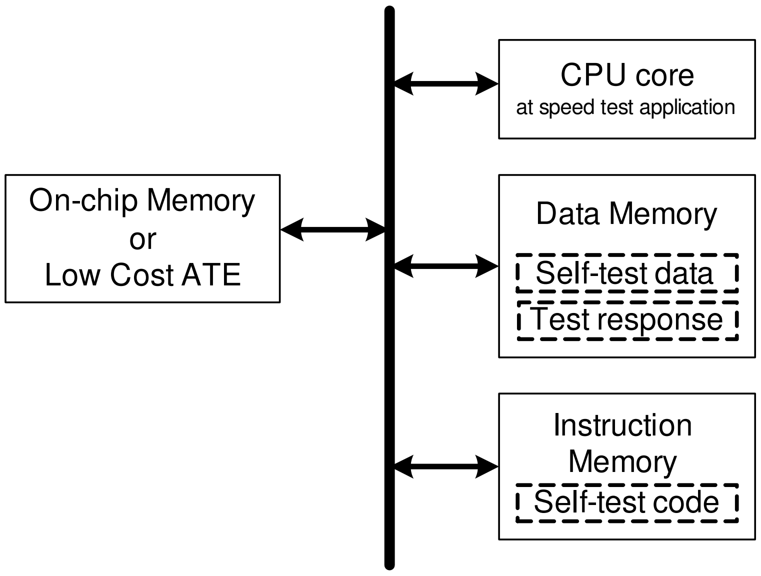


Figure 1 : Software-Based Self-Testing Concept

The published SBST approaches can be classified in two different categories functional test and structural test. For the SBST techniques which are *functional* [7], [8], they use random instruction sequences, operations and operands. Such techniques have low test development cost due to their high abstraction level, they can also achieve high fault coverage with execution of a large number of test instruction sequences. Thus, the derived test program is large and requires excessive test execution time. Also, long fault-simulation time is required for fault grading. For the SBST techniques which are *structural* [9, 10, 11]; they target the components of processor with high structural fault coverage and regarded as promising techniques for efficient testing of a processor. Firstly, the structural SBST identify processor components and their corresponding operations. Secondly, for every Component Under Test (CUT) and for every operation of the CUT, test patterns are generated targeting structural faults. Thirdly, the test patterns are transformed to self-test routines (consisting of processor instruction sequences) which are used to apply test patterns to the inputs of the CUT and collect test responses from the outputs of the CUT. All self-test routines together constitute a test program with stringent requirements in code size, data size, and execution time.

WSN nodes are usually battery powered and their replacement and recharge may not feasible during their lifetime. Therefore, in addition to the test quality, energy consumption is a major concern in testing WSN nodes. Energy optimization of the hardware, and even more so for the 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 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 a wireless node SBST based on an accurate energy model is thus beyond our reach. Instead, for SBST development without a comprehensive energy model, we can rely on measuring the exact energy consumption profile. For this, we require a complete WSN node outfitted with accurate current sensing circuitry.

Our SBST 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 specifications. The test time and energy consumption are further reduced by the interleaving of module tests, 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.

1.2 Thesis Outline

Following the above introduction, chapter 2 will give a literature review of testing methodologies for various kernel components of WSN, software energy optimization techniques and wireless network performance testing.

In Chapter 3, the general WSN node architecture and network test architecture of WSN node are presented first. The SBST method based on instruction set for CPU core is presented next. The March FT algorithm is used for word-oriented embedded FLASH memory testing. We also present the characterization and testing of RF module based on wireless network communication.

In Chapter 4, a current measurement configuration is proposed to monitor the energy consumption of the wireless node. Operand's Hamming distance and weight and selection and combination of low power instruction are considered for the instruction-level low energy concern. Time interleaving of various module is a system-level energy reduction method which further reduces the software energy consumption. The efficiency of the proposed energy reduction methods is experimentally proved by accurate current measurements.

Chapter 5 provides a large number of measurements about the relationship between communication properties that can help to build the statistical models for WSNs. Based on the current measurement architecture, the energy consumptions of wireless nodes under different operating modes are also presented.

Finally, Chapter 6 summarizes the contributions in this thesis.

Chapter 2

Background

In this chapter, we present the related published works for the testing of kernel components in WSN node: microprocessor, FLASH memory and RF transceiver. After that, the software energy profiling and optimization techniques are introduced. The popular communication models that are used to evaluate WSN performance are discussed at last.

2.1 Microprocessor Testing

We have already briefly mentioned the SBST approaches for microprocessor testing in chapter 1. In this section, we will discuss evolution of microprocessor testing methods and talk in detail about the different SBST approaches applied for microprocessor testing.

2.1.1 Traditional Structural Testing Methodology

The definition of structural testing is that a form of testing whereby the goal is to verify the structure of a chip (the wire connections and the gate truth tables). The opposing form of testing is functional or behavioral testing. A microprocessor may be considered to be a structure of interconnected logic gates and storage elements. Structural testing can be the most difficult to accomplish in a practical manner since much added hardware is usually required to increase the controllability (the ability to place nets, nodes, gates, or sequential elements to a known logic state) and observability (the ability to observe nets, nodes, gates, or sequential elements after they have been driven to a known logic state) of internal signals. Most structural testing is designed to augment functional tests. The generation of structural tests requires access to netlists, the interconnection details of the design, and fault simulation to evaluate the quality of the test vectors. Consequently, structural testing is very rarely used on commercially available microprocessors of 100,000 transistors or more unless design for testability has been employed during its design phases to partition the machine into subunits that can be tested separately. If structural information is available for test-set generation, the expected fault coverage for this method can fall in the range of 80-99 percent, depending on the size of the subunits to be individually tested.

2.1.2 Traditional Functional Testing Methodology

Functional test is defined as a form of testing where a design element is tested by the application of functional, operational, or behavioral vectors [20]. Functional test should guarantee that the chip meets system specifications. The development of a functional test for a microprocessor begins with choosing a system model describing the behavior of the microprocessor and a fault model describing deviations from correct behavior of the system model.

In the early 1980's, Thatte, Abraham and Brahme [13, 14] proposed a graph model (s-graph) at the Register Transfer Level (RTL) to represent a microprocessor and used functional level fault models for instruction level test generation. It is considered a landmark paper in processor functional testing. This approach models the system behavior of a microprocessor as an "execution" graph that represents memory elements as nodes. Data flow in the microprocessor is modeled by directed arcs between the nodes involved in the transfer. A fault model is used to describe incorrect operation of the "execution" graph. The microprocessor is identified by a set of functions such as i) register decoding function ii) instruction execution iii) data transfer and data storage function and iv) data manipulation functions. A functional fault model is then developed for each of these functions and tests are generated to detect all the faults in the fault model. Using the graph method, tests can be generated without detailed knowledge of how the instruction execution and control function is implemented. Functional testing methods ignore the internal hardware structure and generate the test sequence based on the instruction set. Since then, many processor functional testing methodologies have been proposed [15, 16, 17, 18, 19].

The various functional testing approaches may be classified into the following two categories: **1**. tests based on functional fault models [15, 16, 17, 18]; and **2**. tests based on the "checking experiment" principle, without assuming any faults models [19]. Those traditional functional test approaches had a high level of abstraction, but required a large amount of manual effort. Large numbers of tests demand a higher storage capacity of the ATE and longer test application time. To reduce the number of tests, some fault grading approach are proposed with grading of faults according to probabilities by which they can occur in the circuit [18], but very little fault grading was done on structural processor netlists, while high fault coverage was not guaranteed. The "checking experiment" method assumes no functional fault models. Instead, it emphasizes the conformity of the hardware implementation with the architecture level specification. There is a detailed report of this methodology by C. Bellon et al. [19] with application to design validation, failure analysis and highly dependable system validation. They conclude that the only suitable approach to functional testing of microprocessors was an extensive test of the representative functionality. However, they also concluded that functional test generation could not provide an alternative to structure-level test generation and manufactures still had to pay more attention about gate level test generation.

Developing good test programs that can guarantee high fault coverage is not easy, even for the designer knowledgeable of the internal structure of chip. An exhaustive microprocessor test set is impossible to implement since an exhaustive test of all possible combinations of instructions, addressing modes, and data patterns would take years to complete. Fault coverage is the only standard to check the efficiency of a proposed testing methodology. Fault coverage is defined as the metric of how many faults are exercised and successfully detected (their fault effect is observed) versus the total amount of fault content in the circuit under test [20]. A precise estimate of the fault coverage can only be made by running fault simulations with the exact application instruction sequence as well as with the microprocessor detailed internal structure. In general, for large microprocessors, fault coverage of 60 to 80 percent is achievable. The length of this functional test is generally O (n_R * n_I), where n_R is the number of registers, n_I is the number of instructions, and O (*) indicates the order of the quantity enclosed in parentheses. So, for complex microprocessors with a large number of registers and instructions, the test set can be lengthy. Furthermore, most of them rely on external ATE to feed the input test patterns and monitor the test response, in contrast with the SBST approaches that apply at- speed in a self-test mode.

2.1.3 SBST of Microprocessor

As mentioned before, the SBST approaches applied to microprocessor can be classified in two different categories. The first category includes the SBST approaches that have a high level of abstraction [7, 8] and are functional in nature. The second category includes the SBST approaches that are structural in nature [9, 10, 11] and require structural fault-driven test development.

A functional test methodology [7] generates a random sequence of instructions that enumerates all the combinations of the operations and selected operands. Test development is performed at a high-level of abstraction based on instruction set architecture. However, since test development is not based on a priority fault models, the high fault coverage can only be generated by using large test code sequences. Furthermore, the use of large code sequences results in excessive test application time and very long fault simulation time for fault grading.

A self-test method [8] combines the execution of microprocessor instructions with a small amount of on-chip hardware that is used to provide a pseudorandom instruction sequence, thus creating a randomized test program along with randomized data operands.

Besides the fact that the proposed methodology cannot be considered as a “pure” SBST methodology due to the insertion of extra test hardware, the manual effort required for test program development is high, while the pseudorandom test sequences result in a very long test application time.

The concept of self-test signatures is introduced and a structural testing methodology for processor cores is presented by Chen and Dey [9]. There are two stages for the proposed SBST methodology. At the test preparation stage, pseudorandom patterns are used for each processor components in an iterative method with the consideration of the instruction set constraints, based on the knowledge of the gate-level netlist of every component. At the test execution stage, pseudorandom test patterns developed in the previous stage and producing self-test signatures are stored in embedded memory. Then, the pseudorandom test patterns are applied by software test application programs and responses are collected into memory again. At the test preparation stage, as an alternative, gate level automatic test pattern generator can be used to generate test patterns for processor components in the iterative constrained test generation method. The self-testing step involves the application of the component tests using a software tester, which consists of an on-chip test pattern generation program, a test pattern application program, and a test response analysis program, as shown in Figure 2. This methodology is restricted by the need of gate-level details of the processor structure. Such information may not be available, but even in the case that it is actually available, the instruction set test generation for functional modules of the processor is a very time consuming task, which may not lead to acceptable fault coverage. Besides, the pseudorandom nature of the methodology leads to large self-test code, large memory requirements and excessive test application time.

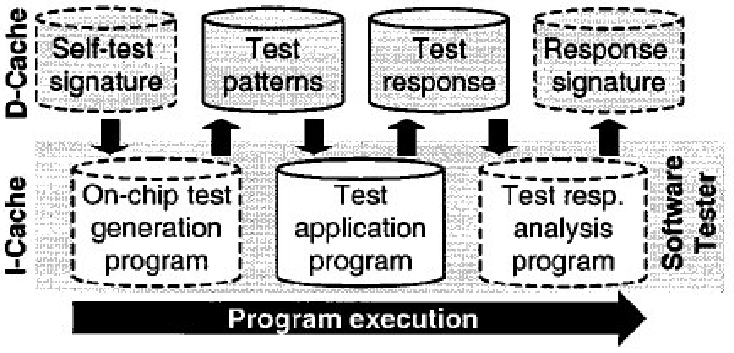


Figure 2 : Self Test Methodology

A SBST methodology for embedded processor cores [10] that is based on the knowledge of the instruction set architecture of the processor and its RTL description is proposed. The RTL description showing the connections among the functional units of the processor, the storage elements and the steering logic modules is usually available information and is much more easily managed than a detailed gate-level netlist. Therefore, a limited engineering effort is required. This methodology is based on the application of deterministic test patterns targeting structural faults of individual processor components. The deterministic tests patterns are not automatic test pattern generator generated but are developed by the methodology in order to excite the entire set of operations that each component performs. For each component operation, a basic self-test routine is developed based on a deterministic test set which maps each operation to a processor instruction sequence. The derived self-test code is compact due to the use of small regular test sets. The regularity of the basic test sets for the functional module components is essential since it is the driving force for the small size of the self-test code and thus its small memory requirements.

The SBST [11] goes a further step by defining different test priorities for processor components and classifying them according to the defined priorities, proving that high-level test development based on ISA and RTL description of a complex processor ISA can lead to low test cost without sacrificing high fault coverage, independently of complex processor implementation and gate-level structure. The proposed SBST methodology has the advantages of the functional-based SBST methodologies like test development at high level using the ISA, but goes one step deeper, using RTL information and a divide-and conquer approach targeting individual components with respect to the stuck-at fault model, thus providing very high fault coverage. The SBST methodology [11] has the advantage over other SBST methodology [9] that it is an independent test development strategy with gate-level net list required only for fault grading purpose. Furthermore, assigning different test priorities to the processor components and then developing low-cost test routines for the most critical components of the processor results in smaller on-chip memory requirements, shorter test program download and test application time while the fault simulation time required for fault grading is minimized, thus providing an efficient and low-cost alternative structural approach.

2.2 FLASH Memory Testing

The rapid-growing market of portable electronic devices such as mobile phones and digital camera has created a large, important demand for FLASH memories. FLASH memories are a type of non-volatile memory based on floating-gate transistors [25]. There are two kinds of FLASH architecture: NOR and NAND FLASH memory arrays, whose programming mechanisms are channel hot electron injection and Fowler-Nordheim tunneling, respectively. They can store charge or remove charge from the floating gate by electrical means. Their in-field programmability and low power consumption make FLASH memories widely used in portable devices. New generations of FLASH memory have higher capacity and lower access time than their predecessors. Various charge mechanisms, cell structures, and array architectures have been developed in the past few years [21]. Furthermore, FLASH memories can be embedded in logic systems to allow software updates. Embedded FLASH memory cores thus play a very important role in the high performance and complex systems.

There are many challenges for embedded FLASH designs. Reliability has been considered as the primary test issue for FLASH memories [22]. They are commonly tested for disturbance problems, including read-disturb fault, program-disturb fault, and erase-disturb fault. March test algorithms are widely used for random access memories (RAMs) testing [23]. March test is a test consisting of a set of March element; each March element performs a finite number of operations on a cell before proceeding to the next cell. March tests are the most efficient tests for detecting stuck-at fault (SAF), transition fault (TF), address decoder fault (AF), and state coupling fault (CFst). However, March tests are not suitable for FLASH memories due to very different physical operations. Most FLASH memories can do random read and random program (write 0), but cannot do random erase (write 1). Instead, they support block erase or chip erase. The erase cycle can be initiated from within FLASH memory. When FLASH erase option is initiated, CPU is held while the erase cycle completes. Therefore, March tests for RAMs are in general not applicable to FLASH memories. Recently, systematic approaches for testing FLASH memories [24, 25] were proposed, in which fault models capture the characteristics of disturbances in the memory structure and test algorithms are used to detect these faults. Furthermore, the FLASH algorithms [24, 25] are March-like, which facilitate the coverage analysis and the test pattern generation. The IEEE 1005 Standard Definitions and Characterization of Floating Gate Semiconductor Arrays [26] is used to derive realistic fault models for FLASH memory, and then used to develop March-like test algorithms for those fault models. The possible disturb mechanisms for NOR-type stacked gate FLASH memories include gate program disturbance (GPD), gate erase disturbance (GED), drain program disturbance (DPD), drain erase disturbance (DED), and read disturbance (RD). A GPD fault occurs when a cell under program (selected cell) causes another unprogrammed cell (affected cell) on the same word line to be programmed. A GED fault occurs when a cell (selected cell) under program causes another programmed cell (affected cell) on the same word line to be erased. The DED fault occurs when a cell (selected cell) under program causes another programmed cell (affected cell) on the same bit line to be erased. A DPD fault occurs when a cell (selected cell) under program causes another unprogrammed cell (affected cell) on the same bit-line to be programmed. An RD fault occurs on the selected cell. Because the bias conditions for reading are the same as for programming, hot electrons can be injected from the channel into the FG even if it is at a low gate voltage. Several functional fault models commonly used for testing RAMs are also considered useful for testing non-volatile memories [23], including SAF, TF, stuck-open fault, address decoder fault, and CFst.

Recently, there have been some efforts to test FLASH memory fault models. The March EF [24] detects FLASH disturbances with low test complexity. Moreover, a March FT algorithm for FLASH memory [25] goes further than March EF. March FT can be presented as follows:

$$(f); \Downarrow(r1, w0, r0); \Updownarrow(r0); (f); \Uparrow(r1, w0, r0); \Updownarrow(r0)$$

$\Uparrow$ ($\Downarrow$) - an increasing (decreasing) address order e.g. from address 0 to address n-1 (or vice versa),

$\Updownarrow$ - address order is irrelevant,

W0/1 - writing a 0/1 into a cell,

R0/1 - reading a cell with expected value 0/1,

f - block erase of the FLASH.

Compared to March EF, March FT adds two additional read operations at the second and fifth March elements. With two additional read operations, the coverage of stuck-open fault and state coupling fault all reaches 100% from 50% and 75% when tested individually. March FT algorithm has the advantages that they are more regular, easier to generate and cover more functional faults. Compared to the March EF algorithm [24] in which the topology information is necessary to explicitly perform row and column operations, March FT does not rely on the array geometry, which make it more general and applicable to FLASH memory. Most memories in the modern applications are word-oriented memories. To cover intra-word coupling fault and intra-word gate program disturbance, 1 and 0 presented in the March FT algorithm should be replaced by {1111, 1100, 1010} and {0000, 0011, 0101} for a 4-bit word-oriented FLASH memory, for example.

2.3 RF Transceiver Testing

Nowadays the RF devices are widely used in wireless electronic products. A high enough quality of the RF part is essential to achieve the intended performance of the Wireless product. The problem of verifying correctness of the RF circuitry is important, and to perform test expensive RF equipment (ATE) still has to be employed**.** On the other hand, the advancing complexity and performance of present RF transceivers are pushing the ATE to the edge of its limits. In this context an alternative approach based on the BIST is appealing and can alleviate the problem.

Some loopback test schemes are proposed to test RF transceiver to avoid the use of ATE [27, 28, 29, 30, 31]. In the loopback test configuration, the transmitter output is looped back to the receiver and the test response is captured in the baseband at the output of the receiver. The advantage of this loopback scheme is that the receiver and transmitter subsystem specification values are decoupled and calculated at the receiver baseband (commonly, a DSP is used to analyze the test response). An additional receiver path is used in order to increase observabiltiy [27, 28]. In [27], the authors propose in a conceptual manner to use a dedicated down-conversion path on the same chip for down-converting the transmitter output back to baseband; at the same time, by the means of an RF coupler, it proposes to feed the transmitter output back into the receiver input. However, the concept presented has not been supported by experimental data. In [28], a pseudo-random test stimulus is proposed to measure the adjacent channel power rejection specification of the transceiver system. The response of the subsystem to this stimulus has a large number of frequency bins in the up-converted RF frequency spectrum. Therefore, measuring the power over such large number of tones requires long test times and expensive RF ATE. In [29], an additional single bit DA converter is used to test the receiver path. In [30], an optimized periodic bitstream generated by the DSP and modulated by the transmit modulator at the baseband is used as the test stimulus. The output of the transmit subsystem to this stimulus is fed back into the receiver subsystem using minimal on-board hardware. The test response at the output of the receiver subsystem is processed by the DSP and the linear and the nonlinear specifications of the transmit and the receive subsystems are computed.

