Chapter 3

Memory protection

3.1 Introduction

The ever-increasing complexity of the digital circuits result in more and more crucial problems in testing, both in the generation of test vectors and in the evaluation of the test responses. The most promising approach for VLSI-based circuits and systems is the application of the error correcting codes, widely used in telecommunication application. However differences in the fault models and in the technical requirements prohibit the simple adaptation of the solutions in the telecommunication for the testing problems and new code designs are required. The example of the error protection of semiconductor memories shows, that information redundancy resulting in hardware overhead can be partially substituted by time redundancy.

Despite the formal analogy between the error control problems in the traditional field of telecommunication and hardware testing in using information redundancy for checking the correctness of the data, there are essential differences both form the theoretical and practical points of view between these fields of application. As example the most widely used error models in the telecommunication originate the corruption of data in the transfer through a noisy channel. In the case of the well-known symmetric channel hypothesis the main consequences for the interrelationship between data and error are the following properties:

- The noise is independent of the data transmitted. This assumption concludes in the symmetric channel model.
- The noise is time-limited, resulting in the independent bit error model, assumed, that the time limit is less than the bit transfer time in the channel, or in the burst error model, if it exceeds that.

The resulting model consists a simple, homogenous fault to error mapping mechanism. The number or faulty bits or burst length becomes to a measure of the error probability. On the technical side:

- From the bit or character error probability a maximal number of errors to cope with is established.
- Additionally to the error detection, their correction to a limited extent is a primary goal
- In telecommunication applications the most simple way to assure a high error detection probability is to subdivide the message data stream into segments and attach to each segment check symbols. In this process moderate increases in the information redundancy, like check bits or characters are tolerable, as they decrease only the code speed and the selection of the block length is within limits at the designer's choice.

However in the case of hardware testing the fault-to-error mapping mechanism is rather complex, even when using simplifying assumptions in the fault model. The most widely used fault model assumes the presence of single or multiple so-called stuck-at faults. Their effect is that at the fault site a constant logic value is present independently of the correct value. This model is asymmetric, as in the case of a match between the correct value and this constant, no error is observable at the fault site, and the fault is manifested only as an error if they are of an opposite value. This model results in the so-called unidirectional error model well-describing the effects of a stuck-at fault at an output of the circuit. However even a single bit transient or permanent fault inside of the circuit, e.g. at the root point of multiple paths to the outputs or inside a feedback loop can cause errors at numerous circuit nodes and outputs during multiple time phases.

On the other side the assumption of the equiprobability of all error patterns is equivalent with independency of the value sequences in the fault-free and faulty case contradicting such observations, like an automaton with a cyclic operation saves this basic property even for many faults causing errors at the outputs.

Accordingly, in contrast to the telecommunication applications

- in typical hardware testing tasks no homogenous, simple error models can be used,
- the limitation on the fault effects can not result in the general case in a simplified error model neither the sense of time nor in space distribution.
- there is no strict correlation between the number of faulty characters in an error pattern and its probability to occur
- the number of characters in a pattern to check is determined overwhelmingly by external design factors, like the device under test, the test procedure developed for it,

From the practical side:

- the allowed amount of information redundancy is low, as in the majority of the applications it results in a significant hardware overhead.
- the processing speed must be at least as high as the normal operation of the circuit, in order to detect dynamic faults, appearing only at a sufficiently frequencies.
- in the majority of applications no error correction is required.

As result instead of a pure analytical methodology either all faults are simulated or benchmark-like error pattern sets are used for the estimation of the fault coverage when using an error control coding (ECC) technique for fault detection.

Additionally to the problems in test response evaluation the generation of the test input sequences becomes more and more crucial, especially in built-in test application. Both the number of the potential faults in a fault model and the length and complexity of the test sequence to detect them make the traditional approaches inapplicable. The exhaustive testing checking the reaction of the circuit under test to all possible input combinations is irrealistic due to the huge number of test vectors to be applied and some simple method is required assuring a tolerable overhead in testing time.

