Column-Matching Based BIST Design Method

Petr Fišer, Jan Hlavička

Department of Computer Science and Engineering

Czech Technical University

Karlovo nám. 13, 121 35 Prague 2

e-mail: fiserp@fel.cvut.cz, hlavicka@fel.cvut.cz

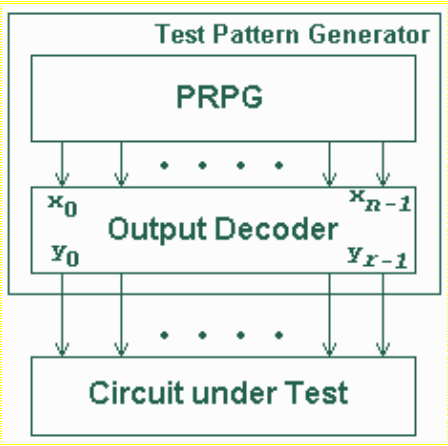
- A new **test-per-clock** BIST method
- For **combinational circuits** only

Perquisites

- The fault coverage is determined by the test patterns
- Test patterns are precomputed by an ATPG tool

Principle of the method

PRPG patterns are transformed into the required test patterns be some combinational logic – **the output decoder**



The Inputs

- A set of patterns generated by a PRPG (LFSR, CA)
- A set of required test patterns

Our Task

Design the output decoder converting these two sets as small as possible

The problem: Transformation of matrices

C matrix: code words produced by a PRPG, dimensions (*n*, *p*)

n – the number of the PRPG bits (LFSR stages)

p – the number of PRPG patterns (cycles)

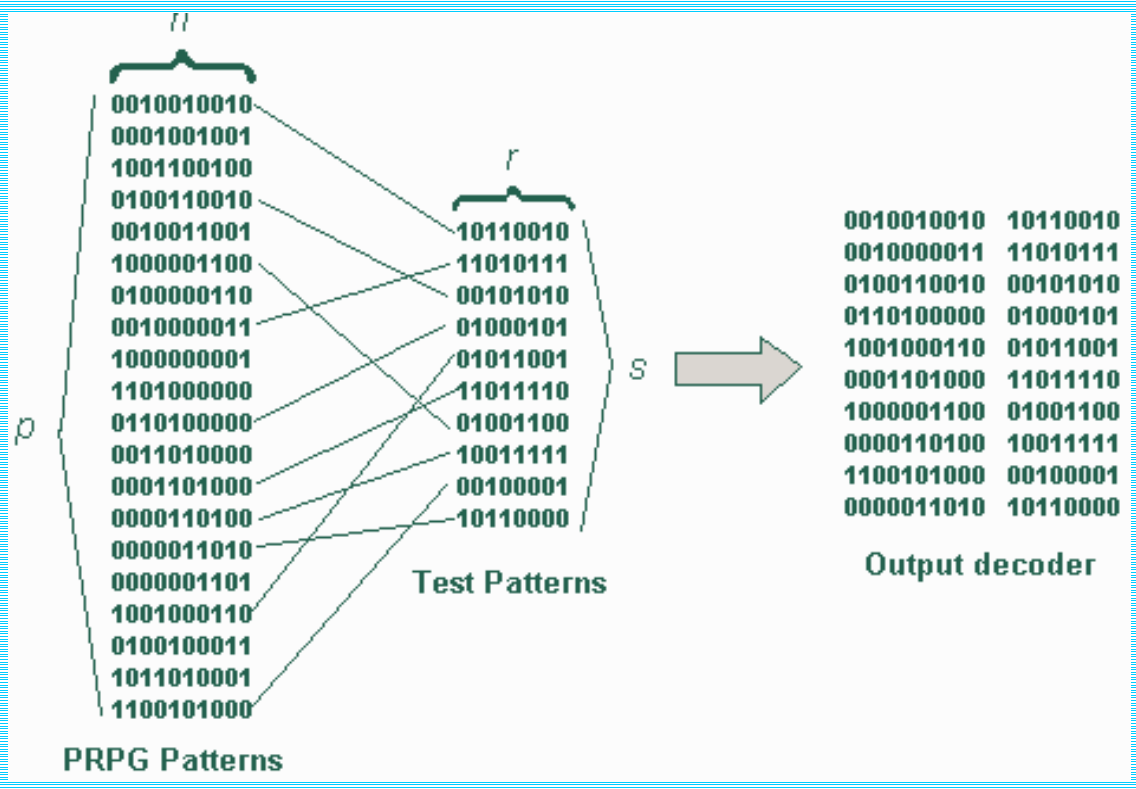
T matrix: test patterns produced by an ATPG, dimensions (*r*, *s*)

r – the number of the primary inputs of the CUT

s – the length of the test

Important facts:

- When testing combinational circuits, the order of test patterns generated by an ATPG tool is insignificant ⇒ the patterns can be **reordered** in any way
- Any vector (row) from the **T** matrix can be assigned to any vector of the **C** matrix
- The rows in the **C** matrix **need not** form a compact block (idle cycles can be present)



- All the **rows** of the **T** matrix have to be assigned to the rows of the **C** matrix
- Unassigned **C** matrix rows represent idle PRPG cycles – make no harm
- After the assignment the output decoder is described by a **truth table**
- **Boolean minimization** is applied to the truth table – ESPRESSO or BOOM

The Column Matching Principle

Main Idea:

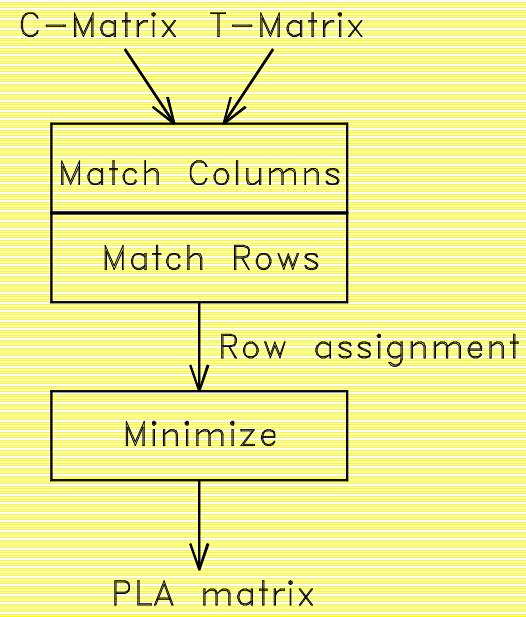
- If in the final assignment *i*-th column of the matrix **C** is exactly the same as *j*-th column of the matrix **T**, there is no combinational logic required to implement *j*-th variable in hardware

⇒ **Matching of the columns**

Also the negative matching is possible (using negated outputs of the flip-flops)

The principle:

1. Find as many column matches as possible
2. Match the rows (using some row matching method)
3. Construct the remaining logic using Boolean minimization (PLA matrix)



The Column Matching Algorithm

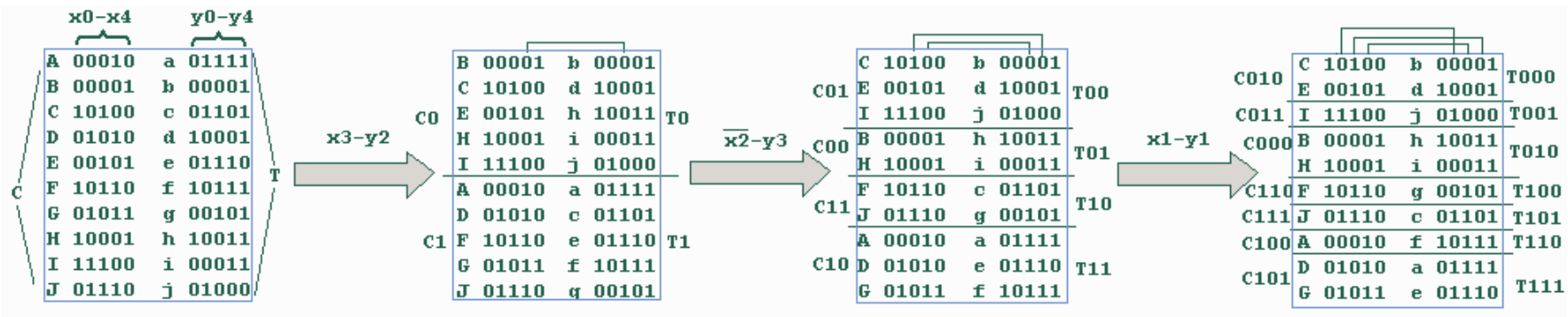
For *p* = *s* (one-to-one matching):