An enhanced BIST approach is proposed to deal with the spot defects that can severely degrade the performance or result in chip malfunction [31]. The spot defects are layout-dependent and result in electrical opens and shorts or can produce resistive breaks or bridges. The fault models follow those abstraction levels: layout, circuit and functional block. A block diagram of the proposed RF transceiver is shown in Figure 3. All the functional blocks with the exception of the Power Amplifier (PA), RF filter, diplexer and antenna are integrated on one chip. To enable BIST the test amplifier (TA) has been added to the chip. The proposed BIST would be arranged in a loop-back configuration, where the base-band processor serves both as a stimulus generator and response analyzer. The test loop comprising of the transmitter (Tx) and receiver (Rx) path is closed by the test amplifier. A possible local test loop aimed at the DA/AD converters is marked with a dashed arrow (filters could be included in the test).

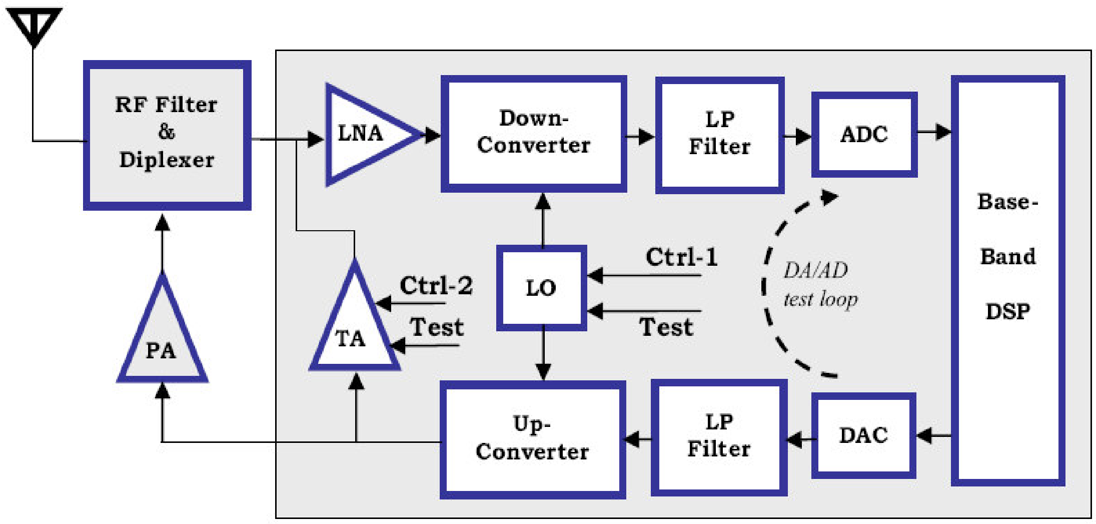


Figure 3 : Block Diagram of RF Transceiver with BIST

They verified that the existence of spot defects at the layout level and resistive break or bridges at the circuit level can be mapped into the impairments in RF specifications of the functional blocks such as gain, noise figure or IP3. Increased noise figure should display low signal-to-noise ratio (SNR) as test response. There is a relation between spot defects in the transmitter and the SNR of the response. The relation drives them to choose a test with a response sensitive to noise figure and SNR. Hence, the proposed BIST [31] uses the PRBS as the baseband stimulus, and the primary test response would be the bit-error rate measured in the baseband.

The loop-back BIST architectures applied to RF front-ends have the advantages that the high performance and expensive ATE is avoided by the use of BIST. However, some extra circuitry must be added to provide the loopback path. Some faults happened in the circuit under test may be masked by the added circuits. The probability of a fault is roughly proportional to the chip area, and the probability that the fault is located in the test circuitry is proportional to the ratio of the respective areas occupied on a chip [31]. The added test circuitry will increase probability of the faults happening. Furthermore, it is not feasible to access the internal nodes of COTS RF transceiver and add extra circuit for loopback. In this thesis, we will propose an SBST methodology for RF testing with the aid of wireless communication among WSN nodes. The SBST methodology avoids the adding of extra circuitry. It is a flexible test scheme without area and performance loss.

2.4 Instruction-level Power Optimization Methodologies

The increasing popularity of low power mobile product drives the need for analyzing and optimizing power consumption in all parts of a system. In the last decade, researchers have devoted increasing efforts to reduce the average power consumption in VLSI systems during normal operation mode, while power consumption during test operation mode was usually neglected. However, during test application, the circuits are subject to an activity level higher than normal: the extra power consumption due to the test application may thus give rise to severe problems in the circuit reliability. Moreover, it can dramatically shorten the battery life when periodic testing of battery-powered systems is considered. Nowadays, energy consumption of software has emerged as an important metric of a system. Especially for embedded systems, there is a high demand for optimization techniques that enable energy reduction for software, since an increasing number of applications are powered by batteries. In order to systematically analyze and assess this impact, it is important to start at the most practical and fundamental level - the instruction level. Accurate energy profiling and analysis at this level is essential to evaluate the software in terms of energy consumption, and also to help software developers in their search for low power software implementations [32, 33, 34].

The average power consumed by a system is given by: $P = I * V_{dd}$, where P is the average power, I is the average current and V_{dd} is the supply voltage. The energy, E, consumed by a program is further given by: $E = P * N * \tau$, where P is the average power, N is the number of clock cycles taken by the program and τ is the clock period. Thus, the ability of testing the current drawn by the CPU during execution of the program is essential for measuring its power/energy cost. Loops of hundreds of the same instruction or instruction sequences are performed on the processor, and the average drawn current is used to get the energy consumption of the instructions.

Due to the lack of accurate power models, measurement-based approaches are widely used for software energy analysis. The energy consumption of software is characterized by examining the data obtained from real hardware. The advantage of the measurement-based approaches is that the resulting energy model is close to the actual energy consumption behavior of the processor, because the data is acquired from the hardware itself. The majority of work published on the field of measurement-based techniques refers to the Tiwari method [32] as a base point. In this approach, the software energy requirements consist of the unique base cost for each instruction and the inter-instruction effects. The base cost for an instruction is defined as the average current drawn by this instruction when executed repeatedly in a tight loop, multiplied by the number of cycles taken by each instance of the instruction. On the other hand, the inter-instruction effect is defined as the additional power cost incurred by executing different instructions sequentially. The Tiwari technique is used at other applications [35, 36]. These techniques provide a simple framework for software energy estimation by summarizing the energy consumption by instructions in the form of a table. However, by relying on the average current, they largely ignore the detailed impacts of various factors that affect the energy consumption at the instruction level. Moreover, these techniques do not provide the information about the energy variation due to various aspects of instructions such as the instruction fetch address and what the operand specifies.

In contrast to the average current measurement, instantaneous current is firstly measured by Russel et al. [37], where a digital oscilloscope is used for reading the voltage difference over a precision resistor that is inserted between the power supply and the core supply pin of the processor. Instantaneous power is then calculated directly from the voltage waveform from which average figures are extracted to guide instruction power modeling. Resistor-based methodologies suffer from supply voltage fluctuations over the processor which reduces the accuracy of the method. A technique to derive more fine-grained energy consumption is proposed by Chang et al. [38], where they measure the cycle-level energy consumption using specific measurement hardware. They also analyze the impact of various properties of instructions on the energy consumption, based on this measurement method. Using this approach, it is shown that the energy consumption of software is dependent on the properties of instructions, such as register numbers, immediate operands, Hamming distance of test vector, etc.

2.4.1 Current Measurement Architecture

Conventional processor boards are composed of memory and many other peripherals. The best way to remove systems dependent bias during measurement is to use a processor board solely composed of a microprocessor core. This is almost impossible in real systems, but we can set up an environment for testing with the above conditions.

To investigate the real power consumption of our wireless node, a current measurement scheme is built, as shown in Figure 4[1]. It measures the instantaneous current drawn by the processor during execution of the test program. 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 $D_{0\text{-}3}$, as with logic analyzers. These signals help identify different *modes* in test routines. We display these modes on the bottom of oscilloscope screen captures. A digital oscilloscope is used for reading the voltage difference over a unit resistor that is inserted between the power supply and the core supply pins of the processor. The energy consumed by test routines is calculated by integrating the product of instantaneous currents with power supply voltage V_{dd} at the node. The current integration is performed by Agilent 54830D. The detailed schematics of the PCB board for I to V conversion is shown in Figure 5[1].

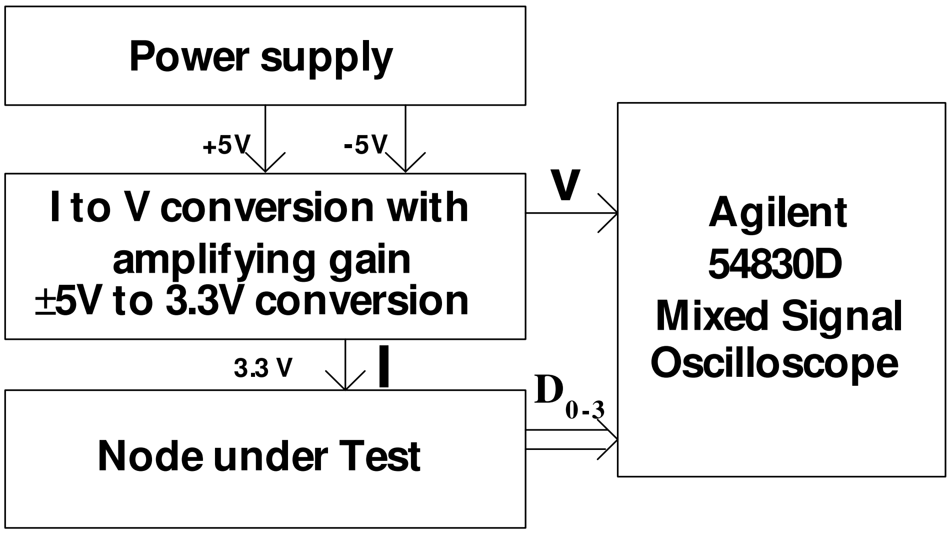

Power supply
+5V
-5V
I to V conversion with amplifying gain ±5V to 3.3V conversion
V
Agilent 54830D Mixed Signal Oscilloscope
3.3 V
I
D0-3
Node under Test

Figure 4 : Current Measurement Architecture

The +5V and -5V power supply is provided by the HP E3630A triple output DC power supply. The ±5V are used as power input for amplifier INA145 and 5V-3.3V regulator in I to V conversion board. The NUT board only contains the sole MSP430F149 microcontroller, external 32 KHz and 8 MHz oscillators, some resisters and capacitors. Port 5 (P5.0-P5.3) of MSP430F149 is used to indicate various test modes. The current to the NUT is amplified by I to V conversion with amplifying gain and is measured by Agilent 54830D 2+16 channel, 600MHz mixed-signal oscilloscope.

Figure 5 shows the schematics of I to V conversion board. The current input to the NUT board goes through a unit resistor R3 (1 Ω) and is equal to the voltage drop of the R3. Since the current cost by MSP430F149 is a few tens of mA that is too small to be measured by scope, the voltage of R3 (equal to the current to NUT since R3 is 1 Ω resistor) must to be amplified with proper gain in order to be readable by scope. A programmable gain difference amplifier IN145 is used to amplify the voltage of R3.

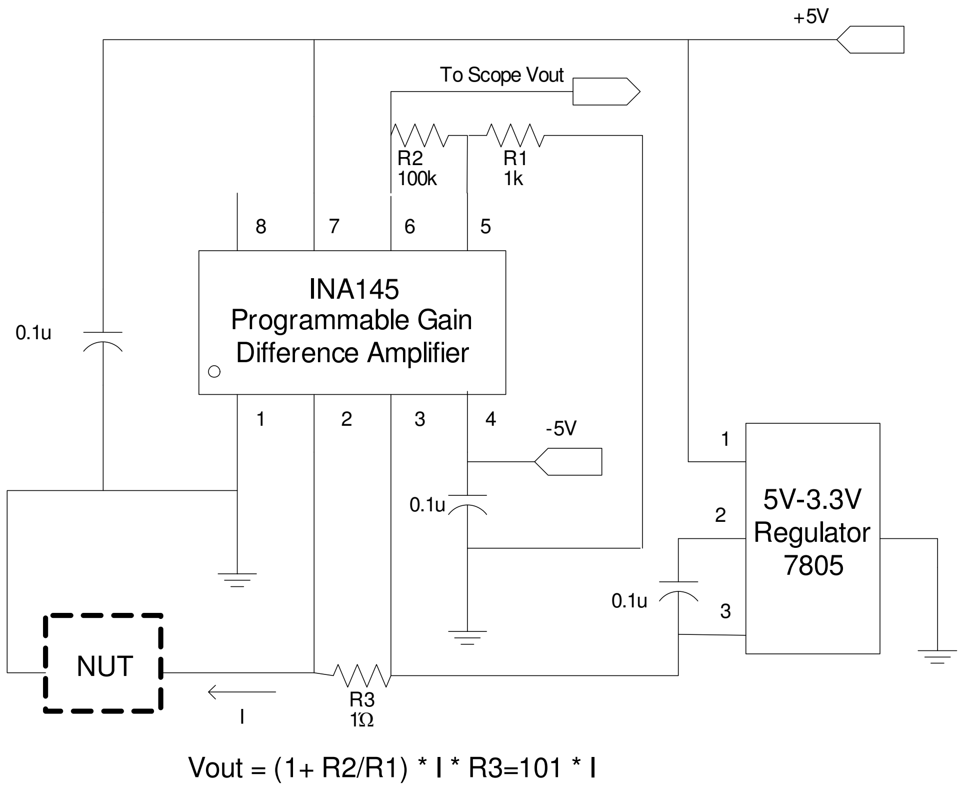


Figure 5 : Schematics of I to V Conversion Board

The INA145 is a precision, unity-gain difference amplifier consisting of a precision op amp and on-chip precision resistor network [47]. Two external resistors (R1 and R2) set the gain from 1V/V to 1000V/V. The input common-mode voltage range extends beyond the positive and negative rails (single supply: 4.5V to 36V, dual supply: ±2.25V to ±18V). Here, according to estimated current cost of NUT, we set the gain as 100 calculated by the equation of Figure 5. A ±5V power supply is provided to INA145 by the external triple output DC power supply. The proper value of Vout is measured by the Agilent 54830D. The 5V-3.3V regulator 7805 is used to provide 3.3V power to the sole chip MSP430F149 on NUT.

Figure 6 shows the photo of our real energy measurement architecture for NUT.

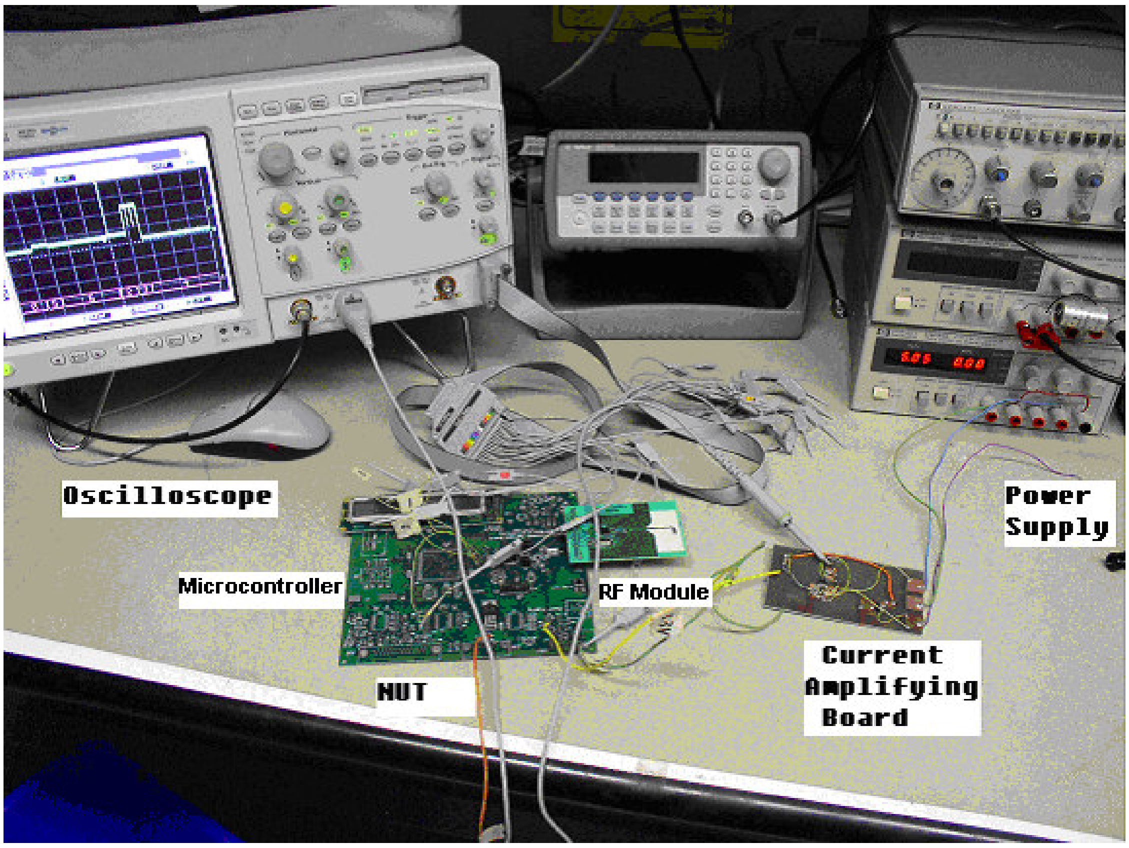


Figure 6 : Photo of NUT Energy Measurement

2.5 Wireless Network Performance Characterization

The performance of many protocols and algorithms for wireless network greatly depend on the underlying communication channel. An accurate communication model plays a key role in simulating and evaluating the network performance. Until now, there were two approaches that had been widely used in the sensor network community. They are unit disk modelling and empirical data traces. However, the unit disk models imply complete correlations between the geometric space and the topology of the network, which was proved to be wrong by many experiments [39, 40, 41]. The empirical data trace approach is difficult and costly when used to characterize large networks.

Recently, there have been a lot of efforts to empirically capture communication patterns in wireless sensor network. Low-power COTS radio transceivers chips are used to deduce properties of communication links in wireless networks in several environments, such as open space and laboratories. These hybrid models introduce empirically observed factors that modify the communication patterns based on the unit disk communication model. A non-parametric statistical technique is proposed in [42] which develop an accurate simulation of network communication environments that are statistically accurate with respect to several features that impact network protocols and algorithms in real networks. To generate these simulated environments, they construct a set of models that map communication properties such as absolute physical location, relative physical proximity and radio transmission power into probability density functions describing packet reception likelihood. For all of these models, an interval of confidence is calculated. The models can help identify future directions for developers of protocols, localized algorithms and power management strategy for wireless sensor networks.

[1] The Current Measurement Architecture and I to V conversion board are built by Jean-Samuel Chenard.

Chapter 3

Software-Based Self-Testing of WSN Nodes

Commonly, a wireless node includes a processing part and a communication part. The processing part is used to control the functionality of the node and to process and store the data. In this case, a microprocessor with embedded memory is used as the processing part. The communication part (an RF module in this case) is used to communicate amongst the nodes of the wireless network. For in-field testing, we are restricted to the at-speed BIST approaches because Automatic Test Equipment (ATE) use is impractical. Since wireless nodes are currently mostly made of IP cores and COTS parts, the possibility of adding self-test hardware is limited, and is certain to cause additional cost in hardware and energy consumption. Hence, SBST is the preferred choice. In this chapter, SBST methodologies are used to test kernel parts of wireless nodes. First, the generic architecture of a wireless node will be given. Second, a component-based SBST approach is used for CPU core testing whose aim is to find structural faults (stuck-at faults). Third, a March-type algorithm, which is implemented by a microcontroller, is used to test the embedded FLASH memory. Finally, an RF module is characterized according to the common protocol for WSNs and tested with the aid of wireless communication (Packets Error Rate –PER testing).

