Remotely Managed Logic Built-In Self-Test for Secure M2M Communications

Elena Dubrova[1], Mats Näslund[2], Gunnar Carlsson[3], John Fornehed[4], and Ben Smeets[2]

***Abstract*—A rapid growth of Machine-to-Machine (M2M) communications is expected in the coming years. M2M applications create new challenges for in-field testing since they typically operate in environments where human supervision is difficult or impossible. In addition, M2M networks may be significant in size. We propose to automate Logic Built-In Self-Test (LBIST) by using a centralized test management system which can test all end-point M2M devices in the same network. Such a method makes possible transferring some of the LBIST functionality from the devices under test to the test management system. This is important for M2M devices which have very limited computing resources and commonly are battery-powered. In addition, the presented method provides protection against both random and malicious faults including some types of hardware Trojans.**

I. Introduction

Over the last 30 years, the wireless communication industry has gone through several stages of development. In the first generation of wireless networks, communications were performed using analog type of transmission. In the second generation, the user's voice was sampled and send using digital signals, resulting in more reliable communications and increased network capacity. In the third and fourth generations, the wireless network users are able not only to exchange voice messages, but also have data access to the Internet.

The number of wireless subscribers has grown to 7 billions worldwide [1]. The next wave of growth is likely to come from connecting more devices rather than more people. Home appliances, cars, meters, sensors are expected to be wirelessly connected, accessible, and controlled via local networks or the Internet. Such *Machine-to-Machine* (M2M) type of communications are expected to provide an entirely new range of services that are appealing to the users [2]. It is forecasted that 50 billions of devices will be connected worldwide by year 2020 [3].

M2M communications create new challenges for security. First, more products which we use in everyday life become security-critical, e.g. connected baby monitor or smart fire alarm. It is difficult to attack an appliance which is not connected to anything but a power plug. Connectivity exposes devices to the outside world, introducing new vulnerabilities and risks. In addition, new applications whose malfunctioning can kill or injure people appear, e.g. implanted sensors that monitor the heart rate and communicate the data to the cloud for analysis. Such applications should be able to guarantee security that withstands potential attacks for the lifetime of the product, regardless of improvements in attacker's computational capabilities.

M2M communications also create new challenges for in-field testing since they typically operate in environments where human supervision is difficult or impossible. In addition, M2M networks may be significant in size. In this paper, we propose to automate Logic Built-In Self-Test (LBIST) by using a centralized test management system which can test all end-point M2M devices in the same network. Such a method makes possible transferring some of the LBIST functionality from the devices under test to the test management system. This is an advantage since M2M devices are characterized by very limited computing resources and commonly are battery-powered. In addition, the presented method provides protection against both random and deliberate faults including some types of malicious circuit modifications known as *hardware Trojans*.

Hardware Trojans (also known as *sleeper cells* [4]) have been around for a while, but in the past it was quite difficult to inject a Trojan into the supply chain. In today's globalized world where manufacturing is outsourced and the use of third-party IP from new vendors is widespread, this is no longer a problem. In modern chips it is nearly impossible to find circuitry which is injected during chip's tapeout and does not belong to the original design. Functional validation is further complicated by the fact that some redundant circuitry is typically added to a chip during the manufacturing stage in order to increase its yield [5].

The threat posed by hardware Trojans is widely recognized. Over the last few years, reports have been published to regulate suppliers of critical components [6]. The discovery of counterfeit chips in safety and security critical industrial and military products [7] made clear the importance of finding efficient protective measures against hardware Trojans.

The method presented in this paper provides a simple but efficient countermeasure against hardware Trojans described in [8] which exploit non-zero aliasing probability of LBIST. Our method executes LBIST using a different set of test patterns at each test cycle. As a result, the expected LBIST signature is *unknown* at the manufacturing stage. Therefore, it is not possible to predict in advance which circuit modifications produce the same signature as a fault-free signature and a Trojan of type described in [8] cannot be inserted.

The paper is organized as follows. Section II gives basic notation used in the sequel. Section III describes the traditional LBIST. Section IV reviews hardware Trojans. Section V gives

[1]E.Dubrova is with the School if Information and Communication Technology, Royal Institute of Technology, 164 40 Stockholm, Sweden, e-mail: dubrova@kth.se.

