Input Vector Monitoring On line Concurrent BIST based on multilevel decoding logic

Ioannis Voyiatzis
Technological Educational Institute of Athens, Department of Informatics, Greece

Abstract - Input Vector Monitoring Concurrent Built-In Self Test (BIST) schemes provide the capability to perform testing while the Circuit Under Test (CUT) operates normally, by exploiting vectors that appear at the inputs of the CUT during its normal operation. In this paper a novel input vector monitoring concurrent BIST scheme is presented, that reduces considerably the imposed hardware overhead compared to previously proposed schemes.

1. Introduction

Built-In Self Test (BIST) schemes utilize a Test Pattern Generator to generate the test patterns that are applied to the inputs of the Circuit Under Test (CUT) [1]. In off-line BIST, the normal operation of the CUT is stalled in order to perform the test. Thus, if the CUT is critical for the function of the circuit, the performance is degraded. Concurrent on line Input vector monitoring concurrent BIST schemes [2] - [9] exploit vectors appearing to the inputs of the CUT during normal operation to perform concurrent on line testing. The measures utilized to evaluate an input vector monitoring concurrent BIST scheme are (a) the Concurrent Test Latency (CTL), i.e. the number of cycles that the CUT must operate in normal mode in order to expect that the concurrent test is complete and (b) the hardware overhead.

Sharma and Saluja [7] proposed the Built-In Concurrent Self Test (BICST) scheme illustrated in Figure 1. BICST is based on a pre-computed test set and monitors the vectors V arriving at the combinational CUT inputs and the respective outputs B. The Concurrent BIST Unit (CBU) compares the incoming vector against the pre-computed test set and, if the input vector belongs to the test set, compares the response of the CUT with the known good response; if the two vectors differ, an error is assumed to have happened in the CUT and the error signal is immediately activated.

In [9] a scheme was proposed that, based on the idea of selecting a subset of the input signals and utilizing them as inputs to a decoder-based structure, achieves a reduction in the hardware overhead comparatively to the scheme proposed in [7], while at the same time provides for off-line test capabilities. Almukhaizim *et al* investigated the problem of concurrent testing of sequential machines in [16]-[20]. Other concurrent online BIST schemes [11] - [14] utilize test patterns containing don't care values.

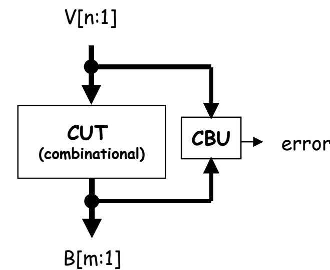

Figure 1: Input Vector Monitoring Concurrent BIST

In this paper we present a novel Input Vector Monitoring Concurrent On line BIST scheme (NEMO); the proposed scheme is based on the idea of combining a specially designed multilevel decoding structure at the input-side of the CBU with space compaction at the outputs of the Circuit Under Test; the proposed scheme is shown to be more efficient in terms of the hardware overhead compared to the schemes that have been proposed to date, i.e. [7] and [9].

The paper is organized as follows. In Section 2 previous works on input vector monitoring concurrent BIST based on a pre-computed test set, namely BICST [7] and MICSET [9] are described. In Section 3 the proposed scheme is presented, and its hardware overhead is calculated; in Section 4, the proposed scheme is compared to the ones proposed in [7], [9]. Finally, in Section 5 we conclude the paper. Figures 5, 9 and Table 3 are given in the end of the manuscript.

2. Previous work

The idea underlying BICST [7] is to monitor the CUT inputs under normal operation and whenever an input pattern is equal to one of the patterns in the pre-computed test set, compare the pre-stored good response to the actual response of the CUT; if it is found that the two responses differ then the CUT is declared as 'faulty'. For example, let us consider the well-known ISCAS'85 c17 benchmark [10]. A test set for the benchmark that consists of four test patterns is presented in Table 1 (labeled 'Input test matrix'), along with the expected responses (labeled 'Output response matrix').