3.1 Overview of the Node and Test Methodology

A generic wireless node has at minimum an embedded microcontroller and an RF module, as seen in Figure 6. A microcontroller with its embedded memory, including a significant amount of FLASH, is used to control the overall node operation and process/store data. It can also be used to implement node self-testing and to interpret and/or communicate test results. An RF module combines the effects 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 some encryption and Media Access Control (MAC) protocol support.

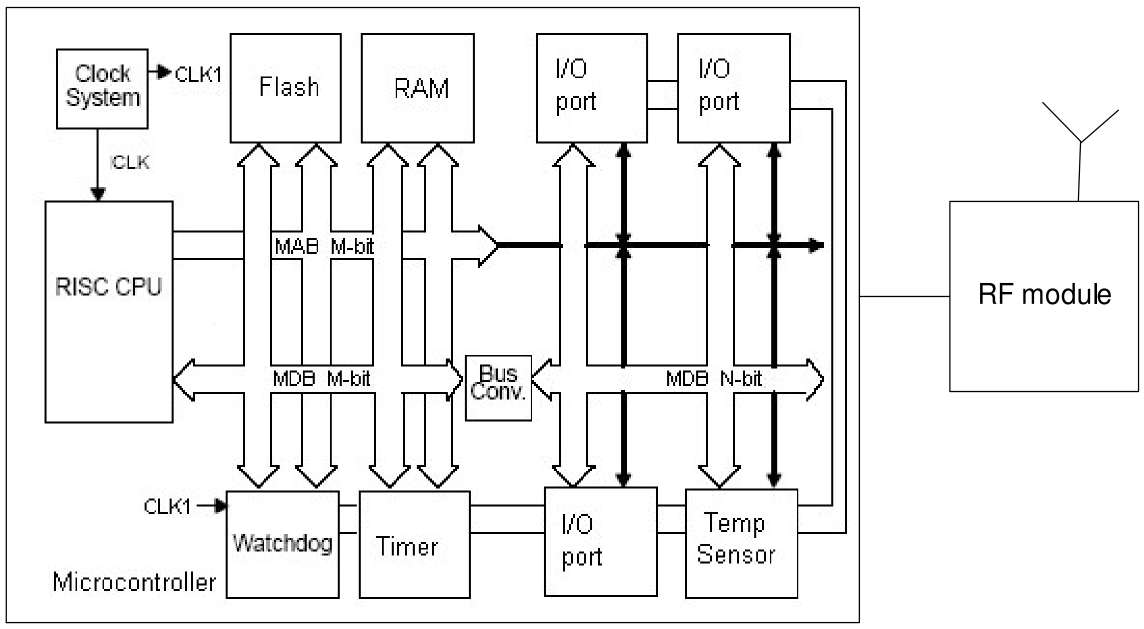


Figure 7 : Generic Node Architecture

High system *availability* requires quick fault detection and its repair; hence to avoid network latency, nodes should test themselves [2], 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) as shown in Figure 8. The node SBST scenario begins with CPU core self- testing, followed by a comprehensive test program for the rest of the system. Testing of the RF module and the wireless link characterization amongst different nodes are also performed under control of the previously tested CPUs.

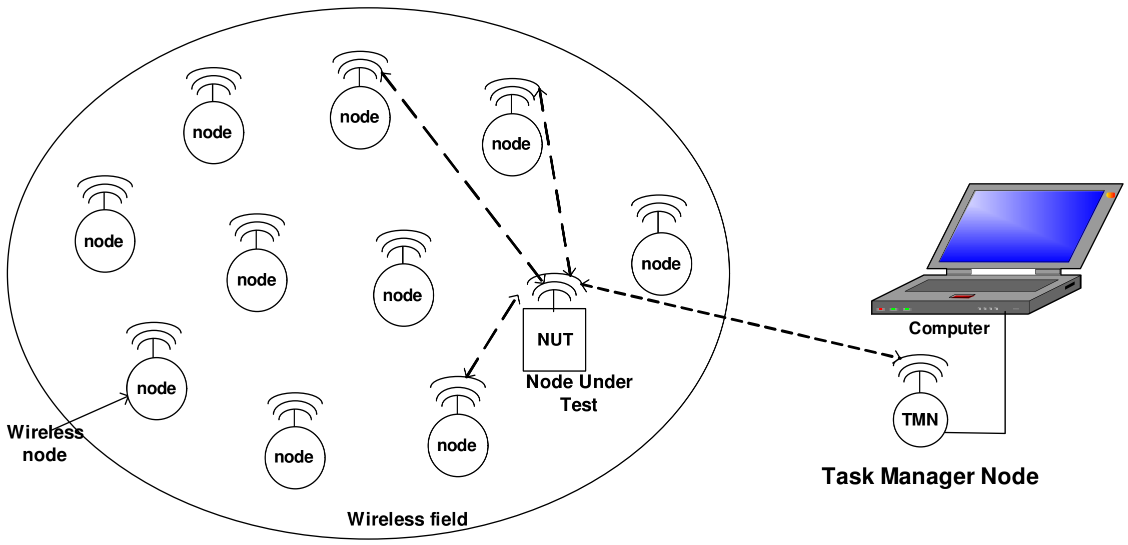


Figure 8 : Test Architecture of a Wireless Network Node

3.2 SBST of CPU

Our WSN node is equipped with a low-power COTS microcontroller, the MSP430F149. The information about the chip is all from the user manual. The gate level structure of the microcontroller is not available to us because manufacturers restrict access to the internal structure of such devices, claiming proprietary rights to this information. Based on the above constraints, a structural SBST methodology [11] is used to test the CPU. The SBST methodology is based on the application of deterministic tests targeting structural faults of individual controller 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 controller components. It combines the desirable characteristics of functional testing (like test development at high level using the processor instruction set) with a good use of the RTL information.

3.2.1 MSP430F149 Microcontroller

The MSP430F149 is a 16-bit low-power mixed-signal microcontroller provided by Texas Instruments (TI). It has the following features [43]:

- A powerful 16-bit RISC CPU core
- Ultra-low power consumption: 280uA in active mode, 1.6uA in standby mode and 0.1uA in off mode.
- Five power-saving modes
- The complete MSP430F149 instruction set consists of 27 core instructions.
- 256KB of Embedded FLASH memory and 2KB of RAM
- Two Serial Communication Interfaces (USART), functioning as asynchronous UART or Synchronous SPI interfaces.
- A Watchdog Timer Controller and a 16-bit timer/counter.

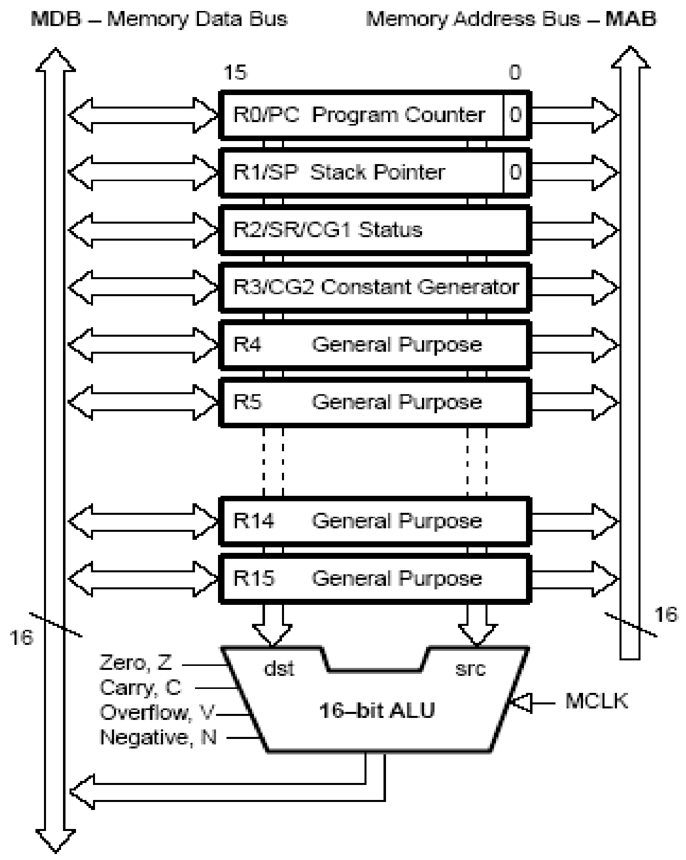


Figure 9 : Block Diagram of MSP430F149 CPU

A block diagram for the CPU core is given in Figure 9 and V, N, Z, C are explained as follows:

- V: Overflow bit. This bit is set when the result of an arithmetic operation overflows the signed-variable range.
- N: Negative bit. This bit is set when the result of a byte or word operation is negative and cleared when the result is not negative.
- Z: Zero bit. This bit is set when the result of a byte or word operation is 0 and cleared when the result is not 0.
- C: Carry bit. This bit is set when the result of a byte or word operation produced a carry and cleared when no carry occurred.

The complete MSP430 instruction set consists of 27 core instructions and 24 emulated instructions. The core instructions are those that have unique op-codes decoded by the CPU. The emulated instructions are instructions that make the code easier to write and read, but do not have op-codes themselves; instead they are replaced automatically by the assembler with an equivalent core instruction. There is no code or performance penalty for using emulated instruction. There are three core-instruction formats: Dual-operand; Single-operand and Jump. The core instruction set is shown in Table 1. Here S means the working register used for source operand and D means the working register for destination operand.

Table 1 : The Core Instruction Set of the MSP430F149

Instruction format	Mnemonic	Operation	Status Bits V	N	Z	C
Dual operand instruction	MOV	Move S to D	-	-	-	-
	ADD	Add S to D	*	*	*	*
	ADDC	Add S and C to D	*	*	*	*
	SUB	Subtract S from D	*	*	*	*
	SUBC	Subtract S not C form D	*	*	*	*
	CMP	Compare S and D	*	*	*	*
	DADD	Add S + C decimally to D	*	*	*	*
	BIT	Test bit in D	0	*	*	*
	BIC	Clear bit in D	-	-	-	-
	BIS	Set bit in D	-	-	-	-
	XOR	Logic XOR	*	*	*	*
	AND	Logic And	0	*	*	*
Single operand	RRA	Rotate left arithmetically	0	*	*	*
	RRC	Rotate left through C	*	*	*	*

instruction	PUSH	Push S onto stack	-	-	-	-
	SWPB	Swap bytes	-	-	-	-
	CALL	Call D	-	-	-	-
	RETI	Return from interrupt	*	*	*	*
	SXT	Extend sign	0	*	*	*
Jump instruction	JEQ/JZ	Jump to label if Z is set				
	JNE/JNX	Jump to label if Z is reset				
	JC	Jump to label if C is set				
	JNC	Jump to label if C is reset				
	JN	Jump to label if N is set				
	JGE	Jump to label if N xor V=0				
	JL	Jump to label if N xor V=1				
	JMP	Jump unconditionally				

* The status bit is affected
– The status bit is not affected
0 The status bit is cleared
1 The status bit is set

3.2.2 The Proposed SBST Methodology

Considering the SBST methodology in [11] and the MSP430 CPU core architecture illustrated in Figure 9, our SBST approach is implemented in three steps:

Step 1: Information Extraction.

Identification of processor components and component operations, as well as instructions that excite component operations and instructions (or instruction sequences) for controlling or observing processor registers is essential. The processing of information extraction is summarized as follows:

1. Find the set of all the processor components C.
2. Find the set O_C of all the operations of each component C, along with the corresponding control signals that the processor control unit drives to C for the execution of the operation.
3. Find the set of instructions $I_{C;O}$ that, during their execution, excite the same control signals and drive component C to perform the same operation O.

4. Find the appropriate instructions or instruction sequences that set the value of processor registers.
5. Find the appropriate instructions or instruction sequences that make the value of processor registers observable at primary outputs.

From Figure 9, there are four functional components in the CPU core: ALU, register files, data/address bus and ALU output status bus (NVZC).

O_{ALU} = {add, subtract, add with carry bit, subtract with carry bit, compare, add with carry bit and the result is decimal, and, or, xor}

$I_{ALU; O}$ = {ADD, ADDC, SUB, SUBC, CMP, DADD, BIT, BIC, BIS, XOR, AND}

$O_{ALUstatus}$ = {four status bit N, V, Z, C are set to be 1/0}

$I_{ALUstatus; O}$ = {JEQ/JZ, JNE/JNZ, JC, JNC, JN, JGE, JL}

The fault models for data/address bus and register files are SAF and the execution of all instructions will affect the values of the bus and register files. The appropriate instructions to set values of registers and make the values of registers observable will be given in detail in Step 3.

Step 2: Component classification and test priority.

Using the information extracted in Step 1, the components that appear in a processor core RTL description are classified in the following three classes:

- Functional components. The processor components that are directly related to the execution of instructions (data processing/data storage) and are in some sense visible to the assembly language programmer. These components include:

1. Computational components, which perform specific arithmetic/logic operations on data, e.g. ALU.
2. Interconnect components between processor components, which serve the data flow in a processor datapath.
3. Storage components, which serve as data storage elements that feed the data to the inputs of the computational components and capture their output. Components classified in this subcategory include special processor registers visible to the

assembly language programmer and the register file.

- Control components. The components that control either the flow of instructions/data inside the processor core or from/to the external environment (memory, peripherals). These components include the processor control unit, the instruction and data memory controllers that implement instruction fetching and memory handshaking, and similar components.
- Hidden components. The components that are added in the processor architecture usually to increase its performance, but they are not visible to the assembly language programmer.

Three classes of components have different test priorities. Test priority determines the order in which test routines will be developed for each component. High priority components will be considered first, while low priority components will be considered afterward and only if the achieved overall fault coverage result is not adequate.

From Step 1, the four components in the CPU core are all functional components. The ALU is a computational component; the data/address buses and the ALU status bus are interconnect components, and the register file is a storage component. The RTL of control components and hidden components is not available from the user manual. Our tests only focus on the testing of functional components.

Step 3: Test routine development

Development of self-test routines emphasizes using compact loops of instructions by considering the optimized software energy consumption (for details refer to Chapter 4). The method provides very high fault coverage for most types of architectures of the processor components. There are two steps to test routine development:

- Instruction Selection. For every component operation O_C derived from Step 1, we select an instruction I from the set $I_{C;O}$ that is the shortest instruction sequence required to apply the specific operand to component inputs and propagate the component outputs to the primary outputs.

- Operand Selection. Application of deterministic operands is considered to each component in order to achieve high structural fault coverage and low software energy consumption.

3.2.3 Implementation of SBST for Functional Components

Four test routines are developed for testing the ALU, the ALU status bus, the register files and the address/data bus. We describe them individually as follows:

ALU testing

All ALU instructions are executed to test ALU functionality. We take the code section to test ADD and ADDC as an example to illustrate its operation.

```
        ………..
// test with ADD and ADDC instruction//
        mov #0x0001,r7        (set register 7 with the value of 0x0001)
        mov #0xf000,r6        (set register 6 with the value of 0xf000)
        mov #0x0001, r8       (set register 8 with the value of 0x0001)
        mov #0xff00, r9       (set register 9 with the value of 0xff00)
        add r9, r6            (add the value in r9 with r6, the carry bit should be 1 now)
        addc r7, r8           (add the value in r7, r8 with carry bit)
        cmp #0x0003, r8       (compare the result in r8 with 0x0003)
        jnz Error             (if the result is not equal, jump to line named Error,
                              otherwise execute next line )
        ..………
        ……....
        jmp   End             (if no error happened, jump to the end of the routine)
        Error   mov #0x2222, r15 (if there is an error, set specified value to r15 which
                                 is the return value of the subroutine. It will notify
                                 the main routine there is error in ALU testing  )
        End                      (the end of test routine, return to main routine)
```

ALU status bus testing

There are four lines (ZCVN) in the ALU status bus; the four bits are used as conditions in jump instructions. All jump instructions are given in Step 1. By instruction and operand selection, the four lines are set to be 1 or 0 individually and the results are proved by the execution of jump instructions with different ZCVN values. A section of test code for JZ and JNZ is shown as follows:

```
// testing the zero bit with JZ and JNZ instructions//
                mov #0xaaaa, r6
                mov #0x5555, r7
                bit  r7,r6          (the value in r7 and the value in r6, the zero bit should be 0)
                jz Node1            (if the result is equal to 0, jump to line named Node 1)
                jnz Error           (if the result is not equal to 0, jump to line named Error. If
                                    error happened, report the error and the rest will not be
                                    tested.)

Node1           mov #0x5555, r6
                mov #0xffff, r7
                bit  r7,r6          (the value in r7 and the value in r6, the zero bit should be 1)
                jnz Node2           (if the result is not equal to 0, jump to line named Node 2)
                jz Error            (if the result is equal to 0, jump to line named Error)
                …………
```

Register file testing

There are sixteen 16-bit registers in MSP430F149. Since the register file has a very regular structure in the form of identical cells, the traditional memory testing methodology can be used for register file testing. We use one test of the March family called “March X” to test the register files. The fault coverage of March X includes SAFs, TFs, CFs and AFs. The scheme of March X is as follows:

.

$$\{\underset{M_0}{\Updownarrow(W0)};\ \underset{M_1}{\Uparrow(R0, W1)};\ \underset{M_2}{\Downarrow(R1, W0)};\ \underset{M_3}{\Updownarrow(R0)}\}$$

$\Uparrow$(R0, W1) and $\Downarrow$(R1, W0) guarantees that the AFs can be detected. SAFs are detected since each byte is read with expected value 0 (by M_1 and M_3) and with expected value 1 (by M_2). All <$\uparrow$/0> TFs are detected since each segment is read after a $\uparrow$ transition (W1 in M_1 then follows R1 in M_2). All <$\downarrow$/1> TFs are detected since each segment is read after a $\downarrow$ transition (W0 in M_2 then follows R0 in M_3). CFs can be totally covered by the March X algorithm; the detailed proof can be found in [23].

The March X algorithm requires a total of 6×n operations and consists of the March elements M_0, M_1, M_2, and M_3.

```
M0: for i :=0 to n-1 do
      begin
          A[i] :=0;
      End;
M1: for i :=0 to n-1 do
      begin
          read A[i]; { Check that 0 is read.}
          A[i] :=1;
      End;
M2: for i :=n-1 to 0 do
      begin
          read A[i]; { Check that 1 is read.}
          A[i] :=0;
      End;
M3: for i :=n-1 to 0 do
      begin
```

```
        read A[i]; { Check that 0 is read.}
          End;
```

The March X algorithm is based on the bit read/write operation. Our register file is 16 bits wide and the word-based test vectors are showed in Table 2.

Table 2 : Word-based Test Vectors for Register Files

March X based on bit	Word-based vectors
0	0101010101010101
	0011001100110011
	0000111100001111
	0000000011111111
	0000000000000000
1	1010101010101010
	1100110011001100
	1111000011110000
	1111111100000000
	1111111111111111

Data/address bus testing

The test routine described here only focuses on SAF of the data bus since the address bus can be exhaustively tested during embedded FLASH testing (using another March-type algorithm). The data buses under test consist of two 16-bit ALU input buses and one 16-bit ALU output bus.

```
        mov #0xaaaa, r6     (one ALU input bus is set to 1010….1010)
        mov #0x5555, r7     (another ALU input bus is set to 0101….0101)
        add  r7,r6          (add value in r7 with value in r6)
        cmp  #0xffff, r6    (compare the ALU output bus with 1111….1111)
        jnz Error           (if the result is not equal, jump to line Error)
```

```
                mov #0x5555, r6         (one ALU input bus is set to 01010….101)
                mov #0xaaaa, r7         (another ALU input bus is set to 10101….010)
                and  r7,r6              (value in r7 and value in r6)
                cmp  #0x0000, r6        (compare the ALU output bus with 0000….0000)
                jnz Error               (if the result is not equal, jump to line Error)

                jmp End         (if no error happened, jump to the end of the routine)
Error           mov #0x1111, r15        (if there is an error, set specified value to r15  ,
                                which is the return value of the subroutine. It will notify
                                the main routine there is error in ALU status bus testing)
End                                     (the end of the test routine, return to main routine)
------------------------------------------------------------------------------------------------
```

3.3 March-type Test Algorithm for Embedded FLASH Memory

The trend of incorporating growing amounts of FLASH in embedded systems will make FLASH testing 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 [23] are not applicable because FLASH cannot perform random access erase. An efficient March-type algorithm (March FT) [25] was proposed for conventional and memory disturb faults [26]. March FT has the highest fault coverage among the published approaches. A 100% percent fault coverage is guaranteed with both specific FLASH fault models and traditional memory fault models. The test routine is written in C language. The March FT algorithm can be presented as follows:

$$\underset{M_0}{(f)};\ \underset{M_1}{\Downarrow(r1, w0, r0)};\ \underset{M_2}{\Updownarrow(r0)};\ \underset{M_3}{(f)};\ \underset{M_4}{\Uparrow(r1, w0, r0)};\ \underset{M_5}{\Updownarrow(r0)}$$

Here, f means block erase of the FLASH.

