The Use of Genetic Algorithm to Reduce Power Consumption during Test Application

Jaroslav Skarvada, Zdenek Kotasek, Josef Strnadel[1],
[1] Brno University of Technology, Faculty of Information Technology,
Božetěchova 2, 61266 Brno, Czech Republic
{kotasek,skarvada,strnadel}@fit.vutbr.cz

Abstract. In this paper it is demonstrated how two issues from the area of testing electronic components can be merged and solved by means of a genetic algorithm. The two issues are the ordering of test vectors and scan registers with the goal of reducing switching activity during test application and power consumption as a consequence of the ordering. The principles of developing an optimizing procedure with the aim of achieving a solution satisfying the required value of power consumption during power consumption are described here. A basic description of the methodology together with the functions needed to implement the procedures is provided. Experimental results are also discussed.

Keywords: test application, power consumption, optimizing procedure, fitness function, genotype, phenotype.

1 Introduction

With the continuing increase in chip density, power dissipation has become one of the major design constraints for today's VLSI circuits. Although there are many techniques for power minimization during normal (functional) operation, power minimization during testing is an emerging research area because power dissipation during testing is becoming a yield and reliability problem. Significantly more *switching activity* occurs during testing than during functional operation. The increased activity can decrease the reliability of the circuit under testing because it causes excessive temperature and current density which can cause problems in circuits designed with a power minimization requirement. Furthermore, as a result of high activity in circuits employing BIST, the voltage drop that occurs only during testing causes some good circuits to fail the testing process, leading to unnecessary manufacturing yield loss.

To summarize, excessive switching activity during scan testing can cause *average power dissipation* and *peak power* during testing to be much higher than during a normal operation. This can cause problems both with heat dissipation and with current spikes.

Various formulas to evaluate power consumption were developed and implemented [1], [2]. They are difficult to be used in practical designs, especially for complex circuits and a great volume of input data (test vectors).

1.1 Power Consumption Metrics

For the purposes of comparing various optimizing procedures that are aimed at power consumption reduction, power consumption metrics were developed and are used. It is evident that if the sequence of input data is reorganized as a result of applying a particular methodology and the implementation of the component is unchanged, then for the purposes of comparing various methodologies, an *NTC* (*Number of Transition Count*) parameter can be used. More precise techniques are based on the use of *WNTC* (*Weighted Number of Transition Count*) [3], and *WSA* (*Weighted Switching Activity*) [4]. These parameters can be evaluated by the following formulas:

$$NTC = \sum_{i=1}^{N_C} n(i) \tag{1}$$

In (1), $n(i)$ represents the number of 0↔1 transitions between two states in $i{-}1/f$, i/f instants, N_c is the total number of clock pulses applied during test application.

$$WNTC = \sum_{i=1}^{N_C} \sum_{j=1}^{N_G} n_j(i) F_j \tag{2}$$

In (2), the meaning of $n_j(i)$ and N_c is the same as in (1), F_j is the fan out factor of node j, N_G is the total number of nodes in the component.

$$WSA = \sum_{i=1}^{N_C} \sum_{j=1}^{N_G} n_j(i) C_j \tag{3}$$

In (3), C_j is normalized node capacity, while the meaning of other symbols is the same as in (1) and (2).

1.2 Low Power Approaches

Two approaches for *low power testing* exist: the first ones are directed to *reducing dynamic portion* of power consumption (*switching power*), while the second group of methodologies have a goal of reducing its *static portion* (*leakage power*). It is important to say that in older implementations, dynamic portion of power consumption was higher than the static one – for example, in [30], it is reported that the dynamic portion of power consumption is about 90% of the total power consumption. As a consequence, in 90 nm technology [7] the dynamic portion of power consumption is only 58% of total power consumption (according to [8], 65 nm technology is seen as the technology in which the static power consumption begins to prevail over the dynamic one). It is even more evident in technologies with higher level of integration (32 nm, 25 nm) in which the static power consumption is much higher than the dynamic one [9]. Thus, to choose proper and effective optimizing procedures to decrease power consumption, the information about the target technology to which the design will be implemented, becomes significant. In this paper, attention is paid to the reduction of the dynamic portion of power consumption.