978-3-9810801-8-6/DATE12/©2012 EDAA

	Input test matrix	Output response matrix
T1	11111	10
T2	10010	00
T3	00101	01
T4	01010	11

Table 1: Input and output test matrices for the c17 benchmark

The CBU of the BICST scheme (Figure 2) operates as follows: when an input vector of the test set (e.g. T3 = 00101) reaches the CUT inputs, the corresponding AND gate (denoted by A3 in Figure 2) is enabled (i.e. its output is set to '1'); therefore the output of the OR gates it drives (the OR gate denoted as 'O_2' in Figure 2) is also set to 1. The outputs of the remaining OR gates (i.e. O_1) are equal to 0. The outputs of the OR gates are compared to the B[1], B[2] outputs of the CUT. If one or more of the respective values differ, then the 'error' signal is issued.

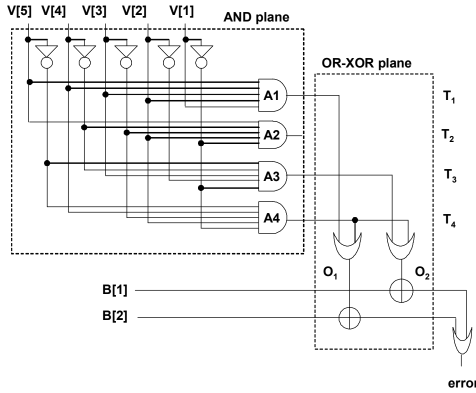

Figure 2: Implementation of the CBU of BICST for the c17 benchmark

MICSET [9] is based on the idea of selecting t, t<n of the input lines of the CUT and using them as inputs to a decoder that drives the AND-OR plane of the Concurrent BIST Unit (CBU) of MICSET. The CBU of the MICSET scheme for the c17 benchmark is presented in Figure 3. In Figure 3(a) an implementation comprising a Response Verifier (RV) is presented. When a new test pattern appears at the CUT inputs V[1:5], the respective cell (C1-C4) is set and the RV is enabled to capture the response of the CUT. When all cells have been set, the response captured in RV is examined to see whether an error exists in the CUT. In Figure 3(b) an alternative implementation is shown, where every time a test pattern appears to the CUT inputs, the CUT response is examined through the comparator denoted as cmp2, and if the response differs from the expected, an error signal is immediately issued.

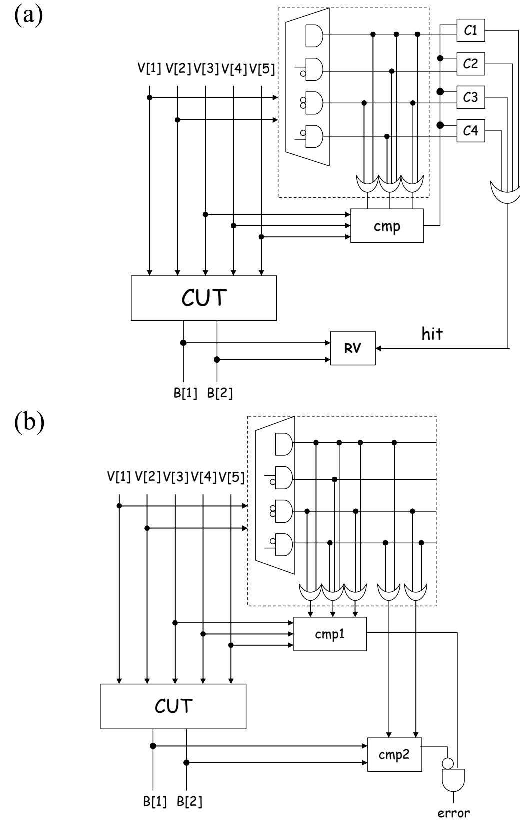

Figure 3: Implementation of the CBU of MICSET for the c17 benchmark using (a) Response Verifier and (b) comparison of the outputs with expected responses