[2]M. Näslund and Ben Smeets are with Ericsson Research, Ericsson AB, 164 80 Stockholm, Sweden.

[3]G. Carlsson is with Development Unit Radio, Ericsson AB, 164 80 Stockholm, Sweden.

[4]J. Fornehed is with BNET Systems & Technology, Ericsson AB, 164 80 Stockholm, Sweden.

details of the presented method. Section VI concludes the paper and discusses open problems.

II. Preliminaries

Throughout the paper, we use "$\oplus$" and "$\cdot$" to denote the Boolean XOR and the Boolean AND, respectively.

The Boolean functions $\{0,1\}^n \to \{0,1\}$ are represented using the *Algebraic Normal Form (ANF)* (also called *Reed-Muller canonical form* [9]) which is an expression of type

$$f(x_0, x_1, \ldots, x_{n-1}) = \sum_{i=0}^{2^n-1} c_i \cdot x_0^{i_0} \cdot x_1^{i_1} \cdot \ldots \cdot x_{n-1}^{i_{n-1}},$$

where "$\sum$" is an XOR-sum, $c_i \in \{0,1\}$ and $(i_0 i_1 \ldots i_{n-1})$ is the binary expansion of i [10]. ANF is the most common representation for Boolean functions used in cryptographic systems [11].

An n-bit *Feedback Shift Register (FSR)* consists of n binary storage elements, called *stages*. Each stage $i \in \{0,1,\ldots,n-1\}$ has an associated *state variable* x_i which represents the current value of the stage i and a *feedback function* which determines how the value of i is updated.

A *state* of an FSR is a vector of values of its state variables. At each clock cycle, the next state of an FSR is determined from its current state by simultaneously updating the value of each stage i to the value of the corresponding feedback function f_i, $\forall i \in \{0,1,\ldots,n-1\}$.

If all feedback functions of an FSR are linear, then it is called a *Linear Feedback Shift Register* (LFSR). Otherwise, it is called a *Non-Linear Feedback Shift Register* (NLFSR).

An FSR can be implemented either in the *Fibonacci* or in the *Galois* configuration [12]. In the former, the feedback is applied to the input stage of the shift register only. All remaining feedback functions are of type $f_i = x_{i+1}$, for $i \in \{0,1,\ldots,n-2\}$. In the latter, the feedback can potentially be applied to every stage.

III. Traditional LBIST

Logic Built-In-Self-Test (LBIST) [13] uses a Pseudo-Random Pattern Generator (PRPG) to generate pseudo-random test patterns that are applied to the circuit under test and an output response compactor for obtaining the compacted responses to these patterns, called signature (see Figure 1). An incorrect signature indicates a fault.

The LBIST controller contains circuitry that controls the testing process: generation of pseudo-random test patterns, their application to the circuit under test and compaction of output responses. The controller first initializes the PRPG to a given initial state and then counts the required number of test patters. The initial state and the number of test patterns are defined by the test initialization parameters.

Pseudo-random patterns generated by the PRPG are fed into a circuit under test and propagated through its components. The resulting output signals are provided to the output response compactor, typically implemented by a Multiple Input Signature Register (MISR). MISR computes a signature representing the cumulative value of the output responses and forwards it to the decision logic.

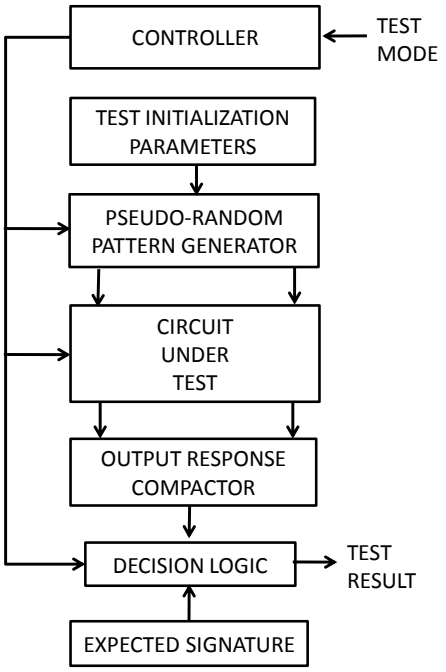

Fig. 1: Traditional LBIST module.