1. 3 Complexity of the Problem

Modern commercial tools are able to generate high quality sets of *test vectors* with a high degree of *fault coverage* which are not usually optimized to reducing power consumption. Therefore, various methods were developed to optimize the sequences of test vectors to reduce switching activities during test application. In *combinational circuits* the responses depend only on the set of test vectors being applied; therefore, it is possible to reorganize their sequence. The responses will be the same, but their sequence will be different. It means that the sequence of test vectors can be reorganized with the goal of minimizing power consumption. It can also be stated that fault coverage is the same as with the original sequence of test vectors generated by the test generator. In *sequential circuits* the situation is different due to the fact that the responses do not actually depend on the applied test vector but on those applied in previous steps as well. The test is generated by an *SATPG* (*Sequential Automated Test Pattern Generator*). If the sequence is modified in some way, then a completely different test will be gained with a different fault of coverage. If a scan register chain is inserted into the component then for the test generation process the component will be seen as combinational and *ATPG* (*Automated Test Pattern Generator*) can be used to generate the test. Fault coverage does not depend on the sequence of test vectors. The sequence of scan registers can be reorganized to reduce power consumption.

The problem of identifying the proper sequence of test vectors/scan registers belongs to the category of *NP-hard* problems [5], its time complexity is $O(n) = n!$, where n is the number of elements the sequence of which is supposed to be optimized. To model both problems (i.e. the sequence of scan vectors and scan registers) separate graph models are often used. To solve the problem, a *minimal Hamiltonian path* must be identified in the graph. After it is found, it represents the solution of the problem, (i.e. the sequence of test vectors) for which the power consumption during test application is minimal is identified. Many methods exist which utilize the above described approach. For example, in [7], Hamming distance between test vectors is analyzed in order to optimize their sequence.

1.4 Problems Related to Power Dissipation Estimation

It can be concluded that power consumption during the test application of test vectors is in some way associated with Hamming distance between test vectors. Anyway, examples can be found in which the results do not correlate. The switching activity is difficult to evaluate if the physical implementation of the component is not known. It can be shown that a change in one bit can cause higher switching activity than a change in several other bits (more than one bit). In [6], the problem of reordering scan registers in the scan chain is solved – a greedy search algorithm is used for this purpose. Methods combining *BPIC* (*Best Primary Input Change time*) approaches with test vectors reordering can achieve even higher reduction of power consumption. In [3], the method combining these two approaches is described – it uses simulated annealing to investigate state space. These methods require a special approach for test application which reduces their use in commercial diagnostic tools. Typically, optimizing methods are used sequentially (e.g. the sequence of registers in scan chains is optimized first, and then the same is done for the sequence of test vectors).

2 Motivation for the Research

We analyzed the state of the art of existing methodologies with the aim of optimizing power dissipation during test application. To solve this issue, it is necessary to reorganize the sequence of test vectors and the sequence of scan registers. **The drawbacks of previously published methodologies can be summarized in the following way: 1) in previous approaches these two issues (the reorganization of test vectors sequence and the sequence of scan registers) were solved separately; 2) the results of various methodologies are not evaluated on platforms to which they will be later implemented.**

The previously published approaches have test vectors as the only input data to the methodology without any information about the internal structure of the component under testing through which test vectors will propagate. The propagation of test vectors through the structure represents additional switching activity which can have a significant impact on power dissipation. Most methodologies are based on the evaluation of Hamming distance between input test vectors without any coupling with an implementation platform. As a result, the impact of test vectors reorganization on power dissipation through switching activity reduction is rather difficult to be precisely evaluated. We also see that both procedures (i. e. test vectors reorganization and scan chain reorganization) are performed in sequence as two separate procedures in previous methodologies. Our approach is based on concurrent optimization of both procedures. For this purpose a *genetic algorithm* (*GA*) was used.

3 Proposed Optimization Method

For purposes of the methodology, a formal model was developed. It is based on the theory of sets. The model reflects *structural* (primary interface of *CUA – Circuit Under Analysis*, elements in CUA, the ports of these elements, connections existing in CUA), *diagnostic* (topology of scan chains, the list of test vectors and the sequence of applying), and *electric* (switching model, power consumption during switching) properties of CUA. Algorithms were developed which operate on the formal model.

As already mentioned, GA was used to find the solution of the problem defined in this paper. In each step, candidate solutions are recognized (*phenotypes*) and encoded into *genotypes* which carry genetic information. Genetic operators are applied on genotypes. All solutions must satisfy the required quality. Therefore, principles of evaluating quality of individual solutions must be defined.

The quality evaluation is performed in several steps. First, the genotype is transformed into phenotype. The quality of the particular phenotype is reflected by a real number, where a special function is defined for this purpose. The principle of problem encoding allows one to encode both partial problems (the sequence of test vectors and scan registers order) into one structure. The structure is scalable and can encode this information for several CUAs and/or for CUAs containing several scan chains. This principle was used in the methodology, the goal of which is the identification of testable blocks in CUA [10].