3. Proposed scheme

In Figure 4 we present the general architecture of the proposed scheme. In Figure 4 the CUT is combinational, has n inputs, m outputs and can be tested with T test vectors, where $T<<2^n$. The normal inputs of the CUT V[n:1] are driven to a n-to-T decoding logic (D). The T outputs of the decoding logic are driven to two modules, denoted as Calc and an OR in Figure 4. The outputs of the CUT are driven to a Space Compactor (SC) that reduces the number of m to q, where q<<m. The output of the CALC module is a q-bit bus that is compared against the output of the SC. If these two q-bit vectors differ, and the input of the CUT belongs to the test set (this is determined by the output of the OR module, which is, in fact, a T-input OR gate), then the error signal is issued. Space Compactor implementations have been given in various works. Among them, the proposal of [21] results in extremely low hardware overhead and will be considered in our calculations.

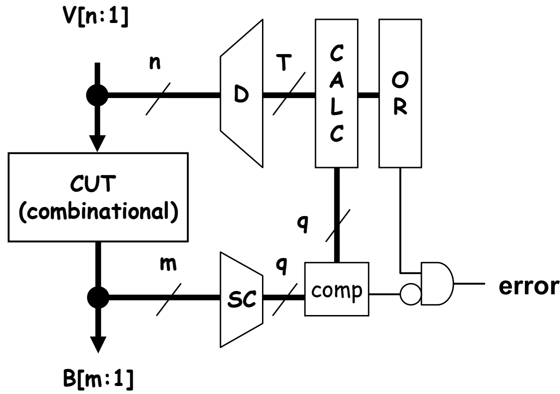

Figure 4: Architecture of NEMO

It is trivial to see that the general diagram of Figure 4 is similar to the one shown in Figure 3(b). The novelty of the proposed scheme lies in the design of the decoding structure; hence, in the sequel we shall present its implementation.

For the decoding structure utilized by the proposed scheme only a few of the possible outputs are needed; for example, for the case of the c17 benchmark, the number of inputs is 5; hence, the number of outputs of the respective decoder would be 2^5=32. However, out of these 32 outputs, only T = 4 << 32 outputs are required.

Therefore, we implement a multilevel decoding logic, as follows: first we separate the columns of the input test matrix in consecutive pairs. For each two-bit combination we utilize a two-input AND gate. Next, we separate the columns in consecutive groups of 4. For each group, we utilize a 2-input AND gate, whose inputs are the inputs of the first level AND gates that constitute the respective group of 4 bits. We continue the procedure until the number of groups equals the number of inputs of the CUT. This procedure is illustrated in Figure 5 (end of manuscript). S is the number of utilized gates.

The number of steps of the algorithm (and corresponding levels of 2-input AND gates) is $\lceil \log_2 n \rceil$; where n is the number of inputs of the CUT. The total number of the required 2-input AND gates is equal to the number of distinct t-tuples ($2 \leq t \leq n$) found during the application of the procedure. In order to illustrate the above procedure, we shall present the following examples.

Example 1: Let us consider a part (6 rows by 8 columns) of the test set for the c6288 benchmark provided by [15] shown in Table 2. The variable S is used to account for the number of required gates and is initially set to 0.

```
01010101
10101010
11011011
01101101
10110110
11111111
```

Table 2: Test set for Example 1

Step 1: We separate the columns in groups of 2. In every pair, we enumerate the 2-bit combinations. For every distinct combination we increase the gate count by 1. The corresponding module is shown in Figure 6 and the gate count is increased by 12. In Figure 6 we denote each gate with the input combination that activates its output.

Step 2: We separate the columns in groups of 4 and enumerate the combinations. For each such combination we increase S by 1. The corresponding module is shown in Figure 7 and the gate count is increased by 12.

Step 3: We separate the columns in groups of 8; we enumerate the combinations and increase S accordingly (in the specific example, this will be the number of lines in the initial table). The corresponding module is shown in Figure 8 and the gate count is increased by 6.

It is trivial to see that every one of the outputs denoted '01010101' to '11111111' in Figure 8, is enabled when the respective combination of input values appears to the inputs I1-I8.