Since the output responses are compacted, a faulty circuit can produce the same signature as a fault-free one. This is known as an *aliasing* error. If an MISR with a primitive connection polynomial is used[1], then the aliasing probability is bounded by $1/2^n$, where n is the size of the MISR [15].

Decision logic compares the signature computed by the MISR to the expected "good" signature. If the MISR signature matches the expected signature, the circuit passes the test. Otherwise the circuit fails the test.

The test initialization parameters and expected signature are typically stored in a memory (e.g. Flash) or hard-wired into a chip during the manufacturing stage [16].

The management of the traditional LBIST is performed using a processor which either resides on a chip, or on the same board as a chip [17]. The processor initiates LBIST by sending a "test mode" signal to the LBIST controller. Typically, LBIST is initiated at power-up and/or restart, or in response to some external trigger, e.g., if a hardware or software supervising the chip indicates a fault.

IV. Hardware Trojans

A hardware *Trojan* is a malicious modification of a design intended to bypass or disable its security. There are two different types of Trojans [18].

1) *Functional* Trojans add or remove transistors, gates, wires, or other components to/from the original circuit.
2) *Parametric* Trojans reduce the reliability of a chip by thinning its wires, weakening its transistors, or subjecting the chip to radiation.

Existing techniques for detecting hardware Trojans include:

[1] An irreducible polynomial of degree n is called primitive if the smallest m for which it divides $x^m + 1$ is equal to $2^n - 1$[14].

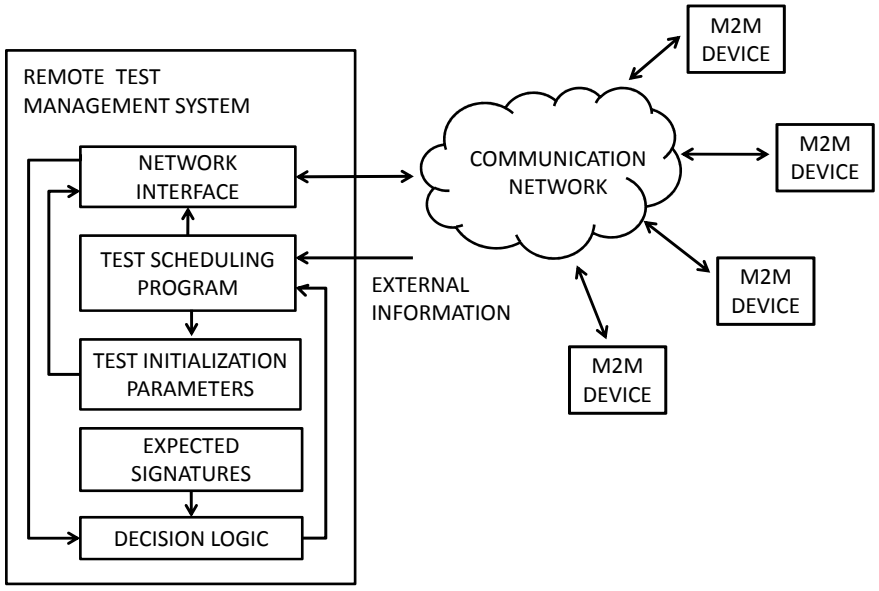

Fig. 2: Presented method.

1) *Physical inspection*, consisting in grinding the layers of a chip one-by-one and scanning the exposed circuitry by various visual inspection methods, e.g. optical inspection [19];
2) *Testing* (functional or BIST), consisting in applying test stimuli to a chip and comparing its output to the specification [20];
3) *Side-channel analysis*, in which signals emitted by a chip, e.g. leakage current, path delays, electromagnetic radiation, etc. are measured and analyzed [21], [22].

Various countermeasures have been developed to protect against activation of certain types Trojans, or to allow for operating correctly in presence of unknown Trojans [23]. The former typically involves data guards such as scrambling or obscurification, or hardening a design against specific triggers. The latter is usually done by replication, fragmentation and voting, as in the traditional fault-tolerant design techniques [24].