In the following sections we present some characteristic examples for the illustration of the main problems and error control codes based solutions for the hardware testing problems.

3.2 Information vs. time redundancy

The protection of semiconductor RAM memories is a very typical and traditional field of application of the error correcting codes in hardware devices, clearly illustrating the basic problems and main concepts in the most simple form. Its speciality is, that in this case an error correction is typically required too [3]. .

Memory chips can fail due to different failure modes:

- Permanent or hard errors originate in manufacturing problems or aging. This defects can affect

- a single cell or word, e.g. the overwhelming majority of defects in DRAM chips result from single-bit capacitor oxide failure. The “hard” (repeatable) failure rate of DRAM chips in the field is by about 200 FIT. (200 failures per 10^9 device-hours).
- a group of adjacent cells (e.g. by a parasitic capacitive coupling between them),
- row(s) or column(s) in the cell matrix (due to errors in the row or column address decoders), or
- all cells on a data line (due to a defective read/write amplifier or data wire)

\- Soft errors originate in particles radiation on power supply noises and result mostly in single or multiple cell errors, occurring according to the experiences by one order of magnitude more often than hard ones.

Permanent faults are typically detected in the power-on phase, when thoroughgoing functional tests can be executed, as the memory contains no valuable data. In the case of an error, degradation can be performed. In mission-time critical applications however both of the hard and soft faults must be masked in some form. When coping with soft errors the only way for their detection and eventually for their correction is the use of the hardware redundancy in the form of some error correcting code.

In a safety-relevant application the fault model consists both of hard and soft bit and word errors. At first, in the following we will present the main effects and concepts in handling soft errors following in a slightly simplified form the ideas presented in [2]. .

Let assume, that we want to build up a memory of a capacity of M words each of a length N. The occurrence of soft errors (Fig. 3.) is usually modelled as a Poisson process of a soft bit error arrival rate λ. We assume, that the whole memory area is used and neglect the effect of