Example 2: Let us consider the test set for the c17 benchmark presented in Table 1. We will proceed in a similar way as follows. We separate the columns in pairs as shown in Figure 9(a) (end of manuscript). Note that, since the number of inputs is not a multiple of 2, the remaining input is treated as a pair for itself. The remaining steps are illustrated in Figures 9(b)-(c).

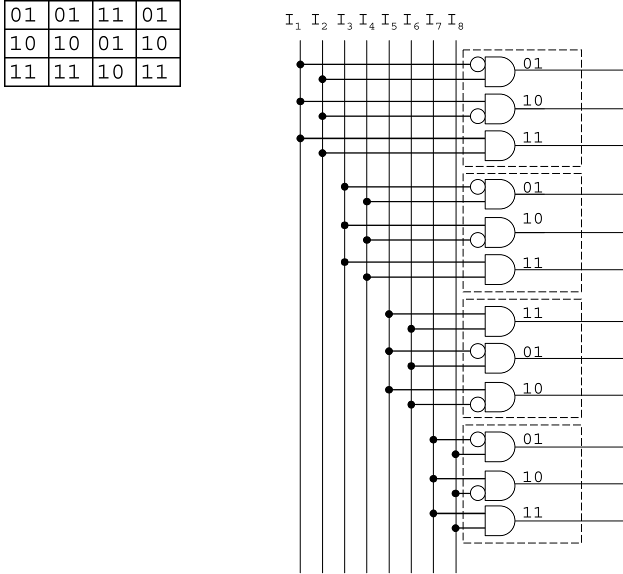

Figure 6: First step of the construction of the decoding structure for Example 1

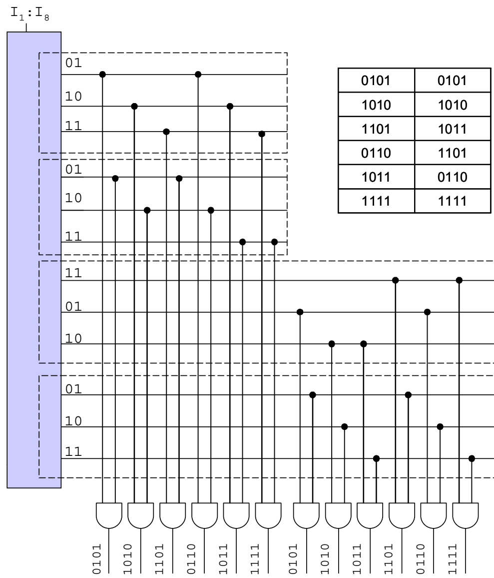

Figure 7: Second step of the construction of the decoding structure

4. Comparisons

In order to compare NEMO to previously proposed schemes, we shall use the hardware overhead as a metric since, the three competing schemes (i.e. the ones proposed in [7], [9] and in this work) have the same concurrent test latency for the same test set. As mentioned, the hardware overhead of the scheme consists of the following constituents (Figure 4):

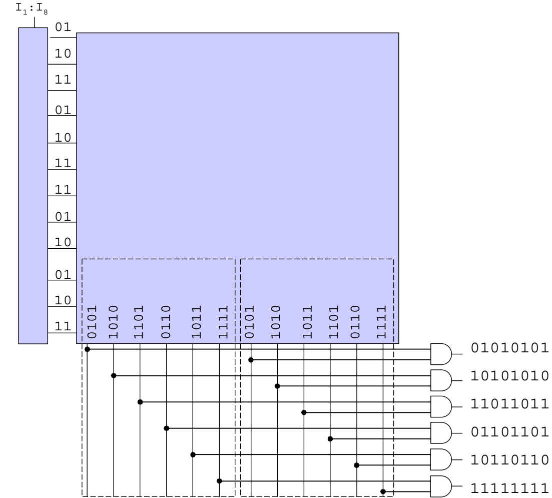

Figure 8: Third step of the construction of the decoding structure

- n-to-T decoding structure whose design was described in the previous Section
- m-to-q Space Compactor
- T-input/q-output CALC module that calculates the q-bit output for each of the T test patterns of the test set. The CALC module is practically a series of q OR gates, whose input is at most T.
- T-input OR that decides whether the incoming vector belongs to the test set.
- q-bit comparator
- error indication 2-input AND gate

For the calculations of the hardware overhead of the proposed scheme we utilized the test sets provided by [15]. For the Space Compactor we have utilized the data provided in [21, Table VII].