Overall, existing methods for hardware Trojan detection are still in their infancy. Typically, they focus on a specific type of Trojans, with no single technique being able to provide a complete coverage. For example, the recent attack on the Random Number Generator (RNG) of Intel's Ivy Bridge processor [8] demonstrated that the traditional LBIST may fail even the simple case of stuck-at fault type of hardware Trojans. Such Trojans can be injected by changing the dopant polarity of selected transistors. The modifications do not change any metal or polysilicon layer of the chip and therefore they are practically invisible to optical inspection. The points of modifications are selected so that the LBIST signature computed for the Trojan-injected circuit is the same as the fault-free signature. Thus, the Trojan does cause LBIST to fail.

It is possible to combat LBIST-based Trojans by introducing a configurable key which determines the initial state of PRPG and hence the expected LBIST signature. The key is programmed into the circuit by the user after the manufacturing stage [25]. The method presented in this paper provides an alternative way of making LBIST signature unknown at the manufacturing stage without a need for extra programming.

V. Presented Method

In this paper, we present an LBIST method which uses a centralized remote test management system to test all devices in the same network (see Figure 2). The remote test management system contains a test scheduling program, test initialization parameters, expected signatures, decision logic and network interface. The expected signatures can be either pre-computed for a given set of test initialization parameters and stored in a database, or computed on-the-fly by simulation.

Upon deciding to initiate a test cycle for all or some devices, the test scheduling program instructs the test initialization parameters module which parameters to send. These parameters are transmitted through the network interface and the communication network to the selected devices.

The diagram of a remotely managed LBIST module is shown in Figure 3. It differs from the traditional LBIST in Figure 1 in that the test initialization parameters, expected signature and decision logic blocks are transferred to the test management system. This is an additional benefit for the end-point M2M devices in which computing and power resources are typically very limited.

We assume that the board implementing an end-point M2M device contains a communication module, e.g. a modem, which performs the communication between the M2M device and the remote test management system, an CPU which runs processes related to the communication, and Basic Input/Output System (BIOS) which provides the functionality and interface for communication during LBIST.

Then the test initialization parameters are received through the communication module, the LBIST test cycle proceeds

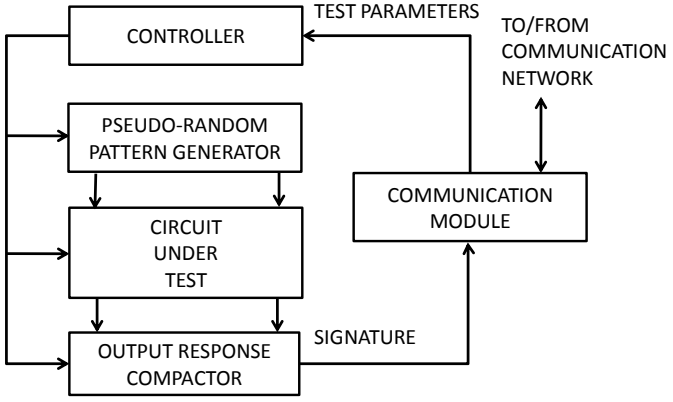

Fig. 3: Remotely managed LBIST module.

as normally and the signature is computed. This signature is returned to the remote test management system for the analysis on passing/not passing the test.

The test management system takes "human-like" decisions regarding which devices to test, when, and how. These decisions are taken on the base of the information received by monitoring global external factors such as environment, the interaction between devices, abnormal responses, etc. For example, meteorological sensors that register the wind in various locations within a given area may be tested immediately after harsh weather conditions, e.g a thunderstorm. As another example, a device may request the management system to test another device if several attempts to communicate with it failed. In both examples, faults might be detected earlier, implying higher availability and safety. The idea of context-aware automation and decision optimization is not new [26], but to our best knowledge it has not been applied to LBIST.

Another advantage of the centralized management is that testing can be carried out using different sets of test patterns. In the traditional LBIST, the same set of pseudo-random patterns is used at every test cycle. At each test cycle, the PRPG starts from the same initial state and generates the same number of test patterns which are defined by the test initialization parameters stored on-chip/board. Accordingly, the same set of input stimuli is applied to a circuit under test and hence the same subset of faults can be detected. By changing the set of test patterns for each test cycle, we can cover different subsets of faults. Therefore, the presented method can potentially provide a higher fault coverage compared to the traditional LBIST. Furthermore, it can detect malicious and unanticipated faults, including hardware Trojans. In the traditional LBIST, an adversary who knows the set of test patterns generated by the PRPG can made suitable circuit modifications which result in the same signature as a fault-free circuit signature. On average, the adversary has to do 2^{n-1} simulation trials in order to inject a Trojan which does not trigger LBIST [8].