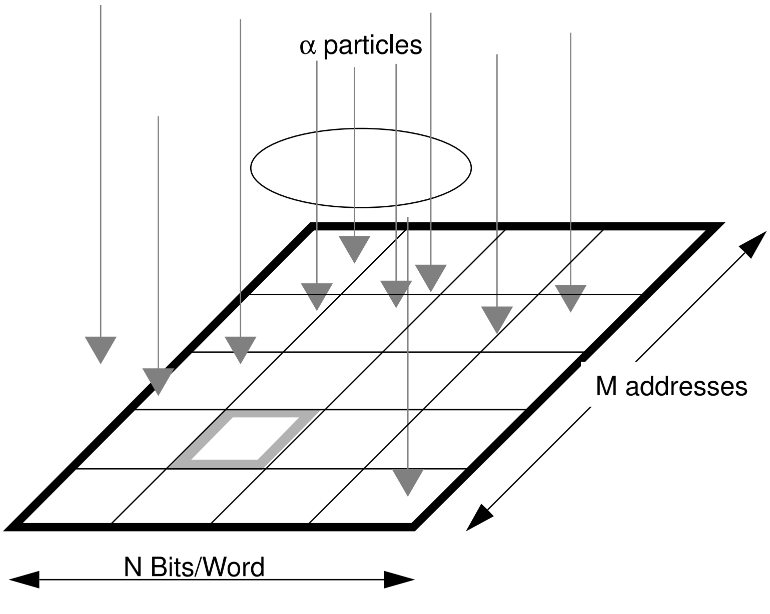

Fig. 3. Soft errors in a memory

fault masking due to overwriting of potentially faulty, but still unused data by new values. The probability of a single cell remaining fault-free during a period T is $P_0(T) = e^{-\lambda T}$ and the reliability of the chip, i.e. the probability that each bit in each word of it remains fault-free is $_{protected}(T) = [P_0(T)]^{N \times M} = e^{-\lambda t}$. It is obvious, that a growth in the memory size drastically reduces the reliability. As the memory is unprotected, it will fail at the first fault and the mean time between failure (MTBF) will be as low, as $MTTF = 1/(\lambda NMT)$. Moreover, the mean time T' of a fault-free memory service period of a reliability of 99% ($R_{unprotected}(T') = 0.99$) is only by about one hundredth of the MTBF. In the case of a medium massively parallel multiprocessor configuration consisting of 100 processors with a memory capacity in each node of 32 MBytes this time period is as low as approximately 130 seconds, even if we assume a soft error rate 1 in 10^8 hours/ bit.

If each word in the memory has a d error correcting code by adding of K check bits, than the probability that a word of the length $(N + K)$ remains fault-free or correctable after a time period T increases to $P_{word, ECC}(T) = \sum_{i=0}^{d} P_{word, i}(T)$, where $P_{word, i}(T)$ denotes the probability of a word being hit i-fold during T, assumed, that bit errors in a word occur independently and repeated hits into a cell do not repair its content. The reliability of the memory, i.e. the probability, that no word in it consists of an uncorrectable error is $R_{ECC}(T) = [P_{word}(T)]$. As shown in Fig. 3. the use of ECC assures a higher initial reliability, but due to the redundancy introduced by the check bits the long-term reliability of the memory decreases. Moreover, as the possibility of tolerate more errors in a word would require more and more redundancy in the form of ECC check bits another way must be selected.

As error detection and correction occur only when the corresponding data is red and if a data word is not used for a longer period it can become uncorrectable due to the cumulative effect of multiple faults. An increase in the error detecting and correcting capabilities of the encoding would require more check symbols, i.e. it would result in an additional hardware overhead.

An alternative solution is based on the idea of the reduction of the mission time of the ECC based protection. In the case of memory scrubbing each word in the memory is periodically read out with a relatively low time period T_s and correctable errors are immediately removed. In this way information and hardware redundancy is substituted by time redundancy. The reliability in this case will be

$R_{scrubbing}(T) = [R_{ECC}(T_s)]^{T/T_s}$., as the required total error-free service time T will be subdivided into T/T_s time slots each of the length T_s. The three approaches can be well-characterized by comparing their reliability, as shown in Fig. 3.

The most typical ECC-based memory protection methods used in the practice are in an increasing fault-tolerance capability as follows [3]. :

- Single bit error detecting (SED) with a parity bit
- Single bit error correcting, double bit error detecting (SEC-DED) codes with the use of a modified Hamming code
- Single byte error correcting (SbEC) codes (Hamming, Burton, Fujiwara, Hong-Patel codes)
- Single byte error correcting, double byte error detecting (SbEC-DbED) codes, like Reed-Solomon [4]. , [5]. , Kaneda-Fujiwara codes.

Another idea in reducing the information redundancy is based on the explicit utilization of the permanent nature of the hard faults. The main idea is as follows: let assume, that a memory word, protected by a SEC-DED code consists a hard stuck-at and a soft error. As there are to bits in error, the syndrome decoding logic signalize an uncorrectable error. However when writing back the binary complement of this faulty data, the bit value in the hard fault position will coincide with the stuck-at constant, and in the read data only the soft error will be active. After correcting this, the hard error reduces to an erasure error.