March FT Algorithms:

```
$M_0$:   erase FLASH segment block
$M_1$: for i :=0 to n-1 do
       begin
              read A[i]; { Check that 1 is read.}
              A[i] :=0;
              Read A[i];  {check that 0 is read}
       End;
$M_2$: for i :=n-1 to 0 / I ;=0 to n-1 do
        begin
              read A[i]; { Check that 0 is read.}
       End;
$M_3$: erase FLASH segment block
$M_4$: for i :=n-1 to 0 do
        begin
              read A[i]; { Check that 1 is read.}
                A[i] :=0;
              Read A[i];  {check that 0 is read}
        End;
$M_5$: for i :=n-1 to 0 / I ;=0 to n-1 do
        begin
              read A[i]; { Check that 0 is read.}
        End;
```

The byte-oriented March FT algorithm lists as follows:

(f); ⇓(rFF, w00, r00); ⇕ (r00); (f); ⇑ (rFF, w00, r00); ⇕ (r00) (f); ⇕ (w0F); ⇕ (r0F); (f); ⇕ (wF0); ⇕ (rF0); (f); ⇕ (w33); ⇕ (r33); (f); ⇕ (wCC); ⇕ (rCC); (f); ⇕ (w55); ⇕ (r55); (f); ⇕ (wAA); ⇕ (rAA);

3.4 Characterization and Testing of RF Module

Failures such as a broken/dislodged antenna or RF circuit parameter drifting can prevent the RF module from meeting the specifications. For the same reasons cited in previous sections, we concentrate on an in-field testing scenario, using our network test architecture. Several wireless nodes with exactly the same hardware structure are used to test the performance of an RF module. We continuously send test packets and calculate the required RF specifications with the aid of communication properties (Packet Error Rate). The test architecture for each specification will be discussed in detail later on. The test results will be affected not only by the RF transceiver, but also by the printed antenna performance and by the test environment. The test is held in a real wireless node application environment and the test results will indicate the performance of our RF module.

3.4.1 CC2420 RF Transceiver

The RF transceiver integrated in the WSN node is the CC2420 provided by Chipcon. The CC2420 is a true single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver designed for low-power and low-voltage wireless applications [44]. The main features of the CC2420 are listed as follows:

- 2400 – 2483.5 MHz RF Transceiver
    - Direct Sequence Spread Spectrum (DSSS) transceiver
    - 250 kbps data rate
    - Very low current consumption (RX: 19.7 mA, TX: 17.4 mA)
    - High receiver sensitivity
    - High adjacent/alternate channel rejection
    - On-chip VCO, LNA and PA
    - Low supply voltage (2.1 – 3.6 V) with on-chip voltage regulator
    - Programmable output power (8 levels from 0dBm to -25dBm)
- Separate transmit and receive FIFOs
- Very few external components, only a reference crystal and a minimum number of passives components are required.
- Easy configuration interface with controller

- 4-wire SPI interface
- Serial clock up to 10 MHz
- 802.15.4 hardware support:
  - Automatic preamble generator
  - Synchronization word insertion/detection
  - CRC-16 computation and checking over the MAC payload
  - Clear Channel Assessment
  - Energy detection / digital Receive Signal Strength Indicator (RSSI) indication
  - Link Quality Indication

The above features reduce the load on the host controller and allow the CC2420 to interface with a low-cost microcontroller. The configuration interface and transmit/receive FIFOs of the CC2420 are accessed via an SPI interface through the USART0 modules of the MSP430F149.

3.4.2 RF Characterization and Specifications Testing

Previous schemes have tested RF specifications in a loopback mode by capturing the test response in the baseband of the receiver. Here we follow a different approach, and devise a scheme to characterize the complete RF module (including the antenna) according to a common protocol for WSNs [45]. The overall testing is performed by communication among various nodes and controlled by the MSP430F149. The main specification of an RF transceiver can be calculated by the Friis transmission equation [46] (with the aid of the required PER under different test conditions in [45]) during wireless network communication.

$$\begin{aligned} P_R(dBm) &= P_T(dBm) + G_T(dB) + G_R(dB) - 32.44 \\ &-20\log f(MHz) - 20\log d(km) \end{aligned} \tag{1}$$

Where P_R, P_T are the receiving and transmitting signal power; G_T, G_R are the antenna gain of the transmitter and the receiver; f is the working frequency and d is the distance between the transmitter and the receiver.

According to the requirements of the 802.15.4 [45], the specifications of the RF module are listed in Table 3.

Table 3 : Specification Requirement for IEEE 802.15.4

Parameter	Test configuration/ (Specification)
Transmitted power (dBm)	Nominal output power: 0dBm / (> -3dBm).
Receiver sensitivity (dBm)	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)

Transmission Power & Receiver Sensitivity

Figure 10 shows the test setup for the first two specifications. Received power is determined for test packets that are continuously sent from the NUT to the TMN using the registers of the receiver IC. The *transmitted power* is then calculated from the received power, frequency and distance using Eqn. (1). Similarly, *receiver sensitivity* of the NUT is obtained by sending test packets from TMN to NUT, and searching for the transmission level at which the 1% PER is observed. Finally, Eqn. (1) directly determines the receiver sensitivity as the received power at which PER becomes smaller than 1%.

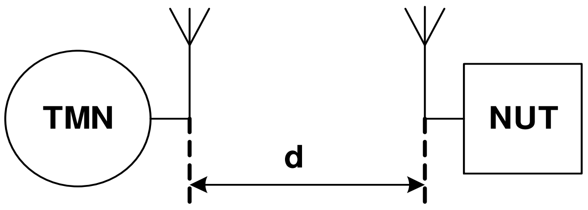


Figure 10 : Transmitted Power and Receiver Sensitivity Test

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

Adjacent /Alternate Channel Rejection

A multi-channel physical layer specification requires good interference rejection between channels. The specification distinguishes *adjacent* and *alternate* channels. For instance, channel 13 has channels 12 and 14 as adjacent, while channels 11 and 15 are alternate ones.

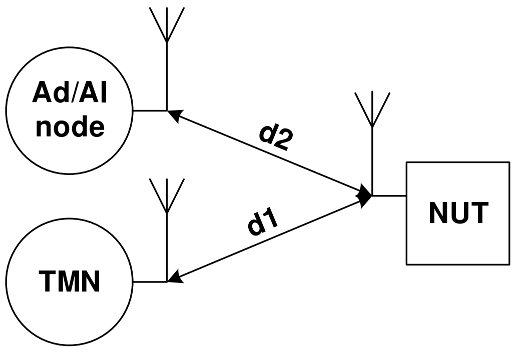


Figure 11 : Test of Adjacent/Alternate Channel Rejection

The test setup for Adjacent/Alternate channel rejection uses three nodes, as seen in Figure 11. The TMN is the transmitter, the NUT is the receiver, and the third node is the interference source. The signal level from the TMN is set to values required by the wireless standard. By sweeping through the interference levels, when PER crossing the 1% threshold, we then apply the Friis equation twice. From the given frequencies, distances *d*1 and *d*2 (Figure 11), and the received power levels, we calculate the emitted power levels. Channel rejection is then equal to the difference in the two power levels, expressed in dB.

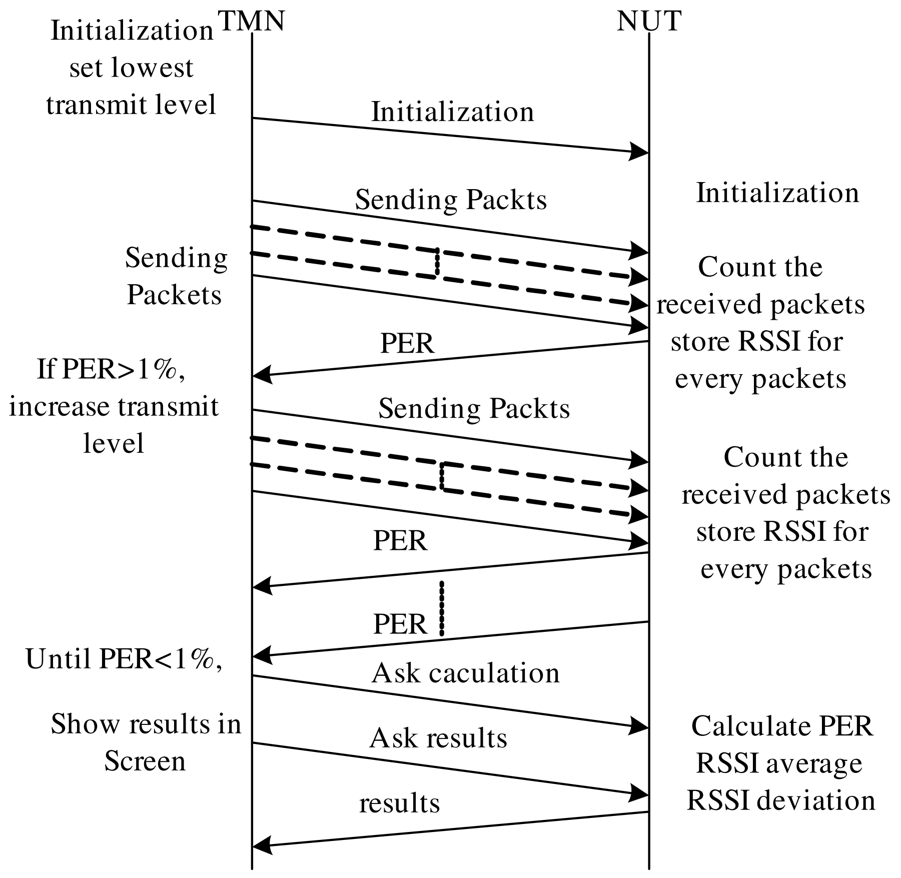

TMN
NUT
Initialization set lowest transmit level
Initialization
Initialization
Sending Packts
Sending Packets
Count the received packets store RSSI for every packets
PER
If PER>1%, increase transmit level
Sending Packts
Count the received packets store RSSI for every packets
PER
PER
Until PER<1%,
Ask caculation
Show results in Screen
Ask results
Calculate PER RSSI average RSSI deviation
results

Figure 12 : Sequence Chart of RF Test Codes

The sequence chart of basic test code (in C language) is shown in Figure 12. The detailed measurement for each specification is described as follows and the results are shown in Table 4.

Transmitted power: The transmitted power testing is a little different as shown in Figure 10. Instead, the NUT continuously sends test packets to the TMN and the RSSI of the TMN is calculated. The average RSSI of the TMN, the distance between TMN and NUT, the working frequency and antenna gains are used to calculate transmission power of the NUT by the Friis transmission equation under the required PER (<1%). The real test is implemented based on the following test conditions: f=2.405GHz (channel 11); d=0.01km; P_R=-70.05 dBm (calculated from RSSI value); G_T and G_R are experimentally -8 dB [49].

Receiver sensitivity: Programmable output power of the TMN is determined by sweeping from the lowest output level to the level that achieves the required PER (1%), with the proper distance between TMN and NUT. The receiver sensitivity is calculated by the Friis equation. The real test is implemented based on the following test conditions: f=2.405GHz (channel 11); d=0.0125km; P_T= -10 dBm (programmable by CC2420); G_T = G_R = -8 dB.

Adjacent channel rejection: Another WSN node with the exact same hardware structure (adjacent channel working frequency) with a TMN and NUT is used as an interference source. Here, we use the Friis equation twice. In the first use, the desired RF signal from the TMN is set to be -82dBm when it reaches the NUT (required from [45]) with the proper configuration of other factors in the Friis equation. There are two cases for the second use of the Friis equation.

1. Set the d_1=d_2, sweeping the output power of the interference source from low level to high level for the PER crossing 1%. The difference between TMN output power level and the interference node output power level is the result for adjacent channel rejection.
2. Set the same output level of the interference node and the TMN, and increase d_2 to find the PER crossing 1%. The distance difference (in dB) between d_1 and d_2 is the result for adjacent channel rejection.

The real test is implemented based on the following test conditions: f_1=2.405GHz (channel 11 that is used for TMN and NUT); f_2=2.41GHz (channel 12 that is used for interference source); d_1=0.015km; d_2=0.0067 km; P_{T1}= P_{T2}= 0dBm (programmable by the CC2420).

Alternate channel rejection: The test method is similar to the adjacent channel rejection test except that the interference node works at the alternate channel frequency. By considering two distances (d_1 and d_2) and two output power levels (from TMN and from the interference node), the alternate channel rejection is calculated. The real test is implemented based on the following test conditions: f_1= 2.405GHz (channel 11 that is used for TMN and NUT); f_2= 2.415GHz (channel 13 that is used for interference source); d_1= 0.015km; d_2= 0.0019 km; P_{T1}= 0dBm; P_{T2} = -25dBm (programmable by CC2420).

Based on the test configuration presented in Section 2.4.2, the RF characterization test results are shown in Table 4. The test results prove that our RF module meets the specification requirement from [45] and our method efficiently implements the characterization of the RF module.

Table 4 : RF Module Characterization

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

Chapter 4

Energy Reduction Methods for WSN Node Testing

A WSN node is usually battery-powered, making its replacement and recharge difficult and almost unfeasible during its lifetime. Energy consumption is one of the main concerns. Energy optimization of hardware, and even more so of 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.

In this chapter, we experimentally measure instruction-level energy profiling using current measurement architecture proposed in chapter 2. Based on the energy profiling, instruction-level software energy optimization methods are proposed for CPU core testing. Time interleaving of different test routines is an efficient system-level energy optimization method, especially for advanced embedded systems with large sizes of embedded FLASH memory. We implemented time interleaving in two cases: interleaving between embedded FLASH testing and RAM testing, as well as interleaving between FLASH testing and RF packet transmission. Finally, we measure software energy consumption under different wireless communication modes and conclude that the beacon transmission scheme is the optimized mode with respect to software energy consumption.

4.1 Instruction-Level Energy Reduction Methods

According to [32], the energy consumed during the execution of instructions can be identified as: 1. The base costs of instructions. 2. Overhead costs between adjacent instructions (inter-instruction cost). The base current of an instruction is measured by putting several instances of the target instruction in an infinite loop. If a pair of different instructions, say i and j, is put into an infinite loop for measurement, the current is always different from the average of the base cost of i or j. The difference is called the overhead cost of i or j, and is considered as inter-instruction cost. The total energy consumed by a program is the sum of the total base costs and the total inter-instruction costs, over all the instructions executed. The contribution of the inter-instruction costs remains small. Most of the inter-instruction costs are less than 5% of the corresponding base cost [35].

Except for the above mentioned pure base cost and inter-instruction cost, some published work [35, 38] indicates that there is a dependency between energy consumption of the instructions and the values of their parameters (operand values, addresses). There are a large number of experiments showing that the opcode, the instruction fetch address, register value, register number, data fetch address and immediate operand value can significantly effect the overall software energy consumption: these are called energy sensitive factors. From the measurement result in [38], the power consumption is proportional to the Hamming distance between previous and current values, or the number of 1's (weight) in the current values of energy sensitive factors. Compared to the pure base cost and inter-instruction cost, the above result offers much more information regarding various software-level power reduction techniques, because it shows that each cost is not a constant but a function of Hamming distance or weight. The base cost and inter-instruction cost are useful for power estimation. However, for reduction purposes, they are less important than the energy sensitive factors because there are fewer optimizations that can be made with them.

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 [50]. The optimization criterion is expressed here in terms of software energy consumption, rather than the program length, with the instruction energy profile obtained from measurements.

In the absence of good models, we ultimately rely on measuring the node current during a test. By repeatedly executing short instruction sequences we obtain energy consumption profiles accurate to the instruction level, which account for energy sensitive factors and addressing modes, as shown in Figure 13 and Table 5. Energy profiles obtained through such current measurements let us employ instruction-level energy reduction methods based on the exact knowledge of the per-instruction energy consumption. Methods used for energy reduction include instruction selecting and combination with least CPU cycles and operands selection with least Hamming distance and weight. The exact examples will be given to show how to implement the operand and instruction selections in the real test routines at next section.

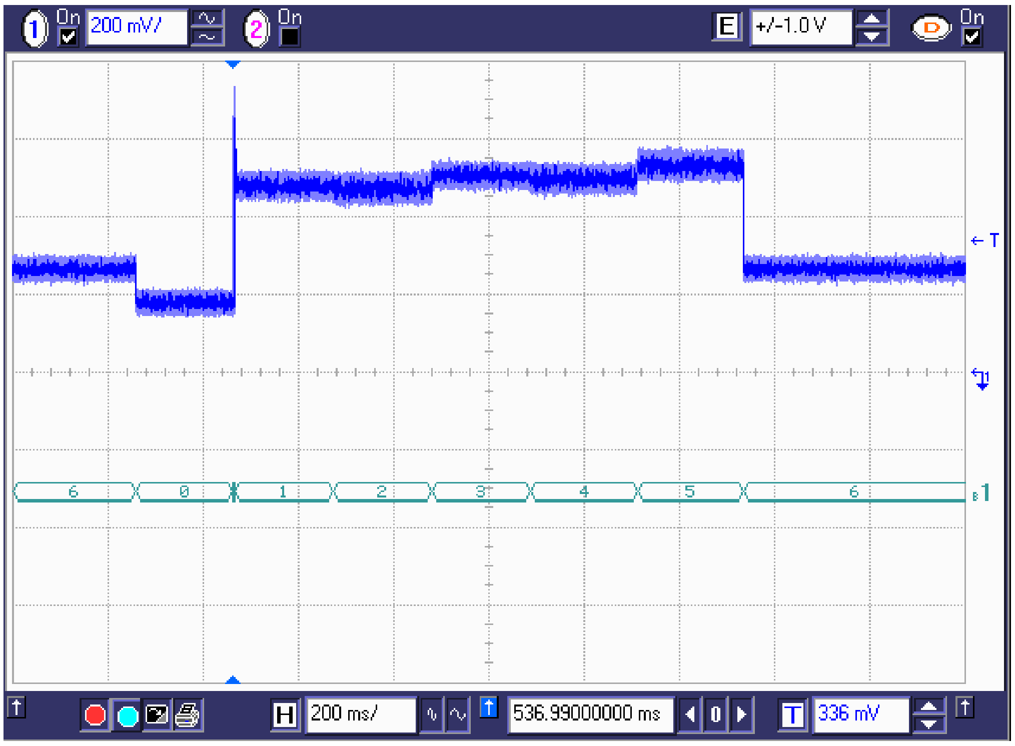


Figure 13 : Instruction-level Power Consumption

Table 5 : Modes for Instruction-level Power Consumption Test

Mode	Description
1	Operand with the maximum Hamming distance (repeatedly execute mov 0xFFFF, Rn and mov0x0000, Rn, Hamming distance is max=16)
2	Operand with the minimum Hamming distance and minimum weight (repeatedly execute mov 0x0000, Rn and mov0x0000, Rn, Hamming distance is min=0, weight is min=0)
3	Operand with the max weight and min Hamming distance (repeatedly execute mov 0xFFFF, Rn and mov0xFFFF, Rn, Hamming distance is min=0, weight is max=16)
4	Mov instruction with register addressing mode
5	Mov instruction with immediate addressing mode

4.1.1 Instruction Selection and Combination

Starting with SBST code generation such as in 3.2.2, our procedure performs a series of instruction selection towards obtaining energy-optimized test code. We have identified the set $I_{C;O}$ which consists of those processor instructions I that, during execution, cause component C to perform operation O. The instructions that belong to the same set $I_{C;O}$ have different controllability/observability properties since, when operation O is performed, the inputs of component C are driven by internal processor registers with different controllability characteristics while the outputs of component C are forwarded to internal processor registers with different observability characteristics. Therefore, for every component operation O_C, we select an instruction I from the set $I_{C;O}$ (the instructions with different addressing mode are regarded as different instruction) that results in selecting the shortest instruction sequences (least CPU cycles) required to apply the specific operand to component inputs and propagate the component outputs to the processor primary outputs.