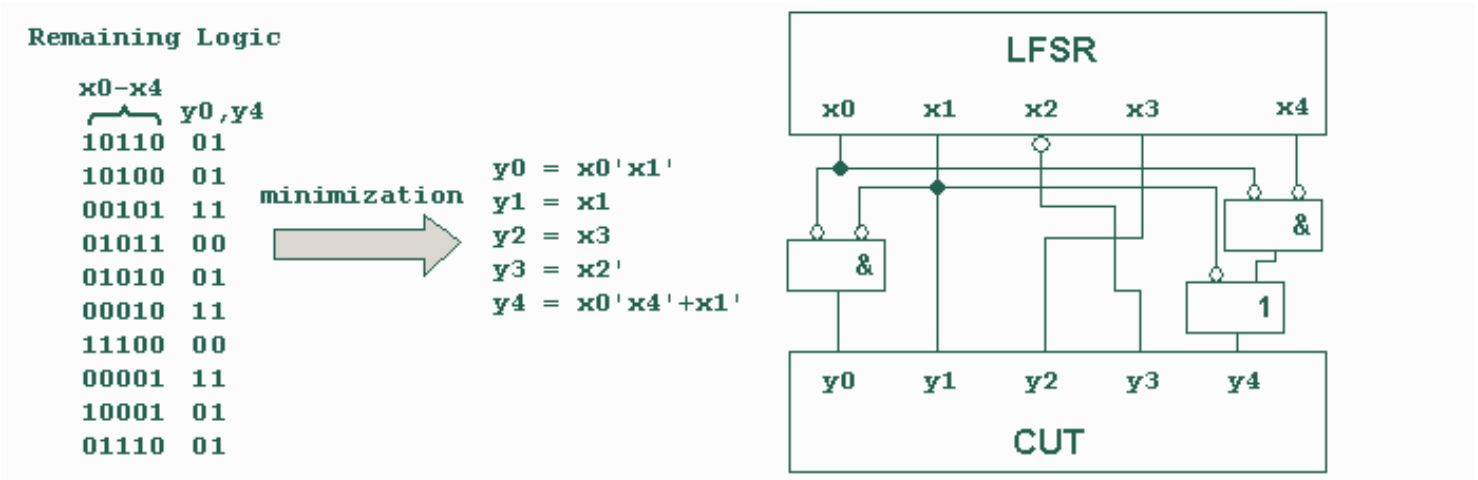
- The column match is possible if the counts of ones (and zeros) in the corresponding columns are **equal**.
- After finding one column match, the two matrices are decomposed into two disjoint parts: the rows with zeros and ones respectively in the corresponding columns, let them be denoted as $\mathbf{C_0}$, $\mathbf{C_1}$ and $\mathbf{T_0}$, $\mathbf{T_1}$. Then any vector from the sub-matrix $\mathbf{T_0}$ can be assigned to any vector from $\mathbf{C_0}$, and any vector from the sub-matrix $\mathbf{T_1}$ can be assigned to any vector from $\mathbf{C_1}$, but not otherwise.
- The successive decomposition of both matrices into systems of subsets is performed, until no decomposition is possible

Generalized column matching (*p* > *s*):

- Similar to one-to-one matching
- The number of vectors in each $\mathbf{C_i}$ must be **greater or equal** to the number of vectors in the corresponding $\mathbf{T_i}$

An Illustrative Example





ISCAS Benchmarks

- Test patterns computed by ATOM tool (100% fault coverage)
- An LFSR with *r* stages seeded with a random vector was used as a pseudo-random pattern generator

benchmark	LFSR (*n* / *p*)	test size (*r* / *s*)	matches	cost
c1355	41 x 5000	41 x 192	8	1475
c1908	33 x 5000	33 x 210	10	2043
c432	36 x 5000	36 x 100	10	1180
c499	41 x 5000	41 x 127	9	698
c880	60 x 5000	60 x 133	10	3024

Conclusions

- A new test-per-clock BIST method for combinational circuits was described
- The pseudorandom patterns are generated by a PRPG and then transformed by a combinational block into given test patterns
- It is based on the column matching approach, where as many outputs as possible are directly matched to the inputs

Acknowledgment

This research was in part supported by grant #102/01/0566 Built-in Self-Test Equipment Optimization Methods in Integrated Circuits of the Czech Grant Agency (GACR).