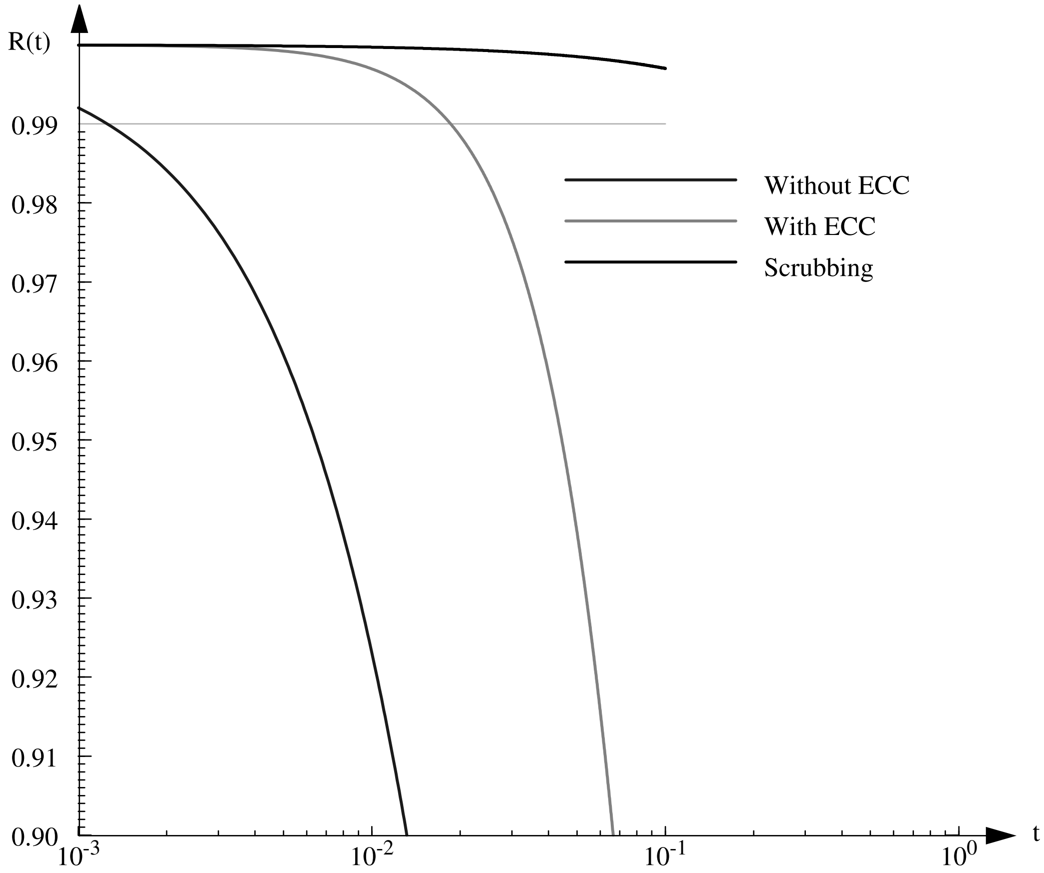

Fig. 4. Reliability of the memory for different protection strategies

3.5 Error detecting codes in compact testing

In the traditional scheme for hardware testing the output of the device under test (DUT) as response to some test input sequence is compared with the output sequence from a known fault-free reference device, or its model. However for the high-speed testing of complex devices, especially in the case of self-test neither the physical reference unit nor the model based approach can be applied, as the first one is equivalent with the duplication, and the second one can not fulfil the speed requirements. A possibility is the storage of the reference output sequences, but for long sequences the huge amount of storage area makes it impractical. [6]. .

The idea of compact testing dates back to 1975 [7]. . It is based on the observation, that when the output sequences of the DUT and of the reference device are treated as data parts of a systematic code their match can be evaluated by generating and comparing the check part of this code.

For checking the following information compaction methods are used:

- Ones-count compression using the binary Hamming-weight of the data stream as check information;

-Transition-count, i.e. the total number of rising and falling edges in the stream, corresponding to the Hamming weight of the derivate of the stream;

- Accumulator compression testing, using the Hamming-weight and its time integral as reference;

- Signature analysis, using binary linear compaction techniques; (a circuit is said to be linear over GF(2), if it consists only of storage elements and XOR type gates);