To minimize SBST energy, an *instruction selection* step (as shown in Example 1) selects instructions requiring the least amount of CPU cycles, while preserving the test coverage.

Example 1: *Let the March X algorithm test a register file. The operation set is* O_{Reg} = {Write 0, Read 0, Write 1, Read 1}. *Here, the WriteX element can be implemented by instruction set* I (Reg, WriteX) = {mov X, Rn}. *We compare the instruction sets that are used to implement the operation* O_{Reg} = {Write 0} *before and after instruction selection.*

```
I1:              I2:
mov 0, R1        mov 0, R1
mov 0, R2        mov R1, R2
…                …
mov 0, Rn        mov Rn-1, Rn
```

The instruction sequences in `I1` *move the immediate number 0 to each register of register file. The value in the register file should be all 0 after the instruction set* `I1`. *If there is stuck-a-1 fault in register, it will be detected during Read0 element in March X. The first instruction in* `I2` *moves immediate number 0 to R1. The rest instructions in* `I2` *move the value in Rn-1 to Rn. The value in register file should be all 0 after* `I2`. *If there is only stuck-a-1 fault in register, it will be detected during Read0 element in March X. For the case both stuck-at-1 and stuck-at-0 happened and stuck-at-1 happened earlier than stuck- at-0, Write1 and Read 1 elements in March X algorithm can detect the fault.*

The software energy consumption in `I2` *is lower than in* `I1`. *The reason is that mov instructions with immediate addressing mode* `(I1)` *takes two CPU cycles, while mov instructions with register addressing mode* `(I2)` *only cost one CPU cycle. By this instruction selection,* n *CPU cycles are saved in* `I2` *while faults are properly detected even in the presence of faults in registers, hence, 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 testing such as ALU test and data bus test. The duplicated instruction sequences can be combined with the similar dynamic programming approach in [50]. This instruction combination will decrease the total test program length without harming fault coverage of each component.

4.1.2 Operand Selection

After instruction selection and combination, we consider the deterministic operands that should be applied to selected $I_{C;O}$ to test component operation O_C with high structural fault coverage and low energy consumption. The weights of the instruction operands and the Hamming distance between successive instructions' operands are of major concern in energy reduction. An operand set that can satisfy the test requirement of the operation O of component C is firstly identified. Then we select a deterministic operand with least weight and Hamming distance from the operand set. The selected operand is applied to $I_{C;O}$ for component inputs and the component outputs are propagated to the processor primary outputs.

The operand selection refers to choosing the minimum energy operand that preserves the given test goal for a given instruction. The implementation of operand selection in the ALU test routine is shown in Example 2.

Example 2: *Consider one of the operations for ALU testing* O_{ALU} = {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:                         I4:
mov 0xFFFF, Rn              mov 0x8000, Rn
mov 0xFFFF, Rm              mov Rn, Rm
add Rn, Rm                  add Rn, Rm
```

Both `I3` *and* `I4` *test the ALU operand add with carry by adding two operands that cause the carry bit. The carry bit will be checked afterwards.* `I3` *move two immediate numbers 0xFFFF to registers with register addressing and add them to cause one carry bit. The first instruction of* `I4` *moves immediate number 0x8000 to Rn. The second instruction* *moves the value in Rn to Rm with register addressing. The third instruction adds the two values and also cause one carry bit. As shown in Figure 12, instruction energy consumption is proportional to Hamming distance and weight of the operand. The value 0x8000 has less weight (only 1) than any other operand such as 0xFFFF (weight is 16) to test operation of the ALU. Although the Hamming distance of* `I3` *and* `I4` *are the same (both are 0),* `I4` *use register addressing mode which cost less energy than the immediate addressing mode used in* `I3`.

Considering both of the above instruction-level energy reduction methods and SBST proposed in 3.2, the pseudo code of our energy reduction SBST method is shown in algorithm 1. Steps 1-3 identify CPU component tests, as well as instruction sequences that test components fully. Step 4 uses the information extracted in the previous steps; the components that exist in a CPU core are sorted by test priorities. Steps 5-8 apply to each component test and use the greedy search to find instruction sequences of least software energy by instruction selection and combination with (step 6), followed by operand selection with least Hamming distance and weight (step 7).

```
1.  C = {$C_1$, $C_2$....$C_N$};            // set of CPU components
2.  $O_C$ = {$O_1$, $O_2$...$O_M$};           // set of ops for component C
3.  I(C, O) = {$I_1$, $I_2$....$I_P$};        // instructions causing C to perform O
4.  Sort C in decreasing test priority
5.  for (i=1, i<N+1, i++)           // for untested component $C_i$
      {
        for (j=1, j<M+1, j++)       // for operation $O_j$ from set $O_{C_i}$ ;
            {
6.    Find a least energy test sequence $I_k$ from I($C_i$ , $O_j$) by
      dynamic programming // instruction selection and combination;
7.    Select $I_k$ operands of least Hamming distance and weight;
8.    Apply $I_k$ to $C_i$ and propagate the result to outputs.
            }
      }
```

Algorithm 1: Low-Energy SBST Generation

4.2 Current Measurement for CPU Testing

The above two types of energy reduction methodologies are used for overall CPU core testing routines. The proposed current measurement architecture is used to measure the software energy consumption. Figure 14 and Figure 15 show the current measurement result for the original software energy consumption and the energy consumption after operand selection and instruction selection/combination, respectively. The energy consumption and the total number of CPU cycles before and after optimization are described in Table 6.

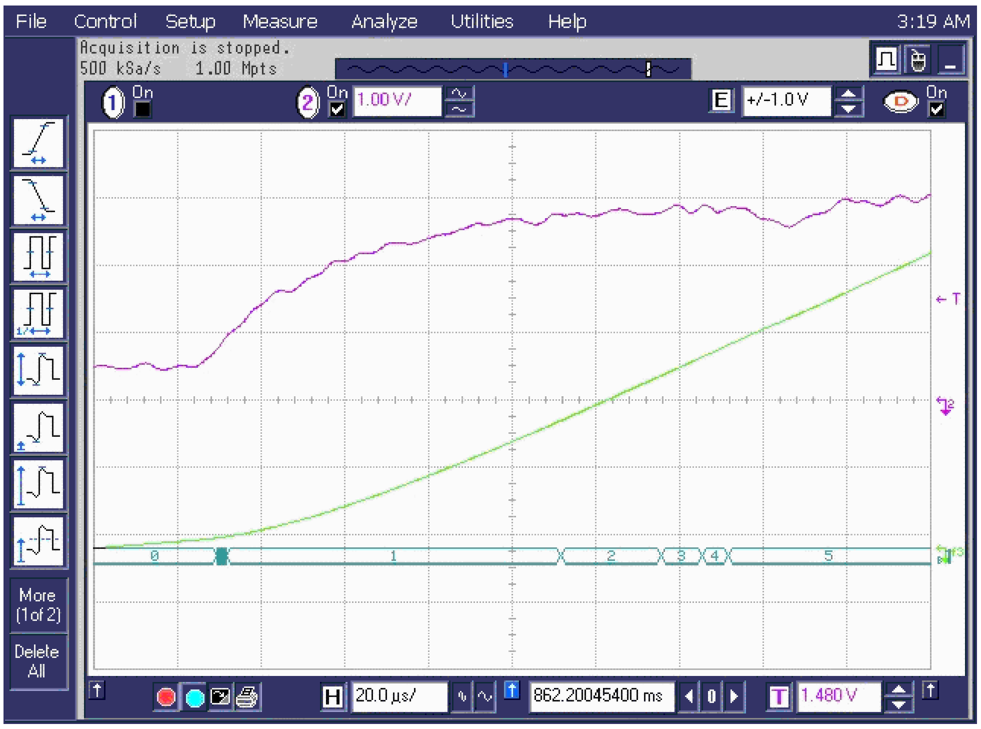


Figure 14 : Current Measurement of CPU Testing before Optimization

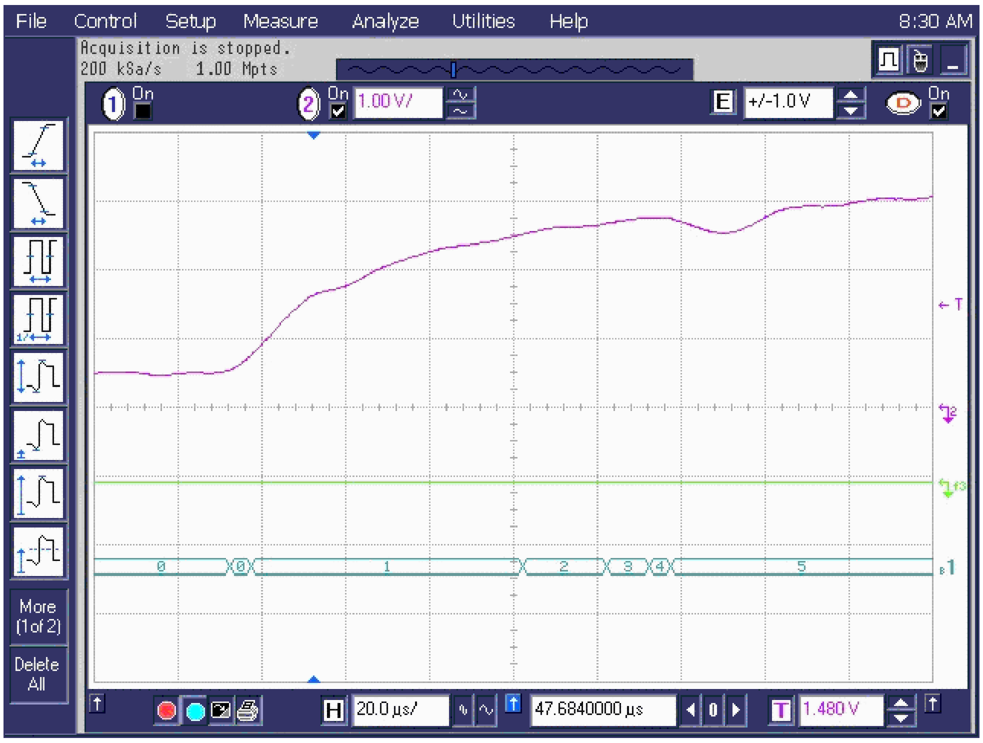


Figure 15 : Current Measurement of CPU Testing after Optimization

Table 6 : Energy Consumption of CPU Test Routines

Test item	Original test	Operand selection	Instruction selection
Register file test (µJ)	1.99	1.87	1.49
ALU test (combined with data bus testing) (µJ)	0.384	0.379	0.354
ALU status bus test (µJ)	0.184	0.181	0.17
Total energy (µJ)	2.558	2.43	2.014
Total CPU cycles	940	940	751

From these results we can see that after selection of the operand with the least Hamming distance and weight, although the total number of cycles has not changed, the energy is reduced by 5%. After selecting instructions with least CPU cycles and combining identical instructions amongst different algorithms, we can achieve a totally 21.2 % energy reduction and a 20.1 % time reduction for overall CPU core testing by using all instruction and combination techniques.

4.3 Efficiency of SBC Addressing in Memory Testing

A method that minimizes the Hamming distance between the consecutive addresses during March-type tests was introduced in [48]. The 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%. The total energy reduction claimed is between 18% and 77% for different sizes of standalone RAM memories.

Due to the lack of detailed energy models, the energy profiling capability is indispensable in devising energy-efficient memory SBST. Using measurements we establish that the SBC method might actually *increase* energy consumption. Figure 16 compares measured current for SBC and for that of the binary addressing used in on-chip FLASH testing on the 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 due to switching activity minimization on the memory bus requires the energy profiling to infer the energy increased by extra instructions.

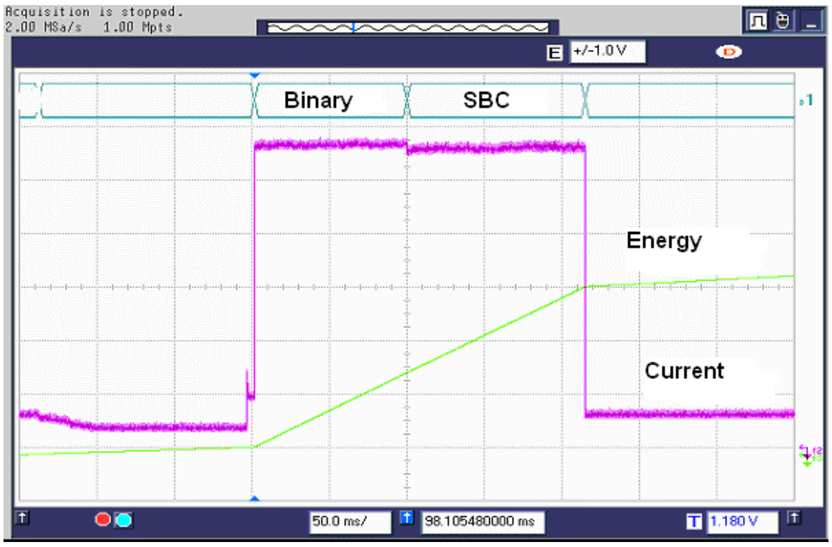


Figure 16 : Comparison of Binary and SBC FLASH tests

4.4 Time Interleaving of FLASH and other Tests

FLASH memory can perform only block or chip erase, with the erase operation being much slower than read or 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, the 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 17. The efficiency of time interleaving depends on the size of the FLASH, test code, timing and energy consumption of FLASH (erase/program/read) and other components. There is also a possible overhead in transferring the test code to RAM.

Figure 17 (a) shows a normal test routine sequence and the approximate average power used. By applying time interleaving, as seen in Figure 17 (b), the tests are rescheduled by interleaving the FLASH erase cycle and other component testing, causing a reduction in overall test time and energy consumption:

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

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 Figure 17.*

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

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

From the above equations, the amount of energy and time reduction due to interleaving increases both with the size of FLASH memory and with the proportional disparity between the FLASH and CPU speeds. Both of these are becoming more pronounced with advances in technology.

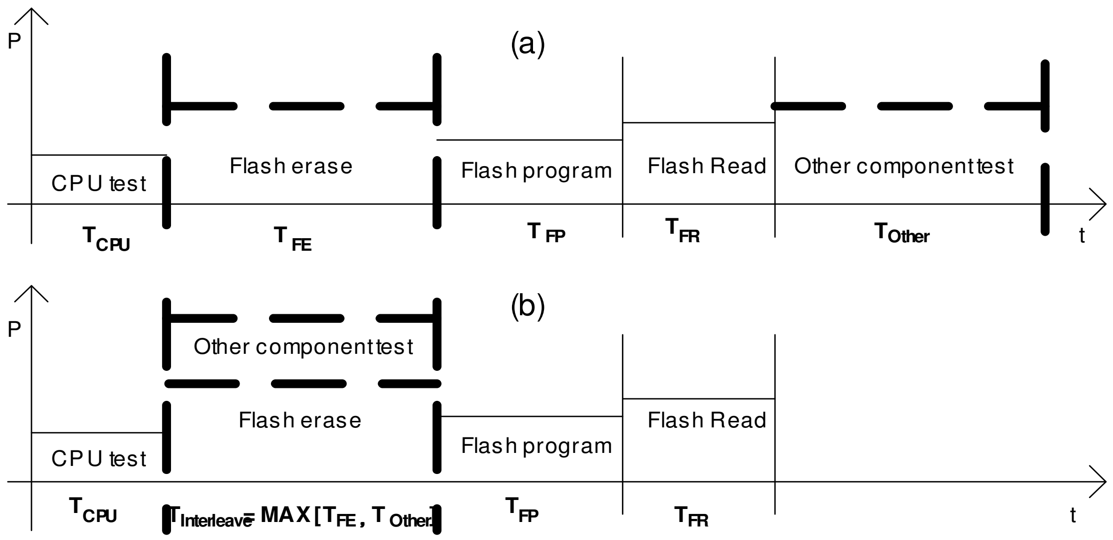


Figure 17 : The Concept of Time Interleaving

4.5 Current Measurement for Time Interleaving

We implemented the proposed time interleaving and present the results for two cases: time interleaving between FLASH and RAM testing, and time interleaving between FLASH and RF module testing. Table 7 gives the description of the different test routines in these two cases.

Table 7 : Test Routines (Modes) indicated in Figure 18-22

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)

4.5.1 Time-interleaved FLASH and RAM Testing

Figure 18 and Figure 19 show the current measurement result before and after time interleaving between FLASH erase and RAM testing within embedded memories in the MSP430 microcontroller (TI), respectively. 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. Since a write/read to RAM is much faster than to FLASH, to fulfill complete time interleaving, ten blocks of RAM are tested. Modes 5-7 are the first three elements of the March FT algorithm for FLASH testing and mode 8-9 are the main elements of the March X algorithm used for RAM testing.

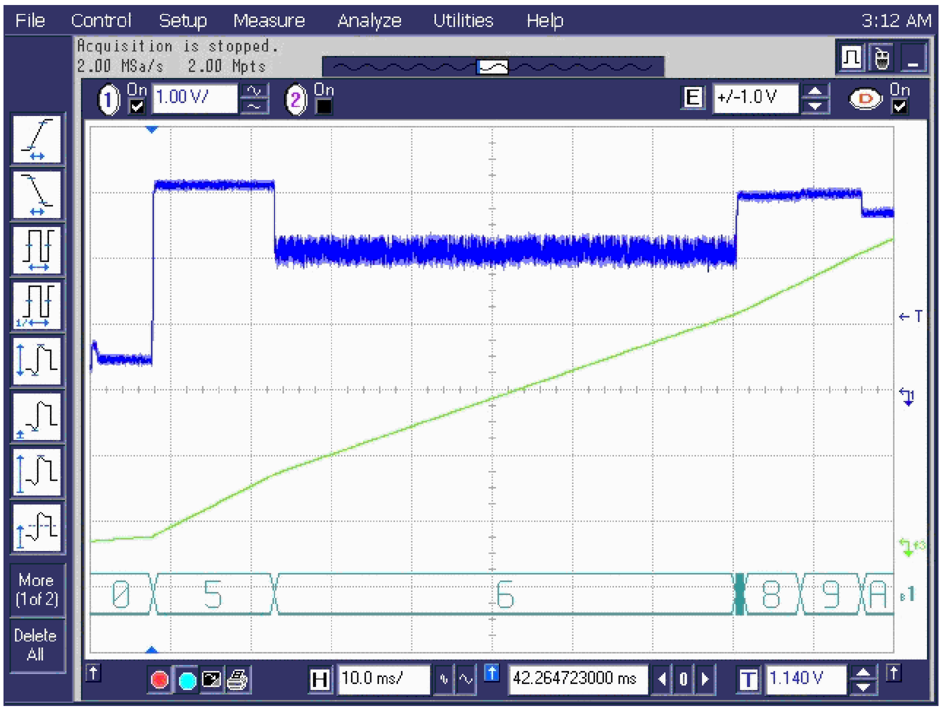


Figure 18 : FLASH and RAM Test Routines before Time Interleaving

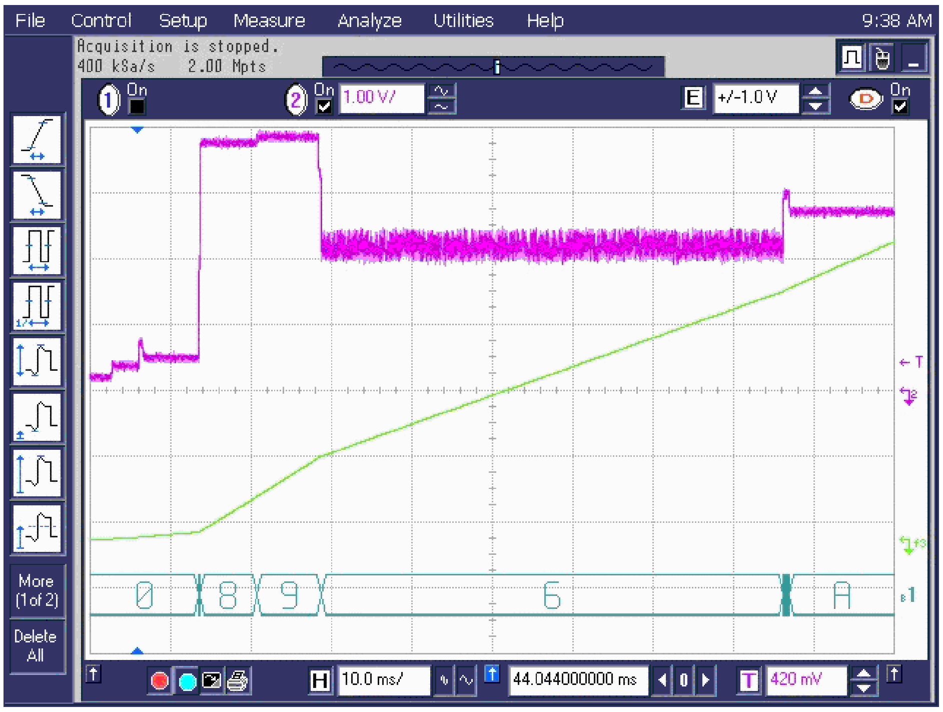
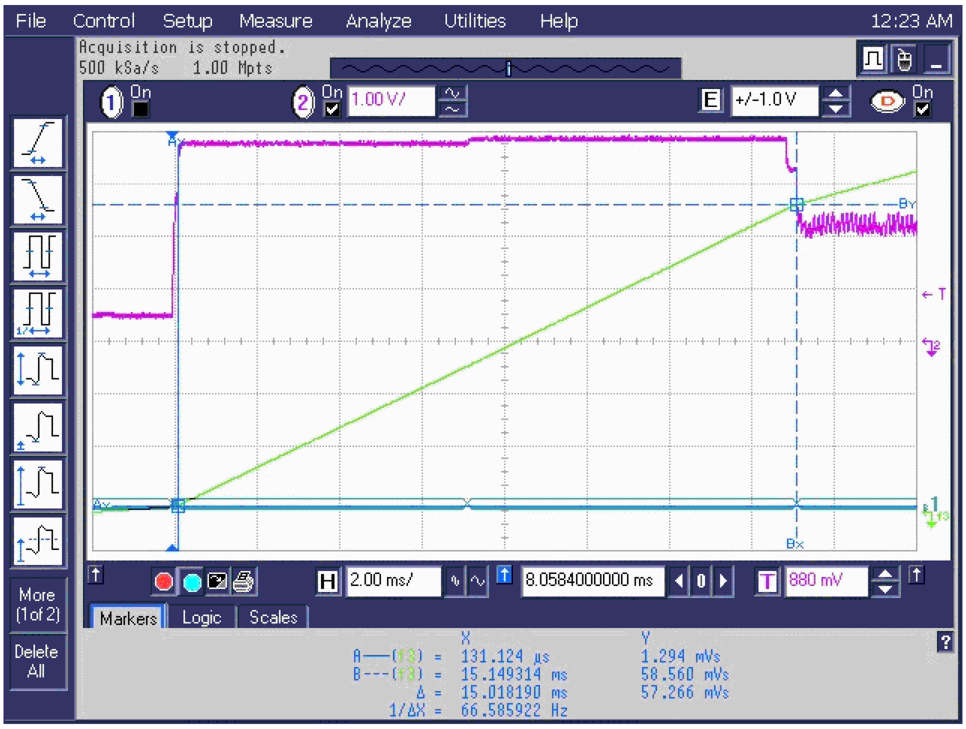


Figure 19 : FLASH and RAM Test Routines after Time Interleaving

Table 8 : Energy for FLASH and RAM Test

Test item (FLASH and RAM test)	Before Time interleaving	After Time interleaving
FLASH erase (μJ)	143.1	189.94
RAM W0 (μJ)	73.06	
RAM R0 (μJ)	78.54	
FLASH r1,w0,r0 (μJ)	413.16	421.77
FLASH r0 (μJ)	7.92	7.86
Energy total (μJ)	715.78	619.57
Time total (ms)	91.24	72.3

From the test result shown in Table 8, the CPU cycles used for FLASH erase and RAM W/R can be totally interleaved and the energy reduced from 294.7 μJ to 189.94 μJ. By time interleaving between the RAM test and FLASH test, we can get a **13.4%** energy reduction and a **20.7%** time reduction.