To show the reader how such an attack can be performed, consider an RNG which generates numbers in the range $\{1,2,\ldots,15\}$. Such an RNG can be implemented by a 4-stage Non-Linear Feedback Shift Register (NLFSR) with the

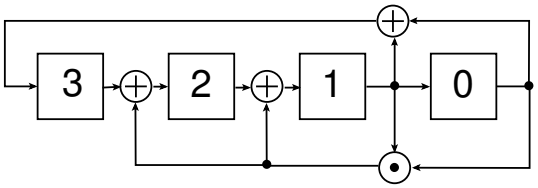

Fig. 4: 4-bit NLFSR from the example.

Test pattern from LFSR $(x_3x_2x_1x_0)$	Fault-free case		With fault injected	
	NLFSR response $(x_3x_2x_1x_0)$	MISR signature $(x_3x_2x_1x_0)$	NLFSR response $(x_3x_2x_1x_0)$	MISR signature $(x_3x_2x_1x_0)$
1011	0011	0011	0001	0001
1010	1101	0101	1101	0100
0101	1010	0001	1000	1010
1101	1110	0111	1100	1001
1001	1100	0110	1100	0001
1011	0011	0000	0001	1000
1010	1101	1101	1101	1001
0101	1010	0101	0001	0101

TABLE I: Example of a hardware Trojan not detected by LBIST.

following feedback functions (see Figure 4):

$$f_0(x_1) = x_1$$
$$f_1(x_0,x_1,x_2) = x_2 \oplus x_0x_1$$
$$f_2(x_0,x_1,x_3) = x_3 \oplus x_0x_1$$
$$f_3(x_0,x_1) = x_0 \oplus x_1.$$

Suppose that the LBIST which protects such an RNG uses 4-bit LFSR with the connection polynomial $1 \oplus x \oplus x^2 \oplus x^3 \oplus x^4$ as a PRNG and a 4-bit MISR with the connection polynomial $1 \oplus x^3 \oplus x^4$ as an output compactor. We assume that, at each clock cycle, the current 4-bit state vector of the LFSR is used as a test pattern. The LFSR is initialized to some non-zero state and the MISR is initialized to (0000). Both, LFSR and MISR are implemented in the Galois configuration.

The probability that an attacker successfully guesses the number generated by an n-bit NLFSR is $1/2^n$. However, by setting k internal flip-flops of the NLFSR to a constant value it is possible to reduce the complexity of the attack to $1/2^{n-k}$.

Suppose that the attacker knows that the initial state of the LFSR is (1011) and that 8 tests patterns are applied to compute the MISR signature. Then, the attacker can calculate the expected "good" MISR signature by simulation. From the 3rd column of Table I we can see that this signature is (0101). The attacker can investigate which of the NLFSR's flip-flops should be set to constant-0 or constant-1 value in order to get the same signature. In our example, the signature (0101) can be obtained if the flip-flop corresponding to the bit 1 of the NLFSR is set to 0 (see last column of Table I). So, the attacker can inject such a fault into the NLFSR and reduce the complexity of the attack by one half. Note that, in general, detecting this type of Trojans in a large design by optical inspection is very difficult since only the dopant polarity of a few transistors has been changed. No metal or polysilicon layers of the chip have been altered. Since optical inspection is not feasible and the Trojan passes LBIST, an engineer who

verifies the design cannot recognize a difference between a Trojan-injected circuit and a Trojan-free one. Consequently, he/she is not able to identify a trustworthy "golden" chip (which is known not to have any malicious modifications). Without such a chip, the majority of post-manufacturing Trojan detection techniques [18] cannot be used.

The presented method mitigates this problem because it executes LBIST using a different set of test patterns at each test cycle. Since the expected signature is unknown at the manufacturing stage, an attack based on selecting suitable values for the Trojan which result in the same signature as a fault-free circuit signature cannot be performed.

In the scenario we described above, the remote test management system sends to a device the test initialization parameters and the device replies with a signature. Another scenario is possible, in which the remote test management system sends to a device both, the test initialization parameters and the expected signature. Then, the device computes the signature, compares it to the received expected signature and replies with pass/not passed. While such a case would involve the same volume of data transferred, it might be preferable for applications in which the downlink bitrate of the receiving device is higher than its uplink bitrate.