This last category is the most widely used one as general-purpose test compactor, as it can be proven, that only it provides a maximal error detection, if at the phase of compaction method selection the reference sequences are unknown. In the case of equiprobable errors, the probability of an undetected error, in other terminology of a fault escape equals approximately 2^{-n}, where n is the signature bit count. In this way a proper detection probability is assured in the majority of the applications using 16 check bits with an approximate fault escape probability of 1.5×10^{-5}. Moreover, the same circuits, which compact the test response, can easily reconfigurated to an exhaustive pattern generator supporting fully the self-test of a combinational circuit. In the built-in self-test of integrated circuits and boards this idea is realized in different forms, like in the following typical examples:

- Built-in evaluation and self-test (BEST [8].) using separate pseudorandom test generator and linear compactor;
- LSSD on-chip self- test (LOCST [9].), combining the concepts of level-sensitive scan design, single-bit pseudorandom pattern generation and single-bit linear compaction;
- Built-in logic block observer (BILBO [10].) using registers for multiple-output pseudorandom test generation and multiple-input linear compaction by implementing their function in a controllable mode;
- CEBS [11]. , combining pseudorandom test generation and linear compaction for the chip level, and boundary scan for the board level testing.

If the circuit has only a single output to be checked, than the use of error detecting polynomial codes, referred as single-input signature analysis is the most widely used solution, due to the high fault coverage and the simplicity of the checking circuit, depicted in Fig. 3.

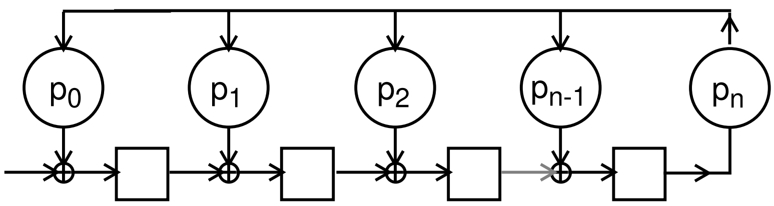

Fig. 6. Polynomial division circuit

Despite the formal analogy of this testing problem with the traditional field of application, the essential fault models in the background are rather different. The first difference is, that the length of the data part is not previously fixed or limitable during the code design phase. The analyzer must be capable of processing a serial bit pattern of an arbitrary length, which can reach several millions of bits.

A typical benchmark set of consists of the following error patterns expressing some characteristic types of correlation due to the hardware faults between the error bits [15]. :

- Errors with a few of error bits (1,2,3).
- Burst type error, where the positions in error are located in a limited interval.
- Cyclic and repeated errors, where the same error subsequence is repeated in time.

The basic phenomena in selecting the first two groups is not the independency of the bits in error, as in a symmetric channel model, but the often performed minimization of the test input sequence in order to shorten the time needed to test. In an ideal minimal test set only a single test is contained for each fault. The burst model us typical, if the test input sequence is composed as concatenation of the subtests of the individual and independent functional units. In the practice primitive polynomial based checks are used, for which a maximal detection of the error patterns enlisted above is granted.

For the check of multiple output circuits many pure heuristic signature analyzers structures were evaluated. In the remaining, more systematic approaches the main idea in information compression is essentially analogous with the single-line signature analysis: the sequence of binary m-tuples of length L appearing on the outputs of the DUT as test response is compressed by generating the check symbols of a multi-bit symbol based error detecting code.

An interesting idea appears in [12]. , where the use of a Read-Solomon (RS) code checker was proposed as multiple-input signature analyzer registers (MISR). We assume, that the reader is familiar with the basics of the RS codes, otherwise please refer [4]. or [5].

In RS encoding the symbols (binary m-tuples) are interpreted as elements of the Galois-field $GF(2^m)$. In the most common case the RS code can be generated by a polynomial having subsequent elements from $GF(2^m)$ as roots, i.e. the generator polynomial is of the form $p(x) = (x-\alpha^j) \cdot (x-\alpha^{j+1}) \cdot \ldots \cdot (x-\alpha^{j+k})$ for an arbitrary integer j (Here α is a primitive element of $GF(2^m)$.) By mapping the input data symbols into the corresponding $GF(2^m)$ elements, the input symbol stream can be interpreted as a polynomial over $GF(2^m)$ with the highest degree term corresponding to the first character in time. As check symbols the remainder from the division of the input and generator polynomials will be used. The polynomial division circuit is similar to that shown in Fig. 3. , however with storage elements, constant multipliers and adders performing operations over $GF(2^m)$.