```
Acquisition is stopped.
500 kSa/s    1.00 Mpts
```

```
                    X                 Y
A——(f3) =   131.124 µs        1.294 mVs
B---(f3) =  15.149314 ms      58.560 mVs
       Δ =  15.018190 ms      57.266 mVs
    1/ΔX =  66.585922 Hz
```

Figure 20 : Energy and Time Measurement Method

Figure 20 shows our measurement method for energy and time for the interleaved sections. The top line is the instantaneous current measured by scope. The bottom line is the integral of the top line. Between the two markers we can read the time it cost and the integral of current (Y). The energy consumed by the code sections can be calculated by Y * 3.3 (power supply) * 1/1000 (current amplifying gain)

4.5.2 Time-interleaved FLASH and RF Testing

The time interleaving can also be used between FLASH testing and RF testing. Since RF transmission costs more energy, the current amplifying gain in this case is 100. A description of the mode is shown in Table 7. The time interleaving is applied for one block of FLASH erase and the sending of 4 RF packets, which was chosen because they were found to have comparable test time.

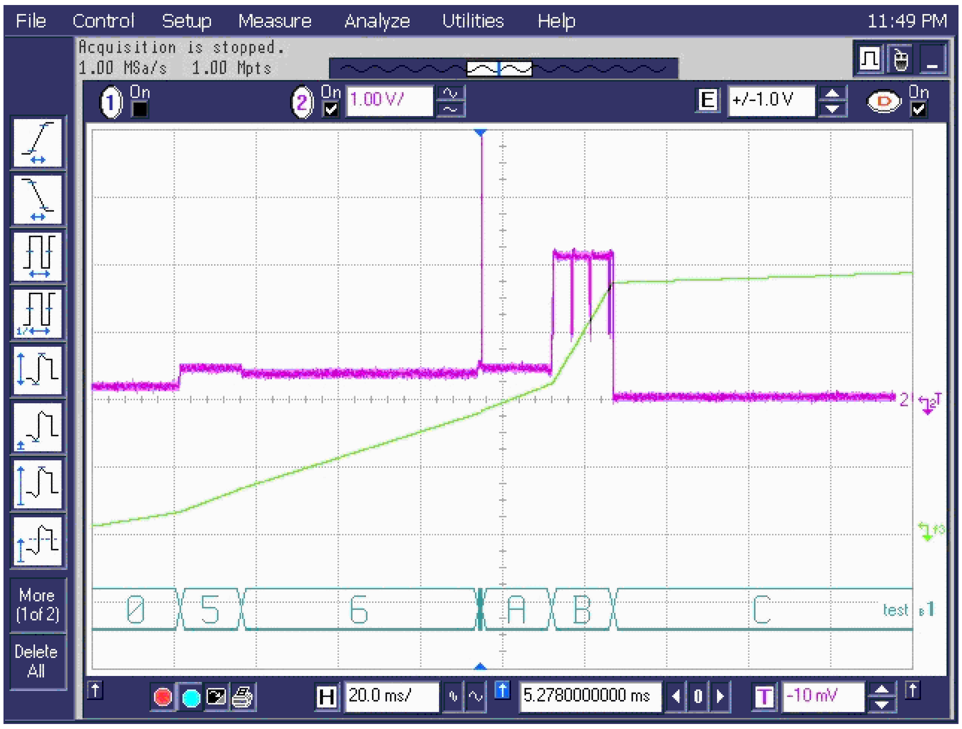


Figure 21 : FLASH and RF Packets Transmission before Time Interleaving

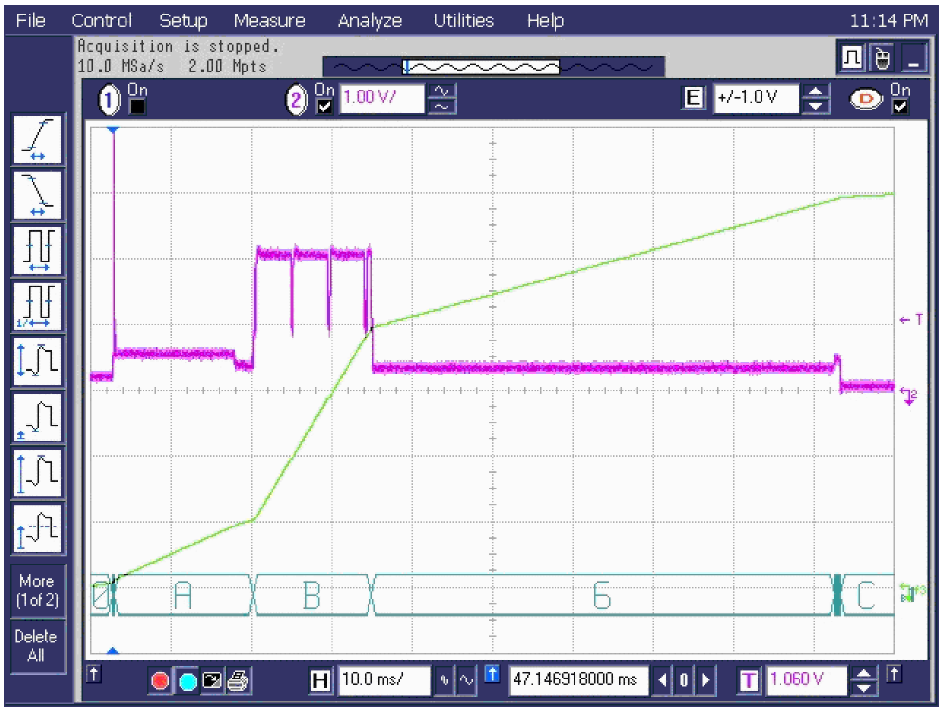


Figure 22 : FLASH and RF Packet Transmission after Time Interleaving

Table 9 : Energy Consumption for FLASH and RF Testing

Mode	Test item	Before Time interleaving	After Time interleaving
5	FLASH erase	231E-6 (J)	1.276E-3 (J)
A	RF initialization	280.83E-6 (J)	
B	RF packets sending	983E-6 (J)	
6	FLASH r1,w0,r0	723.03E-6 (J)	707.46E-6 (J)
7	FLASH r0	13.2E-6 (J)	12.87E-6 (J)
	Energy total	2.236E-3 (J)	1.996E-3 (J)
	Time total	105.2 ms	90.24ms

Table 9 shows the measurement result before and after time interleaving between FLASH erase and RF packets sending. A 10.7% energy reduction and a 14.2 % time reduction can be achieved according to the measurement result.

4.6 Energy Considerations for RF Testing

Popular wireless communication protocols have been developed to favor battery-powered node, such as 804.15.4. These nodes can require duty-cycling to reduce power consumption so that most of their operational life is spent in a sleep mode. However, each node shall periodically listen to the RF channel in order to determine whether a message is pending. This mechanism allows the application designer to decide on the trade-off between battery consumption and message latency. Listening to the RF channel continuously is an option to avoid message latency that will lead to higher power consumption. Table 10 and Figure 23 show the energy consumed by node under different wireless node operating modes.

Table 10: Description of Different Wireless Communication Modes

Mode	Operating mode description
1	Sending 5 packets with max power level
2	Sending 5 packets with medium power level
3	Sending 5 packets with min power level
4	Sleep mode, wait for periodical beacon packet
5	Always awake, monitoring RF channel continuously

Further, modern wireless protocols incorporate several energy reduction techniques, including the use of beacon signals. The testing of WSN nodes is periodically activated by the beacon signal, and the NUT is mostly in the sleep mode between beacons. Figure 23 shows current measurement under the different operating modes of a NUT. Modes 1-3 are sending test packets at different transmission power levels. Mode 4 is the sleep mode. We do not use the beacon scheme in Mode 5.

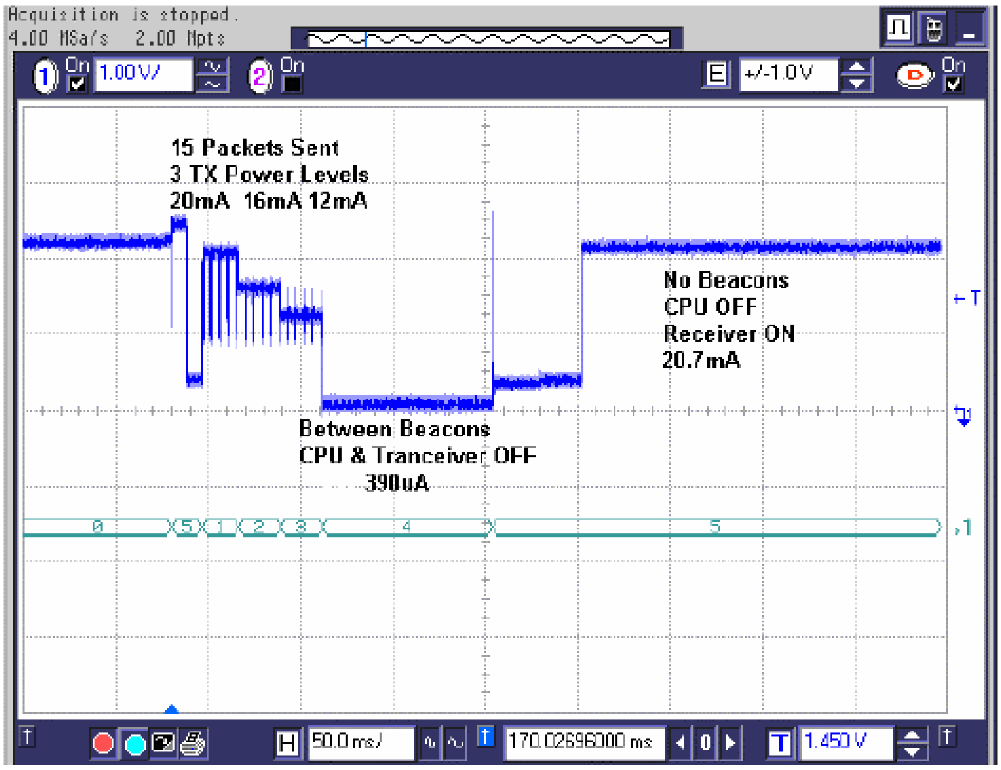


Figure 23: Current Consumption Profile of the Node Testing

From the result of Figure 23, it can be concluded that mode 5 is the worst operating mode among all the operating modes that have been tested, in terms of minimizing the energy consumption. Mode 4 cost the least amount of energy since the CPU and transceiver are set to be OFF between two beacons. Modes 1-3 show that energy consumption is decreased when we set the lower output power levels.

As mentioned in 3.4.2, we use a linear searching method to sweep transmitter output power from a low level to a high level for RF specifications testing. This searching algorithm yields a lower energy consumption than any other search algorithms because it begins from the lowest power levels. The least energy consumption with required PER is achieved for node testing using the linear searching method. Furthermore, we should also use the cycling beacon RF communication scheme to activate the node, since the beacon scheme guarantees that the node works in sleep mode for most of the time.

4.7 Summary

At the beginning of this chapter, based on the current measurement architecture, some new instruction-level software energy optimization methods are proposed for CPU core testing. These methods focus on operand selection with least Hamming distance and weight and on instruction selection and combination with least power consumption. From the current measurements, a 21.2% reduction in software energy consumption for CPU core testing is achieved by using the proposed optimization methods. The traditional SBC addressing method for energy reduction of memory testing has proved to be inefficient for testing on-chip memory according to the current measurements. Time interleaving of different test routines has proved to be an efficient system-level energy optimization method especially for advanced embedded system with large sizes of embedded FLASH memory. We take one segment (512 bytes) of FLASH testing and 10 segments (5kBytes) of RAM testing as an example to implement the time interleaving between FLASH testing code and RAM testing code. A 13.4% energy reduction and 20.5% time reduction are achieved by this interleaving according to current measurement results. The time interleaving is also implemented between FLASH testing and RF tests. A 10.7% energy reduction and 14.2 % time reduction are achieved by this interleaving according to current measurement results. Finally, we measure the software energy consumption under different wireless communication modes and conclude that the beacon transmission scheme is the optimal mode of operation with respect to software energy consumption.

Chapter 5

Structuring Measurements for Modeling and the Deployment of Industrial Wireless Networks

The true convenience of interconnecting devices without the use of wires has lead to the unprecedented success of wireless technologies in the computer and consumer electronics industry [52]. Various wireless networks are now beginning to appear in industrial settings. They promise to reduce the cost and save the time needed for the installation and maintenance of industrial control networks. The large number of cables normally required in such an environment can be substantially reduced, thus making plant setup and reconfiguration easier. For example, a typical commercial building can contain hundreds of sensors that are wired to central air conditioning and ventilation systems. Replacing wired units with WSN nodes offers more flexibility, and ultimately a better, energy efficient installation. Eliminating wiring is especially important in industrial environments where chemicals, vibrations, or moving parts can damage any cabling.

Wireless networking technology poses, however, many challenges [53][55], especially in guaranteeing the sufficient and reliable coverage during its deployment. Wireless networking devices are inherently power-limited, which limits the ability to combat communication channel errors. Even without power limitations, phenomena such as obstruction and multipath interference on the transmitted signal path make the link quality hard to predict and design for. The industrial setting, with numerous pieces of metal machinery, racks and moving parts is especially plagued by link obstruction and multipath interference. If a transmitter node (TX) is trying to connect to a receiver (RX) located in a typical industry environment full of metal surfaces (as shown in Figure 24), there will be many transmission paths, including a direct Line-of-sight (LOS) connection path and other multiple-reflection Non-line-of-sight (NLOS) paths. Since each path has different delay and attenuation, the received signal is badly affected by those destructive interferences.

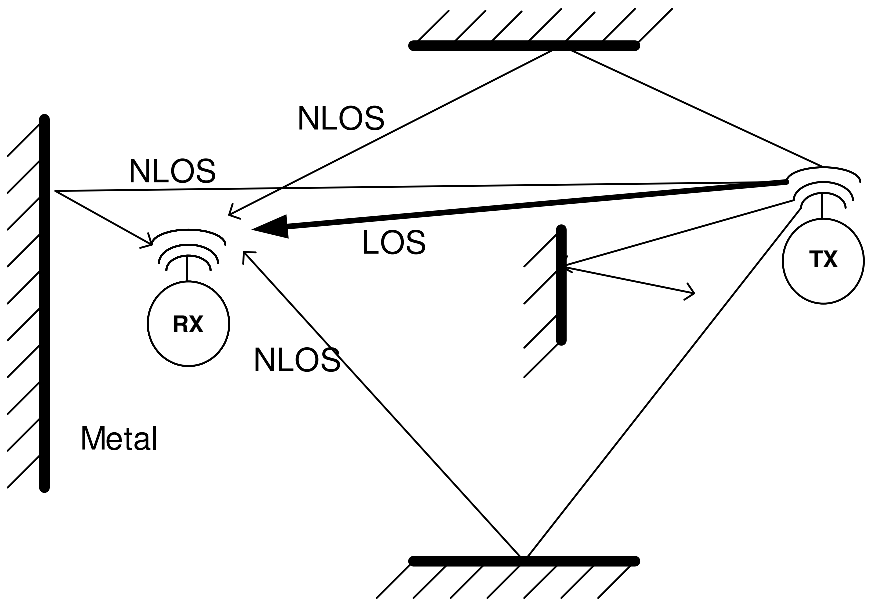


Figure 24: The Real Industry Environment.

The performance of deployed wireless networks greatly depends on the details of the underlying communication channel [42]. Hence, to evaluate performance of wireless networks, an accurate communication model is necessary. Until recently, two major approaches have been in widespread use in the sensor network community: unit disk modeling and empirical data traces [42]. The unit disc model states that communication between two wireless nodes is solely a function of the distance and that communication is conducted without any loss of packets if the nodes are closer than a specified communication range. However, the complete correlation between the properties of geometric space and the topology of the network has been refuted by numerous experiments in actual deployments [39]. At the other end of the spectrum are networks and communication patterns that are empirical traces of deployed systems. These networks are, of course, completely accurate samples of real life wireless communications. However, it is difficult and expensive to create a large number of physically large network instances that are properly characterized [42].

Recently, a statistical model of lossy links for wireless sensor network (WSN) was proposed [42] to produce network models of arbitrary sizes with realistic properties. This work provides a foundation for extracting the relationship between wireless node locations (distance) and communication properties (e.g. Reception Rate -RR) using non-parametric statistical techniques. The objective is to use a non-parametric method to obtain a Probability Density Function (PDF) that completely characterizes the relationship between the distance and RR. Based on the study of PDF about properties of individual and group links, an iterative improvement-based optimization procedure is used to generate network instances that are statistically similar to empirically observed network.

The IEEE 802.15.4 standard was finalized in October 2003 with the aim of creating a low cost, low power, two-way wireless communication solution that meets the requirements of sensors and control devices. In contrast to other wireless protocols such as IEEE 802.11, IEEE 802.15.4 has been specifically developed for use with applications in which a static network exists that has many infrequently used devices that transmit only small data packets. Such applications exactly match the needs of many industrial environments. The unique properties of wireless links in the 2.4-GHz range (commonly used in IEEE 802.15.4) are: radio waves can penetrate walls and are reflected by several materials. As a result, multiple copies of a signal may travel on several paths with different distances from transmitter to receiver. Errors occur not only due to noise, but also due to the multipath fading. In addition, distance-dependent path loss and co-/adjacent channel interference influence the channel. Hence, the wave propagation environment (number of propagation paths, their respective loss) and its time-varying nature (moving people, machines or wireless nodes) play a dominant role in constituting channel characteristics.

In this chapter, we present our methodology for conducting measurements for the purpose of assisting in their seamless deployment. We apply this methodology to the

IEEE 802.15.4 wireless protocol. The chosen measurement scenario shares some common characteristics of industry environments: many metal surfaces, moving parts, and machines switching on and off. A targeted set of measurements about the physical and communication properties of WSN are presented. The measurement results can be used to set up experimental models for WSN [42] or to evaluate the performance of wireless networks prior to their deployment to a particular site. Based on the measurement results, we provide the foundation for analyzing the influence of these features to the WSN performance and validate their suitability for the actual deployment.

5.1 Measurement Methodology

It is beneficial to parameterize the current wireless link models from “real data”, obtained from measurements, or to use the measurement results as a motivation for developing better models. For the deployment of wireless network, some network features measurements in real application environment can be used to evaluate the performance of targeted wireless networks and give a guideline for user to make the decision. The measurement setup is described next.

Table 11: Wireless Network Testing Features

Testing features (Mi)	Testing configuration (Ci)
Transmit Power level	8 output power levels ranging from 0dBm to -25dBm
Frequency	Ranges from 2.405GHz to 2.48GHz, 16 channels in total, monitors frequency interference in different channels
Packet size	20, 50 and 100 bytes per packet
Antenna polarization	0, 45, 90 degree between transmitter and receiver antenna
Antenna height	0, 50cm height to the ground
Asymmetry	Detect the difference of transmission direction of A->B and B->A
Temporal	Monitor the relationship of RR and distance during different time

To delve into the wireless communication characteristics, we develop a series of measurements to build the statistical relationships with respect to the features that impact network architecture and protocols in real networks. Our task is to analyze the relationship between two main properties of wireless network under some features about common transmitter, receiver and geometrical location. The details about the test features shown in Table 11 are explained as follows:

Transmit power levels:

One of the fundamental issues that arise naturally in sensor network is the coverage. In radio communications, coverage means the geographical area within which service from a radio communications facility can be received. Energy is another key concern with wireless networks. The proposed power level measurement will try to consider the coverage and power issues at the same time.

Frequency:

In the future, it will be standard for multiple wireless technologies to be used in a single environment. This is general not a problem unless the technologies are placed in the same frequency band. For the protocols at the same band, it is necessary to investigate the performance of coexisting networks and to find methods for reducing mutual disturbance between them. The associated interference between IEEE 802.15.4 and IEEE 802.11 is quantitatively assessed here.

Packet size:

The network performance measurements with different packet sizes are used to quantitively mention the influence of some protocol design to the wireless communication performance.

Antenna features:

As mentioned before, low power consumption is critical for the application of WSNs. The proper location of WSN node and the orientation with respect to the antenna directionality can help to reach better coverage, as well as reduce the power consumption of node.

Asymmetry and Time-variable characteristics:

The measurements presented here try to answer the following questions: Is there an asymmetry in WSN links? Does the temporal variable cause the change of wireless communication?

The above measurements are implemented by a pair of WSN nodes; one is a transmitter and the other is a receiver. The network measurement architecture is shown in Figure 25.

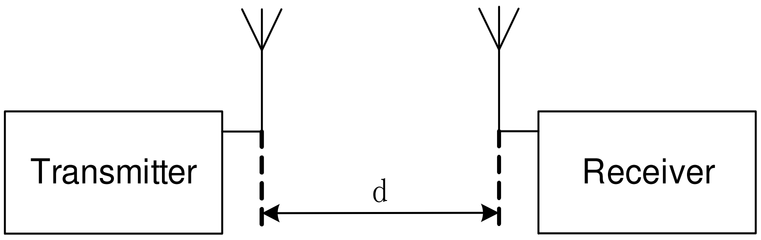


Figure 25: Wireless Network Measurement Architecture

The pseudo-code for transmitter and receiver are shown as follows:

Transmitter:

```
(For each test feature Mi) {
   Set the default configuration of TX node;
      (For each test configuration Ci){
    Send the new configuration to RX node;
    Change the TX node with new config;
           If (TX and RX node are working in
              new configuration)
           {Sending packets ;}
    Receive test result ;}
 Set back to the default config ;}
```

Receiver:

```
(For each test features Mi) {
   Set the default configuration of RX node;
 (For each test configuration Ci) {
          Receive new configuration from TX node;
          Change the RX node with new config;
          Counting the packets received;
    Send the test result to TX node ;}
 Set back to the default configuration ;}
```

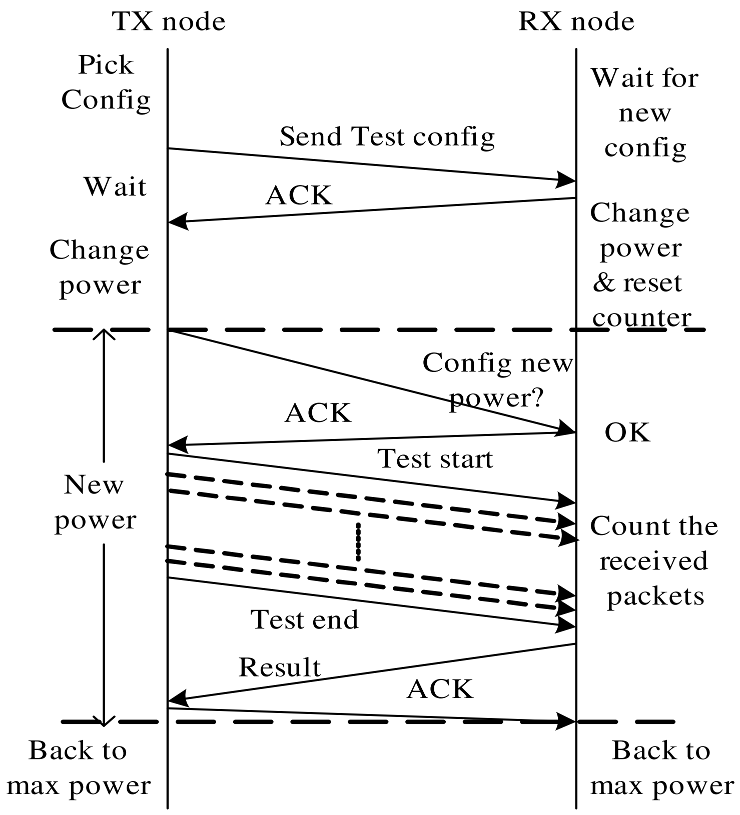


Figure 26: Wireless Network Testing Sequence Chart

Figure 26 shows the sequence chart of the code to test the relationship among RR, distance and output power levels. The testing code for other features has a similar program sequence.

5.2 Understanding Measurement Results

Our testing is implemented in two kinds of environments: the laboratory (indoors) and the campus (outdoors). The typical indoors environment includes the furniture (mental or wood), walls, electronic equipments, e.g. printers, microwave oven. The WSN node is built with low power microcontroller MSP430 from TI; Zigbee compliant RF transceiver CC2420 from Chipcon [44] and our own printed antenna. A pair of nodes with the same hardware is placed at the increasing distances (say, 5m, 10m, 15m, and 20m) with or without line-of-sight between them. The outdoor environment includes trees and buildings. The pair of nodes are placed at the distances (such as 10m, 20m, 40m and 60m) and at different height, such as near the ground or elevated off the ground. The indoors and outdoors testing environments are zones covered by the McGill wireless signal (802.11.b). At each test position, 10,000 test packets are transmitted for packet RR testing. In summary, the data set used in our testing consisted of packet delivery data for more than 2 million packets in experiments performed in 2 different environments, 8 different output power settings, 3 different working channels, 3 different packet sizes, 3 kinds of antenna polarization, 2 different antenna heights, two transmission directions for asymmetry and 10 different time points.

5.2.1 Power levels

Power level is an important characteristic of a wireless network node for power optimisation techniques. Here, the measurements are used to consider the tradeoffs between the coverage and energy consumption. For the employed Chipcon CC2420 RF transceiver [44], there are 8 programmable output power levels in total. Figure 27 shows the relationship between output power and current consumption under different power levels. We test the relationship between RR and distance under each output power level.

Table 12: Output power under different power levels

Power level	Output Power (dBm)	Current Consumption (mA)
8	0	17.4
7	-1	16.5
6	-3	15.2
5	-5	13.9
4	-7	12.5
3	-10	11.2
2	-15	9.9
1	-25	8.5

Figure 27 shows the 3-D graph of the measured relationship between distances, power level and RR in outdoor environment. From this graph we can see that for a fixed power level, RR decreases as the distance is increased. For a fixed distance, RR decreases as the power level is decreased. At the highest output power level, the communication range can reach to 60m with the required packet RR. This testing can be used to optimize the power consumption of the wireless node. With compliance to the required RR and distance, the output power of the CC2420 should be set as low as possible.

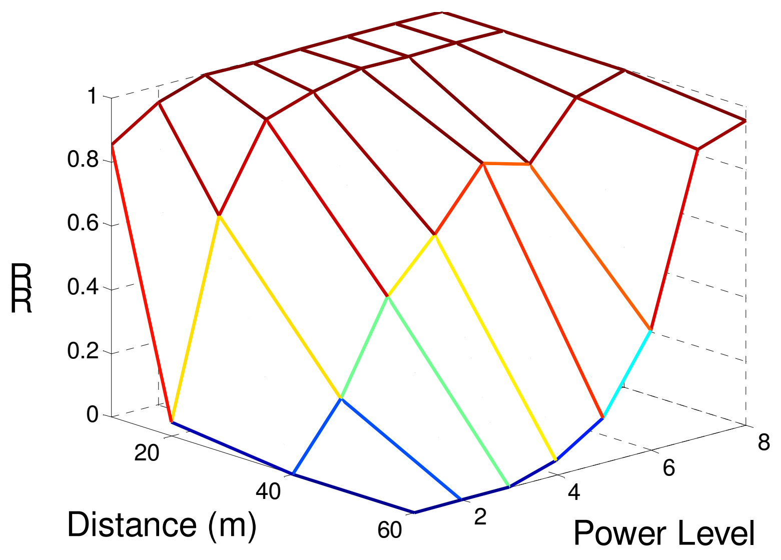


Figure 27: The Relationship between RR and Distance under Different Power Levels.

5.2.2 Asymmetry

Asymmetry in communication refers to the difference in RR of packets communicated strictly between two nodes. Two cases are possible for nodes A and B- first, the transmitter is A and the receiver is B; second, the transmitter is B and receiver is A. When the difference is beyond 50%, asymmetry is considered to be happening [42]. Very often, it is assumed that RR is the same in both directions. We design the tests to capture whether there is an asymmetry in the RR (as a function of the node distance).

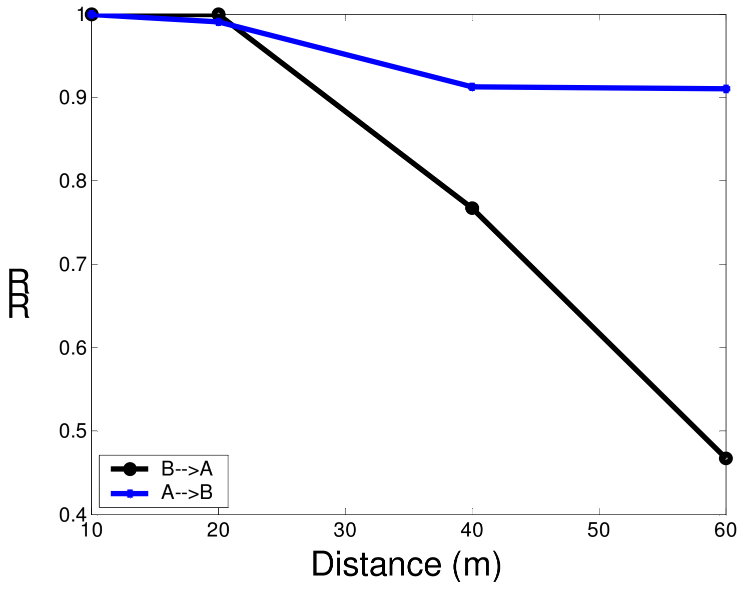


Figure 28: Dependency on Asymmetric RR to Distance

Figure 28 shows the dependency of asymmetric RR as a function of distance in outdoors environment. From the result shown in Figure 28, we can see there is big difference (44%) when the distance is very long (60m). By the definition of asymmetry (50% difference), it may not be consider as asymmetry. The possible reason for the big difference could be the minor circuit differences between A and B.

5.2.3 Temporal Variability

The goal of this measurement is trying to find the influence of the measurement time to the communication link performance (RR). Figure 29 shows the temporal variability of the relationship between RR and distance in indoor environment. The test is taken from 11AM to 9PM at two hours intervals. There is no obvious trend shown in the graph. There is only a minor difference (all from 97% to 100%) in the RR. The possible reason is the interference (e.g. machinery, microwave owens, etc.) applied randomly over time. We observe that time is not an important factor influencing the relationship between RR and distance.

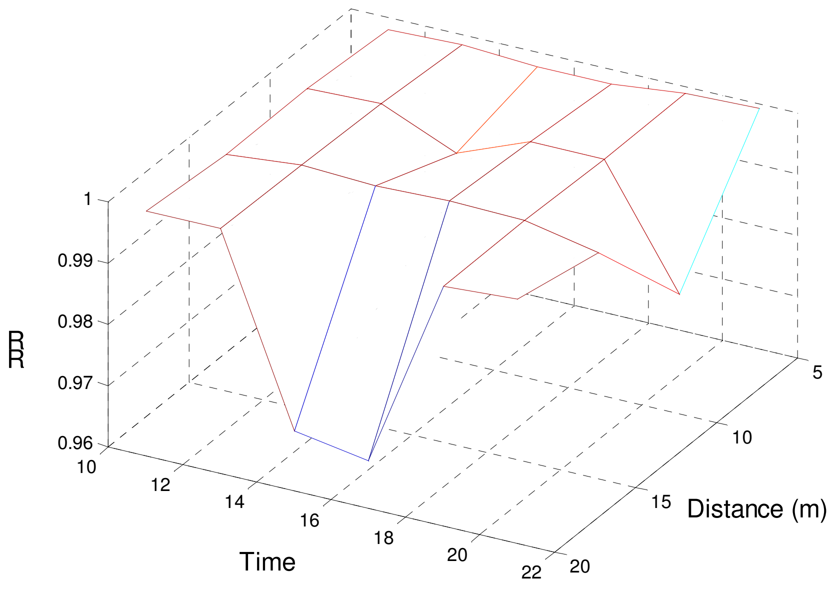


Figure 29: The Relationship of Distance and RR at Different Times

5.2.4 Interference from other wireless networks

Multiple wireless standards can use the same frequency band. The 2.4GHz ISM band is used for both IEEE 802.15.4 and IEEE 802.11 standards. The next measurement is designed to give the quantitive assessment about the interference between different wireless networks.

For the 802.15.4 networks [45], there are totally 16 channels (channel 11 to channel 26)and they are located within the range from the 2.405GHz to 2.48GHz at an interval of 5MHz. The measurement tries to capture the influence of different frequency to communication performance. Since the printed antenna is narrowband and designed to tune to Channel 11 (2.405GHz), channel 11 is expect to be the strongest channel with the highest RR compare to other channels. Figure 30 shows the relationship between RR and distance for three different channels (channel 11, 26 and 17) indoors.

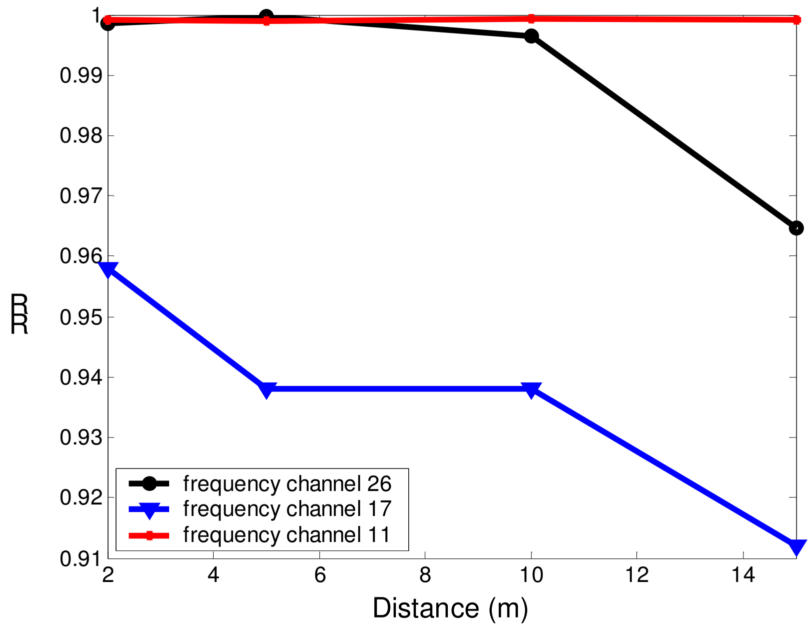


Figure 30: The Relationship between RR and Distance over Three Different Channels.

From the measurement result, we can see the RR of channel 17 is much lower than channel 11 and channel 26. The RR of channel 11 is better than channel 26. The reasons for these observations are explained as following:

1. Using a spectrum analyzer, we can see that there is a wide range of interference within the frequency band ranges from 2.43GHz to 2.45GHz. The strongest interference is caused by the 802.11.b wireless internet access (both indoor and outdoor environments).

2. The printed antenna is tuned to channel 11; therefore the antenna performance is best for this channel. That explains that the RR in channel 11 is higher than RR in channel 26 at each distance, even though there is no wireless interference in both channel 26 and channel 11.

5.2.5 Packet size

This measurement is used to detect the dependency of the transmission packet size to the RR and distance. Figure 31 shows the relationship between RR and distance for three packet sizes (20 bytes, 50 bytes and 100 bytes) in an indoor environment.

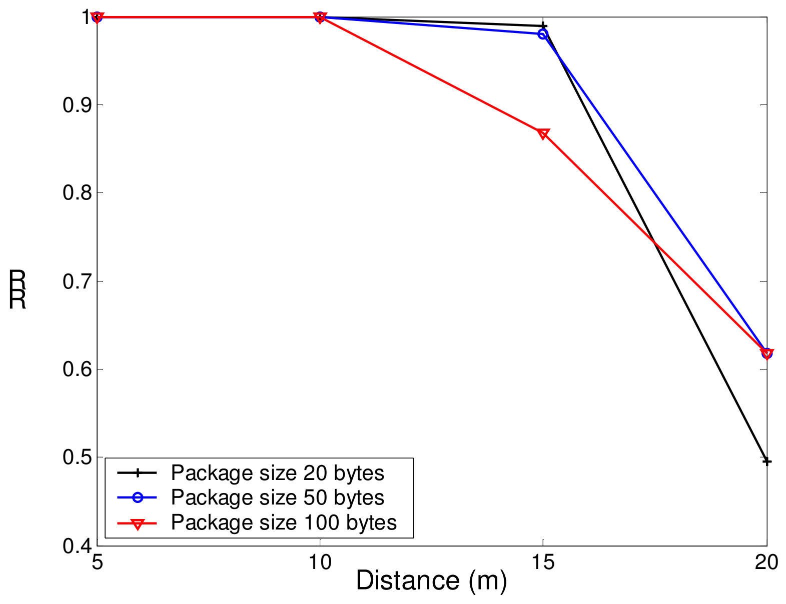


Figure 31: The Relationship between RR and Distance for Different Packet Sizes

From Figure 31, we can see that when the distance is short, the minimum packets size (20 bytes) corresponds to the best RR. When the distance is long, that tendency is not present. From the test result, there is no obvious regularity for different packet sizes. We can conclude that packet size is not one of the important factors affecting communication properties.

5.2.6 Antenna Features

Antenna is a critical component of wireless node and its design plays an important role for the whole wireless network performance. Here, we present the measurement for two kinds of features of the antenna. One is the antenna polarization and the other is the antenna height above the ground. The measurement results can provide a guideline for refining the antenna design.

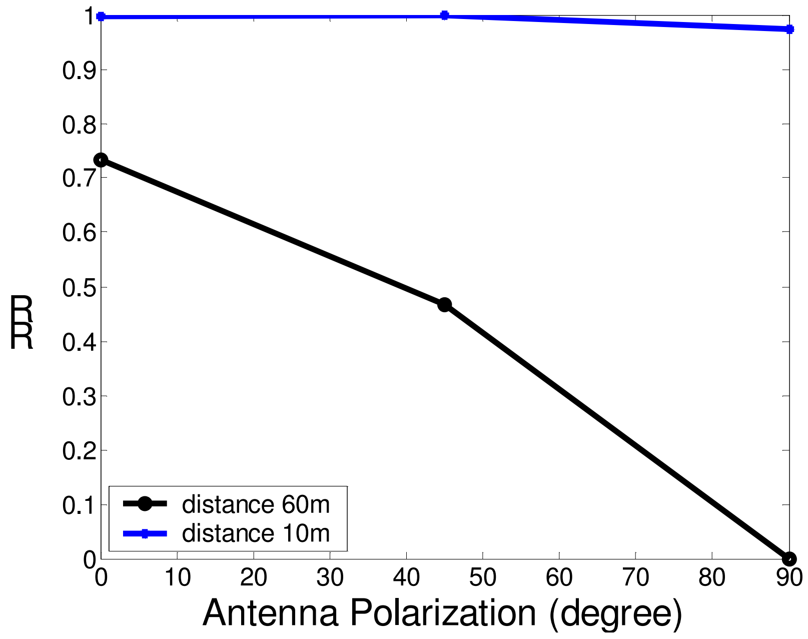


Figure 32: Dependency of Antenna Polarization to RR and Distance

Figure 32 shows the dependency of RR and antenna polarization at two distances (10m and 60m) in outdoor environment. There are 3 antenna polarization tested in Figure 32: 0 degree (the antennas of transmitter and receiver are parallel), 45 degree (45 degree angle between antennas of transmitter and receiver) and 90 degree (the antennas of transmitter and receiver are perpendicular). From the results shown in Figure 32, the RR is highest when the antennas are parallel (0 degree) and the lowest RR happened when the antennas are vertical (90 degree). This result provides a guideline for the node location of wireless network. The antennas of the nodes in the real wireless network should be a parallel set. Also we can see from Figure 32, the effect of antenna polarization at short distance (10m) is not as strong as at long distance (60m). The possible reason is that when distance is long, the transmission signal is weak and easily affected by antenna polarization.

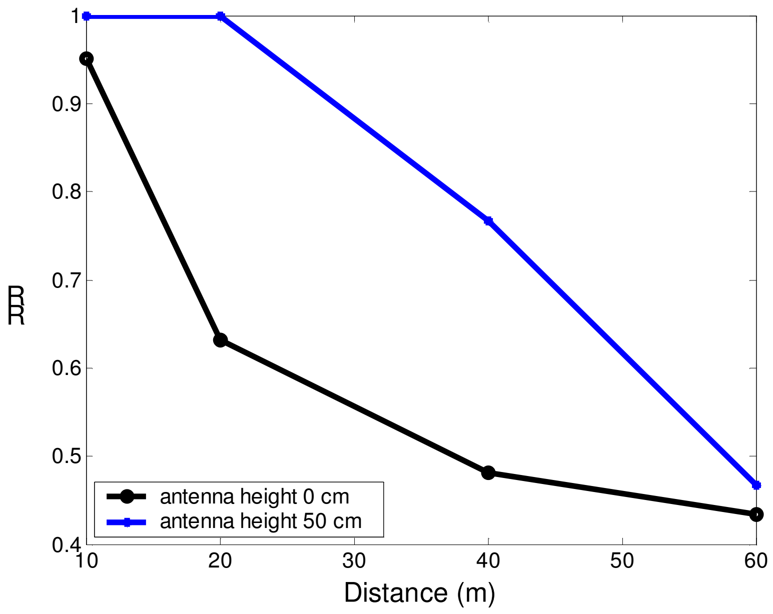


Figure 33: The Relationship between RR and Distance from the Antenna to the Ground.

Figure 33 shows the relationship between the RR and distance at two antenna heights in outdoor environment. We set two test conditions: putting the wireless node on ground and at a height of 50cm to ground. Based on the testing presented above, we can see that the antenna design and its placement are important factors for the system performance.

5.3 Facilitating Deployment and Model Building

Parts of our measurements results such as the relationship between distance and RR, asymmetry and temporal variation can be used to help building statistical models for WSN [42]. The collected data can be used as original data to calculate the probability density function (PDF) that establish a complete characterization of the relationship among network features. The PDF provides the likelihood that any particular value of one feature is associated with a given value of another feature. Based on the PDF of realistic network features, a series of wireless network generators are developed to produce networks of an arbitrary size. The generated instances of the network are statistically similar to the empirically observed networks.

Coverage is a key parameter to evaluate the wireless networks. The coverage of IEEE 802.15.4 can be easily obtained from our power levels measurement. Channels measurement shows the interference from coexisted wireless networks in the real industrial application. Both antenna polarization and antenna height shows that proper node location will increase the coverage as well as decrease the power consumption. Power and coverage are all key concerns to the deployment of wireless networks.

5.4 Summary

In this chapter, we have developed a methodology for measuring a set of network features for characterizing links in wireless network communication. Our measurements can help building communication link models for an arbitrary network that is statistically similar to observed networks. These measurements also greatly impact the power management techniques and WSN node location and configuration. For example, the antenna polarization and height should be considered during node placement. With required packet RR and distance, the output power levels can be configured as low as possible to reduce the node energy consumption with required coverage. The insight gained while building these relationships gives a guideline for developers of protocols, localized algorithms and antenna design for wireless networks and the users of wireless products.