VI. Conclusion

We presented an LBIST method which uses a remote management system which can test all end-point M2M devices in the same network.

Acknowledgement

The first author was supported in part by the research grant No SM12-0005 from the Swedish Foundation for Strategic Research.

References

[1] B. Sanou, "ICT facts and figures," 2014. http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2014-e.pdf.

[2] D. Boswarthick, O. Elloumi, and O. Hersent, *M2M Communications: A Systems Approach*. Willey, 2012.

[3] Ericsson AB, "More that 50 billions connected devices," 2012. www.ericsson.com/res/docs/ whitepapers/wp-50-billions.pdf.

[4] E. Sperling, "The next big threat: Manufacturing," 2014. http://semiengineering.com/ manufacturing-and-integration-risks/.

[5] P. Gupta and E. Papadopoulou, "Yield analysis and optimization," in *The Handbook of Algorithms for VLSI Physical Design Automation*, RC Press, 2011.

[6] Department of Defence, "Federal register," Nov. 2013. http://www.steptoe.com/assets/ htmldocuments/DFARS

[7] C. Gorman, "Counterfeit chips on the rise," *IEEE Spectrum*, vol. 49, no. 6, pp. 16–17, 2012.

[8] G. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, "Stealthy dopant-level hardware Trojans," *Proceedings of Cryptographic Hardware and Embedded Systems (CHES'2013), LNCS 8086*, pp. 197–214, 2013.

[9] D. H. Green, "Families of Reed-Muller canonical forms," *International Journal of Electronics*, vol. 70, pp. 259–280, 1991.

[10] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and their Applications*. Cambridge Univ. Press, 1994.

[11] T. W. Cusick and P. Stănică, *Cryptographic Boolean functions and applications*. San Diego, CA, USA: Academic Press, 2009.

[12] E. Dubrova, "A transformation from the Fibonacci to the Galois NLFSRs," *IEEE Transactions on Information Theory*, vol. 55, pp. 5263–5271, November 2009.

[13] E. McCluskey, "Built-in self-test techniques," *IEEE Design and Test of Computers*, vol. 2, pp. 21–28, 1985.

[14] S. Golomb, *Shift Register Sequences*. Aegean Park Press, 1982.

[15] M. Damiani, P. Olivo, M. Favalli, S. Ercolani, and B. Ricco, "Aliasing in signature analysis testing with multiple input shift registers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 12, pp. 1344–1353, 1990.

[16] H.-J. Wunderlich, "BIST for systems-on-a-chip," *Integration, the VLSI Journal*, vol. 26, no. 1-2, pp. 55 – 78, 1998.

[17] K. Chakrabarty, "Modular testing and built-in self-test of embedded cores in system-on-chip integrated circuits," in *The Embedded Systems Handbook* (R. Zurawski, ed.), pp. 27–2–27–27, CRC Press, 2006.

[18] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," *IEEE Design Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[19] S. Skorobogatov, "Physical attacks and tamper resistance," in *Introduction to Hardware Security and Trust* (M. Tehranipoor and C. Wang, eds.), Information Security and Cryptography, Springer Berlin / Heidelberg, 2011.

[20] E. Dubrova, M. Näslund, and G. Selander, "Secure and efficient LBIST for feedback shift register-based cryptographic systems," in *Proceedings of European test Symposium (ETS'2014)*, pp. 183–189, May 2014.

[21] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, (London, UK, UK), pp. 104–113, Springer-Verlag, 1996.

[22] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," pp. 388–397, Springer-Verlag, 1999.

[23] M. Beaumont, B. Hopkins, and T. Newby, "Hardware Trojans - prevention, detection, countermeasures," Tech. Rep. DSTO-TN-1012, Australilan Department of Commerce, July 2011.

[24] E. Dubrova, *Fault-Tolerant Design*. Springer, 2013.

[25] E. Dubrova, M. M. Näslund, G. Carlsson, and B. Smeets, "Keyed logic BIST for trojan detection in SoC," in *Proceedings of International Conference of System-on-Chip (SoC'2014)*, 2014.

[26] W. Dargie, *Context-Aware Computing and Self-Managing Systems*. Chapman & Hall/CRC Studies in Informatics Series, 2009.