RS codes are maximum distance separable (MDS). For a code consisting no more binary m-tuples as symbols, than 2^m-1, RS encoding ensures the detection of as many random symbol errors, as checking characters are applied. This means a maximal utilization of the redundancy in the form of the checking characters. Due to this feature this codes are favorite candidates for a wide field of applications extending from telecommunication to consumer electronics.

An RS code is improper solution as MISR, despite its excellent error detection characteristics in telecommunication type applications. The typical benchmark pattern set of the multiple-input signature analysis consists of the following time and/or space limited patterns [17].

- Single in phase (SIP), with a corresponding error polynomial $e(x) = a_j x^i$. Here the value of the error symbol is described by the first, and its position in time by the second term.

- Same error pattern (SEP), where the same error symbol is repeated in time, with an error polynomial $e(x) = \alpha^j r(x)$. Here the binary polynomial $r(x)$ describes the repetition in time of the error character of value α^j.

- Single in line (SIL), as a special case of SEP with errors at a single output of the DUT. The error polynomial is of an identical form, as for SEP, however here $0 \le j \le m$, because the single bit errors to be repeated correspond to the first m states in the state transition graph of the polynomial $g(x)$ generating $GF(2^m)$.

As all repetition polynomials are binary ones, and the error pattern is obviously undetectable, if his polynomial is a multiple of the generator one, the error detection features of the RS encoder depend on the minimal degree nonzero binary polynomial $G(x)$, for which $G(x) | p(x)$. From the mathematical side this problem is essentially identical with the estimation of the generator polynomial of a Bose-Chaudhuri-Hocquenghem (BCH) code with the solution $G(x) = \mathrm{l.c.m}(m_j(x), m_{j+1}(x), \ldots, m_{j+k-1}(x))$,where $m_i(x)$ denotes the minimal degree binary polynomial, for which $m_i(\alpha^i) = 0$.

E.g. in the j=1, m=4, k=4 case (four check characters of a width of four bits) over $GF(2^4)$ generated by $(x^4 + x + 1)$, the minimal binary polynomial of the RS checker becomes to $G(x) = \mathrm{l.c.m}(m_1(x), m_2(x), m_3(x), m_4(x)) = (x^4 + x + 1) \cdot (x^4 + x^3 + x^2 + x + 1)$ i.e. the degree of $G(x)$ is only 8, and the fault escape probability is as high, as $2^{-8} = 1/256$! A traditional MISR design having the same number of signature bits would assure an approximate fault escape probability of the value $2^{-n} = 2^{-16} = 1/65536$. For practical sense that means, that applying RS codes e.g. in CPU testing, there is a non-negligible probability to leave a bit manipulation fault resulting in error patterns of the SIL type undetected.

As summary of the error detection features of the RS encoders in MISR applications, we can state, that they are usually far away from the theoretical upper limits. It should be pointed out, that this drawbacks are direct consequences of the different fault models in hardware testing as for telecommunication applications. A main conclusion from the example above is, that even the well-proven solutions need a careful rechecking while transplanting them from a field of application to another one.

References

[1]. Hohl, W.; Michel, E.; Pataricza, A.: Hardware Support for Error Detection in Multiprocessor Systems - A Case Study. Microprocessors and Microsystems, Vol 17., No 4., May 1993., pp. 201-206.

[2]. Goodman, R.M.; Sayano, M.: The Reliability of Semiconductor RAM Memories with On-Chip Error-Correction Coding, IEEE Trans. on Information Theory, Vol. IT-37, No. 3, May 1991, pp. 884-896.