3.1 Encoding of the Problem

When GA is used to find a solution of a particular problem, encoding plays an important role. An incorrect problem encoding can possibly bring about a GA malfunction. The goal of a GA application is to gain candidate solutions of certain (i.e. required) quality, so principles of quantitative evaluation of individual solutions must be clearly defined. The quality evaluation is performed in several steps: 1) the conversion of genotype to phenotype (for this *purpose function* Δ was defined); and 2) the *fitness function* Φ is used to evaluate phenotype quality by means of a real number – for details, see section 3.2.

The principle of problem encoding that was applied here allowed us to encode the representation of both tasks to genotype: the ordering of test vectors together with the scan registers ordering into scan chain. The encoding is based on partitioning *chromosome* CH = (b_{i1}, b_{i2}, …, b_{il}) into separate blocks, each block encodes the ordering of test vectors during test application or the ordering of scan registers into a scan chain. The number of blocks in the chromosome is equal to the sum of CUAs and the number of scan chains. Each block consists of one or more genes.

As an example of the encoding, let a *two-block chromosome* (b_{i1}, …, b_{ik-1}, b_{ik}, …, b_{il}) be presented now which is typical for circuits with one scan chain. The first block, represented by the (b_{i1}, …, b_{ik-1}) sequence, reflects the ordering of test vectors to be applied to CUA; while the second block, represented by the (b_{ik}, …, b_{il}) sequence, reflects the ordering of scan registers in the scan chain.

The values in blocks are encoded independently. A system of priorities is used in which the gen reflects the priority of either a test vector or a scan register. It must be possible to compare gens; therefore, comparison operators must be defined. By means of an ascending reordering of code sequences (performed separately in each block) we gain the order of entities in each block (the ordering of test vectors and the ordering of scan registers). To demonstrate the mechanism, see the examples in Tables 1 and 2. – the two-block chromosome with 3 test vectors v_1, v_2, v_3 and 3 scan registers sc_1, sc_2, sc_3 is expected here.

Table 1. Impact of Chromosome Block Sequence on Test Vector Ordering.

Ordered code sequences	Test vectors ordering
$b_{i1} \leq b_{i2} \leq b_{i3}$	v_3 after v_2 after v_1
$b_{i2} \leq b_{i3} \leq b_{i1}$	v_1 after v_3 after v_2
$b_{i1} \leq b_{i3} \leq b_{i2}$	v_2 after v_3 after v_1
$b_{i3} \leq b_{i1} \leq b_{i2}$	v_2 after v_1 after v_3
$b_{i3} \leq b_{i2} \leq b_{i1}$	v_1 after v_2 after v_3
$b_{i2} \leq b_{i1} \leq b_{i3}$	v_3 after v_1 after v_2

In Table 1**,** all the alternatives of ordering code sequences b_{i1}, b_{i2}, b_{i3} together with corresponding ordering of test vectors for the chromosome are demonstrated. The b_{i1} value determines the order of applying v_1 vector while b_{i2} value determines the order of applying v_2, a similar relation is found between b_{i3} and the order of applying v_3. A vector corresponding to a lower value of code sequence will be applied before a vector with a higher value of code sequence. For $b_{i2} \leq b_{i3} \leq b_{i1}$, v_2 will be applied first while v_1 will be the last applied vector.

Table 2. Impact of Chromosome Block Sequence on Scan Register Ordering.

Ordered code sequences	**Scan registers ordering**
$b_{i4} \leq b_{i5} \leq b_{i6}$	sc_3 after sc_2 after sc_1
$b_{i5} \leq b_{i6} \leq b_{i4}$	sc_1 after sc_3 after sc_2
$b_{i4} \leq b_{i6} \leq b_{i5}$	sc_2 after sc_3 after sc_1
$b_{i6} \leq b_{i4} \leq b_{i5}$	sc_2 after sc_1 after sc_3
$b_{i6} \leq b_{i5} \leq b_{i4}$	sc_1 after sc_2 after sc_3
$b_{i5} \leq b_{i4} \leq b_{i6}$	sc_3 after sc_1 after sc_2

In Table 2, all alternatives of ordering code sequences b_{i4}, b_{i5} , b_{i6} together with corresponding ordering of scan chain for the chromosome are shown. The ordering reflects the values of b_{i4}, b_{i5}, b_{i6}, the same mechanism as applied for test vectors is used to identify the sequence of scan registers (the comparison of b_{i4}, b_{i5} , b_{i6} values). For a more detailed example to the encoding, see the following section (3.2).