The hardware overhead is presented in Table 3 (end of manuscript), using the gate equivalents as a metric, along with the hardware overhead of the schemes proposed in [7] and [9]. With respect to the data presented in Table 3, it should be noted that, in order to make fair comparisons we have utilized the same test set for the three benchmarks. Hence, the data in Table 3 differ from the respective ones given in [9]. Furthermore, since the scheme proposed in [9] provides for both on line and off-line testing, the hardware components that participate in the off-line testing are not included.

In Table 3, in the first column we present the ISCAS benchmark; in the second through the fourth column we present the number of inputs (n) and outputs (m), as well as the number of patterns in the test set (T). In the fifth column we present the hardware overhead of the BICST scheme proposed in [7]; in the sixth column we present

the hardware overhead of the MICSET scheme using a response verifier at the output of the CUT (as illustrated in Figure 3(a); in the seventh column we present the hardware overhead of the MICSET when the outputs of the CUT are compared with the expected outputs, as shown in Figure 3(b); in the eight column we present the hardware overhead when the outputs of the CUT are compressed using a space compactor and compared with the compressed responses, in a way similar to the one proposed in this work for the NEMO architecture. In the ninth column we present the hardware overhead of the proposed scheme. In the tenth column (*%red*) we present the reduction of the hardware that is achieved by the proposed scheme with respect to the MICSET scheme (under the heading 'Comparator + SC'). We can see that a considerable reduction (73% on average) is achieved. The purpose of the column '%red' is to illustrate that the reduction in hardware overhead is mainly due to the proposed novel design of the decoding logic and not to the utilization of the Space Compaction circuitry at the output of the module. In Figure 10, the data of Table 3 are graphically illustrated.

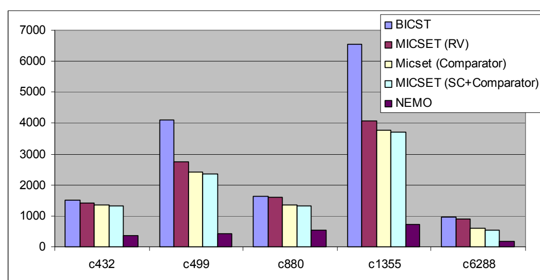

Figure 10: Input vector monitoring concurrent BIST schemes hardware overhead: comparison

5. Conclusions

Concurrent on line Input vector monitoring concurrent BIST schemes exploit vectors appearing to the inputs of the CUT during normal operation to perform concurrent on line testing. The measures utilized to evaluate an input vector monitoring concurrent BIST scheme are the Concurrent Test Latency and the hardware overhead. In this work we have presented an input vector monitoring concurrent BIST scheme that, utilizing a novel design for the decoding logic, achieves a significant reduction in hardware (73% on average for the examined benchmarks) to the techniques that have been proposed previously for the same purpose with the same concurrent test latency.

Acknowledgments

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Archimedes III. Investing in knowledge society through the European Social Fund.

References

[1] M. Abramovici, Breuer M., Freidman A., "Digital Systems Testing and Testable Design", Computer science Press, 1990.

[2] K.K. Saluja, R. Sharma, and C. R. Kime, *"A Concurrent Testing Technique for Digital Circuits"*, IEEE Trans. on Computer-Aided Design, Vol.7, No.12, pp.1259, December, 1988.

[3] K.K. Saluja, R. Sharma, and C. R. Kime, *"Concurrent Comparative Testing Using BIST Resources"*, International Conference on Computer Aided Design, pp.336-339, Nov. 1987.