[3]. Rao, T.R.N.; Fujiwara, E.: Error-Control Coding for Computer Systems, Prentice-Hall International, Inc. Englewood Cliffs, NJ, 1989.

[4]. Györfi, L.; Vajda, I.: A hibajavító kódolás és a nyilvános kulcsú titkosítás elemei Budapesti Müszaki Egyetem, 1991.

[5]. MacWilliams,F.J., Sloane, N.J.A.: The Theory of Error-Correcting Codes, North Holland, Amsterdam, 1977.

[6]. Abramovici, M.; Breuer, M.A.; Friedman, A.D.: Digital systems testing and testable design, Computer Science Press, 1990.

[7]. Benowitz, N. et al.: An Advanced Fault Isolation System for Digital Logic. IEEE Trans. on Computers, Vol. C-24., No. 5., May 1975., pp. 489-497.

[8]. Lake, R.: A Fast 20K Gate Array with On-Chip Test System, VLSI System Design, Vol.7, No. 6, June 1986, pp. 46-66

[9]. LeBlanc, J.: LOCST:A Built-In Self Test Technique, IEEE Design and Test of Computers, Vol.1, No. 4, Nov. 1984, pp. 42-52

[10]. Konemann, B. et al.: Built-In Test for Complex Digital Integrated Circuits, Digest of Papers 1979 Test Conference, pp. 37-41.

[11]. Butt, H.H.; El-Ziq, Y.M.: Impact on Mixed-Mode Self-Test of Life-Cycle Cost of VLSI Based Designs Proc. Int. Test Conf., 1984, pp. 103-111.

[12]. Iwasaki, K., Arakawa, F.: An Analysis of the Aliasing Probability of Multiple- Input Signature Registers in the case of a 2m-ary Symmetric Channel. IEEE Trans. on Computer Aided Design, Vol. 9., No. 4., Apr. 1990., pp. 427-438.

[13]. Iwasaki, K.: Analysis and Proposal of Signature Circuits for LSI Testing. IEEE Trans. on Computer Aided Design, Vol. 7., No. 1., Jan. 1988., pp. 84-90.

[14]. Pataricza, A.: An Improved Signature Analysis Based CPU Self-Test. Microprocessing

and Microcomputing, The EUROMICRO Journal, Vol. 23., 1988., pp. 167-172.
[15]. Smith, J.E.: Measures of Effectiveness of Fault Signature Analysis. IEEE Trans. on Computers, Vol. C-29., No. 6., June 1980., pp. 510-514.
[16]. Saluja, K.K., Karpovsky, M.: Testing Computer Hardware through Data Compression in Space and Time. IEEE 1983 International Test Conference, 1983., pp. 83-88.
[17]. Hassan,S.Z. et al.: Parallel Signature Analyzers Detection Capability and Extensions. IEEE Compcon Spring, 1983., pp. 440-445.
[18]. Pataricza, A.: An Optimal Input-Transformation Based Signature Analyzer Structure for Multi-Output Circuits. Proc. FTSD-11., Suhl, 1988., pp.240-245.
[19]. McCluskey, E.J.; Bozorgui-Nesbat, S.: Design for Autonomous Test, IEEE Trans. on Computers, Vol.C-30., No.11., Nov. 1981., pp.866-875.
[20]. McCluskey, E.J.: Verification Testing. 19th Annual Design Automation Conference. Las Vegas, Nevada, June 14-16 1982., pp.495-500.
[21]. McCluskey, E.J.: Verification Testing - A Pseudoexhaustive Test Technique. IEEE Trans. on Computers, Vol. C-33., No.6., June 1984.
[22]. Tang, D.; Woo, L.S.: Exhaustive Test Pattern Generation with Constant Weight Vectors. IEEE Trans. on Computers, Vol. C-32., No.12., Dec. 1983., pp.1145-1150.
[23]. Selényi, E.: New Methods for Pseudoexhaustive Testing. Periodica Polytechnica. Series Electrical Engineering, Vol.31., No.1-2., 1987., pp.11-20.