3.2 Fitness Function

The value of *fitness* is evaluated by *Φ* function which is designed to transform a *genotype* to *phenotype*.

```
Algorithm Φ(TVS,SRS,CH,K,M)
01 Δ(CH,K,TA,SCS)
02 return 1/pwr(TVS,TA,SRS,SCS,SVS,M)
```

The phenotype reflects the test vector sequence and organization of registers within scan chains. A particular solution is assigned a fitness value proportional to particular power savings during the test application time. Power consumption related to a solution is estimated during the test application simulation phase.

At the input, Φ takes the following data: *TVS* – test vector sequence, *SRS* – scan register sequence, *SVS* – scan vectors set, *CH* – chromosome, *K* – sequence used to divide CH into blocks, *M* – metric utilized for power consumption evaluation, M ∈ {NTC, WNTC, WSA}). Φ returns a real number from <0; 1> interval representing the fitness of CH. Φ works as follows: first (see line 01 of Φ's code) where CH is transformed into separated *TA* (i.e., test vector application sequence) and *SCS* (ordering of registers in scan chains) sequences – Δ function is utilized for this purpose. Both TA and SCS values are needed for the succeeding phase (line 02) in which test process application is simulated – as a result, power consumption is quantified by the power consumption metric M. *Fitness* value is evaluated as a reciprocal of the value produced at the output of the simulation and after the evaluation, it is returned as the output of Φ.

```
Algorithm Δ(CH,K, TA,SCS)
01 TA := ()           ; TA init
02 SCS := ()          ; SCS init
03 KK := K            ; save K into KK
04 k1 := 0            ; the starting index of the actual block in CH
05 k2 := 1            ; the starting index of the next block in CH (|CH|=l)
06 if len(KK)≠0 then ; at least one scan chain exists in CH
07    k2 := car KK      ; prepare it for processing
```

```
08
09 KK := cdr KK         ; remove the first block from KK
10 TA := sort(subset(CH, k1, k2))        ; test application sequence block
11
12 While len(KK)≠0 do
13    k1 := k2          ; actualize block pointers k1, k2
14    k2 := car KK      ; prepare index of the next block
15    KK := cdr KK      ; remove actual block from KK
16    SCS push sort(subset(CH, k1, k2)) ; add the result to SCS's end
17
18 if len(K)≠0
19    k1 := k2          ; actualize block pointers k1, k2
20    k2 := l           ; k2 is the last one
21    SCS push sort(subset(CH, k1, k2)) ; add the result to SCS's end
```

The Δ function works as follows: after the initialization phase (rows 01 – 09), the first block is processed and the sequence of indexes representing particular test vectors is produced in row 10. In rows 12 to 16, the second to penultimate blocks are processed. If there are more than two blocks within CH, the last block is processed in rows 18 to 21. It should be noted that: 1) *len* returns the length of a given sequence; 2) *car* returns the first element within a given sequence; 3) *cdr* returns a given sequence, except the *car*; 4) *subset* returns the sequence of indexes (k_1, …, k_2-1)∈CH; and 5) *sort* sorts a given sequence of indexes in an ascending order. An example: suppose CH = (12, 2, 8, 10, 20, 11, 5, 9) and K = (3, 6); thus, CH is composed of 3 blocks:

- **Block_#1** starts at index 0 and is completed at 2 (i.e., (*car* K)-1=3-1=2) of CH. In (12, 2, 8), it encodes an application sequence of 2-0+1=3 test vectors (v_1, v_2, v_3). The succeeding blocks describe a way in which registers are organized in chains.
- **Block_#2** starts at 3 and is completed at 5 (i.e., (*car* (*cdr* K))-1=6-1=5) of CH. In (10, 20, 11), it encodes the organization of 5-3+1=3 registers in the first scan chain ($sc_{1,1}$, $sc_{1,2}$, $sc_{1,3}$).
- **Block_#3** starts at 6 and is completed at 7 (i.e., *len*(CH)-1) of CH. In (5, 9), it encodes the organization of 7-6+1=2 registers in the 2nd scan chain ($sc_{2,1}$, $sc_{2,2}$).

Then Δ(CH, K, TA, SCS) implies

- TA = (2, 3, 1), that is to say test vectors will be applied in the order given by their indexes: (v_2, v_3, v_1). The result was achieved by the following process: the smallest value within the Block1 is 2, placed at index 1. This corresponds to vector v_{1+1}=v_2. Thus, v_2 will be applied as the first one. The following higher number within Block1 is 8, placed at index 2. So, vector v_{2+1}=v_3 will be applied as the next one. 12 is the highest number within the block. It is placed at index 0, so v_{0+1}=v_1 will be applied as the last one.
- SCS = ((1, 3, 2), (1, 2)), i.e., the ordering of registers in the first scan chain will be ($sc_{1,1}$, $sc_{1,3}$, $sc_{1,2}$), while in the second scan chain it will be ($sc_{2,1}$, $sc_{2,2}$).