[4] K.K. Saluja, R. Sharma, and C. R. Kime, *"Concurrent Comparative Built-In Testing of Digital Circuits"*, Tech. Rep. ECE-8711, Dep. of Elect. Comp. Eng. Univ. of Wisconsin, 1986.

[5] Voyiatzis I., Halatsis C., "A low-cost Concurrent BIST scheme for increased Dependability", IEEE Transactions on Dependable and Secure Computing, vol. 2, No 2, April 2005, pp. 150-156.

[6] I. Voyiatzis, Paschalis A., Gizopoulos D., Kranitis N., Halatsis C., "A Concurrent BIST Scheme Based on a Self-Testing RAM", IEEE Transactions on Reliability, Vol. 54, No. 1, March 2005, pp. 69-78.

[7] R. Sharma, K. Saluja, "Theory, Analysis and Implementation of an On-line BIST technique", VLSI Design, 1993, vol. 1, no. 1, pp. 9-22.

[8] I. Voyiatzis, Paschalis A., Nikolos D., Halatsis C., "R-CBIST: An Efficient Concurrent BIST Technique", IEEE International Test Conference, 1998.

[9] I. Voyiatzis, A. Paschalis, D. Gizopoulos, C. Halatsis, E. Makri, M. Hatzimihail, "An Input Vector Monitoring Concurrent BIST Architecture Based on a Pre-computed Test Set", IEEE Transactions on Computers, Volume 57, Issue 8, Aug. 2008, pp.1012 - 1022.

[10] F. Brglez, H. Fujiwara, "A neutral netlist of 10 combinational benchmarks circuits and a target translator in FORTRAN", International Symposium on Circuits and Systems, 1985.

[11] I. Voyiatzis, D. Gizopoulos, A. Paschalis, "A Concurrent BIST scheme exploiting don't care values", accepted for publication in the 16th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SOC), 2008.

[12] I. Voyiatzis, Gizopoulos D., Paschalis A., "An Input Vector Monitoring Concurrent BIST scheme Exploiting "X" values", in the IEEE International On-Line Test Symposium, 2009.

[13] M. A. Kochte, C. Zoellin, H.-J. Wunderlich: Concurrent Self-Test with Partially Specified Patterns for Low Test Latency and Overhead", European Test Symposium 2009, pp. 53-58.

[14] M. Kochte, C. Zoellin, H.-J. Wunderlich, "Efficient Concurrent Self-Test with Partially Specified Patterns", Journal of Electronic Testing: Theory and Applications (JETTA), 2010, doi: 10.1007/s10836-010-5167-6.

[15] M. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse

Engineering," IEEE Design and Test, vol. 16, no. 3, pp. 72-80, July-Sept. 1999, also available from http://www.eecs.umich.edu/~jhayes/

[16] S. Almukhaizim, Y. Makris, “Concurrent Error Detection Methods for Asynchronous Burst Mode Machines,” IEEE Transactions on Computers (T. COMP), vol. 56, no. 6, pp. 785-798, 2007.

[17] S. Almukhaizim, P. Drineas, Y. Makris, “Entropy-Driven Parity Tree Selection for Low-Cost Concurrent Error Detection,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (T. CAD), vol. 25, no. 8, pp. 1547-1554, 2006.

[18] S. Almukhaizim, P. Drineas, Y. Makris, “Compaction-Based Concurrent Error Detection for Digital Circuits,” Microelectronics Journal (MJ), Elsevier, vol. 36, no. 9, pp. 856-862, 2005.

[19] S. Almukhaizim, Y. Makris, “Concurrent Error Detection in Asynchronous Burst-Mode Controllers,” Proceedings of the IEEE Design Automation and Test in Europe Conference (DATE), pp. 1272-1277, 2005.

[20] S. Almukhaizim, P. Drineas, Y. Makris, “On Concurrent Error Detection with Bounded Latency in FSMs,” Proceedings of the IEEE Design Automation and Test in Europe Conference (DATE), pp. 596-601, 2004.