Chapter 6

Conclusions and Future Work

In this thesis, we detail out a comprehensive system-level low-power SBST method tuned for wireless nodes. Individual testing of the main node components, as well as the combined testing across component boundaries are all considered for power optimization. The flexible low-cost and low-power 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 a dynamic programming approach. The CPU testing energy reduction of 21.2% is observed by current measurement on a prototype node. March FT algorithms used to test the embedded FLASH are similarly passed through the instruction and operand selection algorithm, as well as the address bus power optimization method. Time interleaving of the embedded FLASH test and other components' test routines is a major system-level technique used to reduce the energy consumption as well as the test time. The new scheme for characterization and test of the complete RF module is devised. 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 modern protocols. In the last chapter, we provided a significant amount of data from wireless network performance testing that can be efficiently utilized to build statistical models for protocol, algorithm and hardware design of WSN.

In chapter 5, we use only two WSN nodes to measure the relationship between the distance and the reception rate under different configurations (individual link properties).

The measurement results can be affected by the specific hardware features of the two nodes. Some network performance such as asymmetry may not be detected because of the shortage of collected data; thus, to get better statistical measurement, a real WSN with more nodes (around 10 nodes) are necessary to delve the group link properties. Group link properties are joint properties of subsets of links that are related to each other in a particular way. They include properties of the links that originate from the same transmitter or received by the same receiver, processed by the same radio, or communicated by nodes that are geometrically close. These properties answer some fundamental questions about reasons for particular behavior of communication patterns. For example, these questions include the hypothesis that the performance of a particular node as a transmitter mainly depends on the quality of its radio or its geometric position.

During the SBST of a RF module, each WSN node can work as a NUT or an interference node. At the normal operation of WSN, the test initialization can be broadcast (or multicast to selected sensor areas) by a TMN using any available broadcast/multicast mechanism in WSNs; then, the SBST of each node can be parallelized. Therefore, a TMN need a test management mechanism to organize the SBST of each node and to choose the interference source according to the proper distance from the interference node to the NUT.

Reference:

[1] N.Y. MANHASSET, “Wireless sensor network use to grow, says study”, EE Times, Oct. 2005.

[2] M.W. Chiang, Z. Zilic, K. Radecka and J. Chenard, “Architectures of increased availability wireless sensor network nodes”, *Proceedings of International Test Conference 2004*, pp. 1232-1241.

[3] S. Hellebrand and H.J. Wunderlich, “Mixed-mode BIST using embedded processors”, *in Proc. Int. Test Conf.,* Oct. 1996, pp. 195–204.

[4] F. Brglez, C. Gloster and G. Kedem, “Built-in self-test with weighted random-pattern hardware,” *in Proc. IEEE Int. Conf. Computer Design: VLSI in Computers and Processors,* Sep. 1990, pp. 161–167.

[5] S. Cataldo, S. Chiusano, P. Prinetto and H.J. Wunderlich, “Optimal hardware pattern generation for functional BIST”, *in Proc. Meetings Design, Automation and Test*, Mar. 2000, pp. 292–297.

[6] S. Manich, A. Gabarro, M. Lopez, J. Figueras, P. Girard, L. Guiller, C. Landrault, S. Pravossoudovitch, P. Teixeira and M. Santos, “Low power BIST by filtering nondetecting vectors”, *J. Electron. Test.: Theory Application*, June 2000, pp. 193–202.

[7] J. Shen and J. Abraham, “Native Mode Functional Test Generation for Microprocessors with Applications to Self-Test and Design Validation,” *Proc. Int’l Test Conf.,* 1998, pp. 990-999.

[8] K. Batcher and C. Papachristou, “Instruction Randomization Self Test for Processor Cores”, *Proc. VLSI Test Symp.,* 1999, pp. 34-40.

[9] L. Chen and S. Dey, “Software-Based Self-Testing Methodology for Processor Cores”, *IEEE Trans. Compter-Aided Design of Integrated Circuits and Systems*, Mar. 2001, pp. 369-380.

[10] N. Kranitis, A. Paschalis, D. Gizopoulos and Y. Zorian, " Effective software self-test methodology for processor cores" *Design, Automation and Test in Europe Conference and Exhibition,* March 2002, pp.592 – 597.

[11] N. Kranitis, A. Paschalis, D. Gizopoulos and G. Xenoulis, "Software-Based Self-Testing of Embedded Processors", *IEEE Transactions on Computers*, April 2005, pp.461 – 475.

[12] A. Krstic, Wei-Cheng Lai, Kwang-Ting Cheng, L. Chen and S. Dey, "Embedded software-based self-test for programmable core-based designs*", IEEE Design & Test of Computers*, July-Aug. 2002, pp.18-27.

[13] S.M. Thatte and J.A. Abraham, "Test Generation for Microprocessors," *IEEE Transactions on Computer,* 1980, pp. 429-441.

[14] D. B rahme and J.A. Abraham, "Functional Testing of Microprocessors", *IEEE Transaction on Computers,* 1984, pp. 475 – 485.

[15] A.J.van de Goor and Th.J.W. Verhallen, "Functional testing of current microprocessors (applied to the Intel i860)", *Proceedings of International Test Conference,* 1992, pp. 684-695.

[16] B.S. Joshi and S.H. Hosseini, "Efficient algorithms for Microprocessor Testing" *Proceedings of Annual Reliability and Maintainability symposium*, 1998, pp. 100-104.

[17] K. Jayaraman, V.M. Vedula, and J.A. Abraham, "Native mode functional self-test generation for Systems-on-Chip", *International Symposium on Quality Electronic Design,* March 2002, pp. 280 – 285.

[18] R. Kannah and C.P. Ravikumar, "Functional testing of microprocessors with graded fault coverage", *Proceedings of the Ninth Asian Test Symposium,* Dec. 2000, pp. 204-208.

[19] C. Bellon, R. Velazco and H. Ziade, "Analysis of Experimental Results on Functional Testing and Diagnosis of Complex Circuits*," In Proc. Intl. Test Conf.*, 1988, pp. 64-72.

[20] Alfred Crouch, *Design for Test for Digital IC's and Embedded Core Systems,* Prentice Hall, 1999

[21] P. Pavan, R. Bez, P. Olivo and E. Zanoni, "FLASH memory cells—an overview", *Proc. of the IEEE*, Aug. 1997, pp. 1248– 1271.

[22] S. Aritome, R. Shirota, G. Hemink, T. Endoh and F. Mausouka, “Reliability issues of FLASH memory cells”, *Proc. of the IEEE*, May 1993, pp. 776–787.

[23] A. J. van de Goor, *Testing Semiconductor Memories: Theoryand Practice*, John Wiley & Sons, Chichester, England, 1991.

[24] M.G. Mohammad, K.K. Saluja and A. Yap, “Testing FLASH memories”, in *Proc. 13th Int. Conf. VLSI Design*, Jan. 2000, pp. 406–411.

[25] Jen-Chieh Yeh, Chi-Feng Wu, Kuo-Liang Cheng and Yung-Fa Chou, “FLASH memory built-in self-test using March-like algorithms”, *Proc. IEEE Intl. Workshop EDTA*, 2002, pp. 137-141.

[26] IEEE 1005 Standard Definitions and Characterization of Floating Gate Semiconductor Arrays, *IEEE Standards Department*, Piscataway, 1999.

[27] M. Jarwala, D. Le and M. S. Heutmaker, “End-to-end test strategy for Wireless Systems,” *Intl. Test Conference*, 1995, pp. 940-946.

[28] D. Lupea, U. Pursche, H.J. Jentschel, “RF-BIST: Loopback Spectral Analysis”, *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, 2003, pp. 478-483.

[29] B.R. Veillette and G.W. Roberts, “A built-in self-test strategy for wireless communication systems,” *Proc. Int’l Test Conf.*, 1995, pp. 930-939.

[30] A. Halder, S. Bhattacharya, G. Srinivasan and A. Chatterjee, “A System-Level Alternate Test Approach for Specification Test of RF Transceivers in Loopback Mode” *Proc. of the18th International Conference on VLSI Design*, Jan. 2005, pp. 289-294.

[31] J. Dabrowski, “BiST model for IC RF-transceiver front-end”, *Proc. of IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems,* Nov. 2003, pp. 295 – 302.

[32] V. Tiwari, S. Malik and A. Wolfe, “Power Analysis Embedded Software: A First Step Towards Software Power Minimization,” *IEEE Trans. VLSI Systems*, Dec. 1994, pp.437-445.

[33] V. Tiwari, S. Malik, A. Wolfe and M.T.-C. Lee, “Instruction Level Power Analysis and Optimization Software,” *Proc. of ninth Int. Conf. on VLSI Design,* Jan. 1996, pp. 326-328.

[34] M.T.-C. Lee, V. Tiwari, S. Malik and M. Fujita, "Power Analysis and Minimization Techniques for Embedded DSP Software*," IEEE Trans. on VLSI Systems,* Mar. 1997, pp. 123- 135.

[35] S. Nikolaidis and T. Laopoulos, "Instruction-level power consumption estimation embedded processors low-power applications" *Proc. of International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications,* July 2001, pp. 139 – 142.

[36] S. Steinke, M. Knauer, L. Wehmeyer and P. Marwedel, "An Accurate and Fine Grain Instruction-Level Energy Model supporting Software Optimizations," *in Proc. of Int. Workshop on Power and Timing Modeling, Optimization and Simulation*, 2001

[37] J.T. Russell and M.R. Jacome, "Software Power Estimation and Optimization for High Performance, 32-bit Embedded Processors," *in Proc. of Int. Conf. On Computer Design,* Oct. 1998, pp. 328-333.

[38] N. Chang, K.H. Kim and H.G. Lee, "Cycle-accurate energy consumption measurement and analysis: Case study of ARM7TDMI", *In Proceedings of the International Symposium on Low Power Electronics and Design*, July 2000, pp. 185-190.

[39] A. Cerpa, N. Busek and D. Estrin, "SCALE: A tool for simple connectivity assessment in lossy environments", Technical Report 0021, Center for Embedded Networked Sensing, UCLA, Sep 2003.

[40] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin and S. Wicker, "Complex behavior at scale: An experimental study of lowpower wireless sensor networks," Technical Report 02-0013, Center for Embedded Networked Sensing, UCLA and IRL, UCB, February 2002.

[41] Y. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," *in Proceedings of ACM Sensys,* Nov. 2003, pp. 1–13.

[42] A. Cerpa, J.L. Wong, L. Kuang, M. Potkonjak and D. Estrin, "Statistical Model of Lossy Links in Wireless Sensor Networks", *in Proc. of Fourth International Symposium on Information Processing in Sensor Networks*, April 2005, pp.81 – 88.

[43] Texas Instruments, MSP430x1xx Family User's Guide, 2004.

[44] Chipcon AS SmartRF CC2420 preliminary datasheet (rev 1.2) 2004.

[45] IEEE std. 802.15.4/D18 – 2003: Wireless Medium Access Control (MAC) And Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs).

[46] W. L. Stutzman and G. A. Thiele, *Antenna Theory and Design,* Wiley, second edition, 1997.

[47] Burr-Brown, INA145 datasheet, 2000.

[48] H. Cheung and S.K. Gupta, "A BIST methodology for comprehensive testing of RAM with reduced heat dissipation", *in Proc. of International Test Conference*, Oct. 1996, pp. 386 – 395.

[49] J. Chenard, C.Y. Chu, Z. Zilic and M. Popovic, "Design methodology for wireless nodes with printed antennas*", in Proc. of 42nd Design Automation Conference*, 2004, pp. 291-296.

[50] A.V. Aho and S.C. Johnson, "Optimal Code Generation for Expression Trees", *Journal of the ACM*, July 1976, pp 488-501.

[51] P.S. Neelakanta and H. Dighe "Robust factory wireless communications: a performance appraisal of the Bluetooth^{TM} and the ZigBee^{TM} colocated on an industrial floor", *in Proceedings of the 29th Annual Conference of the IEEE Industrial Electronics Society*, Nov. 2003, pp. 2381-2386.

[52] A. Willig, D. Matheus and A. Wolisz, "Wireless Technologies in Industrial Networks", *Proc. of the IEEE,* June 2005, pp. 1130-1151.

[53] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Network.* , pp. 393–422, 2002.

[54] H. Karl and A. Willig, Architectures and Protocols for Wireless Sensor Networks. Wiley, 2005.

[55] A. J. Goldsmith and S. B. Wicker, "Design challenges for energy-constrained ad hoc wireless networks," *Wireless Comm.*, Aug. 2002. pp. 8–27.