[21] S. Biswas, Das S.R., Petriu E.M., “Space Compactor Design in VLSI Circuits based on Graph Theoretic Concepts”, IEEE Transactions on Instrumentation and Measurement, vol. 55, no. 4, August 2006.

```
0.        k = 2, S=0
1.        Separate the N columns into consecutive pairs and correspond each pair of columns to
          a pair of inputs of the CUT.
2.        For each consecutive group of 2 columns (Ci, Ci-1) that correspond to inputs Ini,Ini-1
          of the CUT:
          2.1   Identify the (at most 4) different 2-tuples that appear to the pair of columns
          2.2   For each 2-tuple from Step 2.1, increase S by one and add a 2 input AND gate as
                follows:
                2.2.1 a gate with both inputs inverted (for the combination 00)
                2.2.2 a gate with the first input inverted (for the combination 10)
                2.2.3 a gate with the second input inverted (for the combination 01)
                2.2.4 a gate with non input inverted (for the combination 11)
          2.3   Connect all the first outputs of all modules in 2.2.1-2.2.4 to input Ini of the
                CUT and the second outputs to input Ini-1 of the CUT.
3. if k>=N then END else k=k*2
4. Separate the n columns into consecutive groups of k.
          4.1   For each group of k columns identify the different combinations of k bits that
                appear in the k columns (at most 2^k).
          4.2   For each combination in 4.1 increase S by 1 and connect the respective signals
                to the inputs of a 2-input AND gate.
5. Go to 3.
```

Figure 5: Algorithm for the construction of the decoding structure

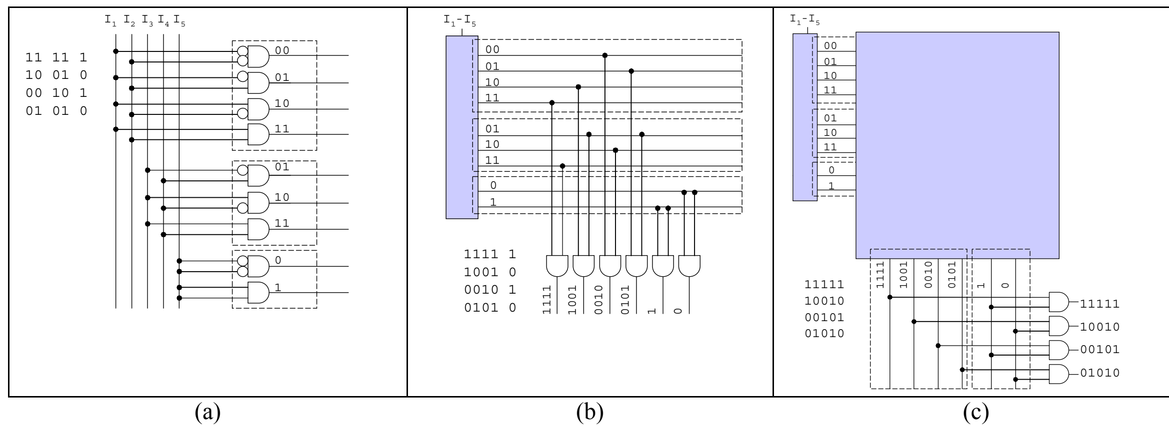

Figure 9: Construction of decoding logic for Example 2

					MICSET [9]				
Benchmark	**n**	**m**	**T**	**BICST [7]**	**RV**	**Comparator**	**SC + Comparator**	**NEMO (proposed)**	**% red**
c432	36	7	32	1507	1413	1343	1335	347	74%
c499	41	32	52	4112	2738	2418	2364	410	83%
c880	60	26	17	1643	1609	1349	1314	537	59%
c1355	41	32	84	6544	4077	3757	3711	723	81%
c6288	32	32	12	964	910	590	538	170	68%

Table 3: Input vector monitoring concurrent BIST schemes hardware overhead: comparison