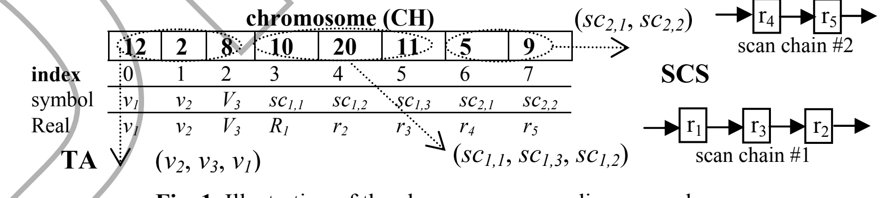
chromosome (CH)
12 2 8 10 20 11 5 9
index 0 1 2 3 4 5 6 7
symbol v_1 v_2 V_3 $sc_{1,1}$ $sc_{1,2}$ $sc_{1,3}$ $sc_{2,1}$ $sc_{2,2}$
Real v_1 v_2 V_3 R_1 r_2 r_3 r_4 r_5
TA
(v_2, v_3, v_1)
($sc_{1,1}$, $sc_{1,3}$, $sc_{1,2}$)
($sc_{2,1}$, $sc_{2,2}$)
SCS
r_4 r_5
scan chain #2
r_1 r_3 r_2
scan chain #1

Fig. 1. Illustration of the chromosome encoding example

3.3 Selection Operators

In this section, two selection operators utilized in the methodology for crossover purposes are described in detail: *roulette-wheel* and *tournament*. If the roulette-wheel selection is applied, probability of selecting an individual (pCHi ∈ <0; 1>) is specified by the formula

$$pCH_i = phi(TVS,SRS,CH_i,K) \; / \; \Sigma^{\lambda}_{j=1} \; phi(TVS,SRS,CH_j,K). \qquad (4)$$

In tournament selection case, k_1 individuals are selected for a tournament competition. As a result, $k_2 \leq k_1$ individuals are selected according to predefined probability p. At the output, set (O) of k_2 individuals is produced, $|O| = k_2$.

3.4 Initialization of the Population

In general, there are two ways how the *initial population* of individuals can be created: 1) in a random way or 2) in an intelligent way (non-random way). In the second case, the population can be generated by means of data gained from a design tool which is able to produce both a test vector sequence and an ordering of registers in scan chains – though the sequences are not optimal from the power consumption point of view. The sequences can be transformed into chromosome form by means of Δ^{-1} function (because of limited space of the paper, its description is omitted). At least one chromosome should be generated in this way while the others can be generated randomly. If the elitism is activated, it is guaranteed that the best solution found by the method will not be worse than the solution produced by the tool.

4 Experimental results

Unless stated otherwise, experiments presented as a result of our research were performed on a PC equipped with two AMD Opteron 2220 dual core CPUs operating at 2.8 GHz.

Table 3. Relation between optimization type and search space size for a b15 circuit.

Optimization	**b15 solution space size (|TVS|=1297, |SC|=416)**	
	Formula	**Enumeration**
Test vectors ordering only	*len*(*TV S*)!	$1{,}44 \times 10^{3476}$
Organization of scan chains only	|*SC*|!	$3{,}84 \times 10^{910}$
Both in sequence	*len*(*TV S*)! + |*SC*|!	$\approx 1{,}44 \times 10^{3476}$
Both in parallel	*len*(*TV S*)! × |*SC*|!	$5{,}54 \times 10^{4386}$

4.1 Problem Size

In Table 3, search space size analysis is summarized for various optimizations related to a *b15* circuit from an ITC99 benchmark set. In the first column, the type of optimization is seen. In the second column, a general formula (i.e., for any circuit) that can be utilized to evaluate search space size corresponding to the optimization is presented as a function of parameters. In the last column enumeration for a b15 is presented. It is evident that the procedure is the most time consuming if both optimizations are performed in parallel.

Table 4. Times needed to explore b15 search space in an exhaustive way.

Number of test vectors	Number of scan registers	Exploration time
8	3	17,9 minutes
9	3	2,7 hours
10	3	26,8 hours
12	3	147,4 days
15	3	1102,5 years
1297	416	4,71 × 10^{4374} years

In Table 4, times needed to explore complete search space corresponding to various numbers of test vectors and scan registers are summarized. Values presented in the first three rows of the table were measured while values in the other rows were identified after extrapolation of data presented in Table 3. It is evident that search space cannot be explored in a reasonable time (Table 4).

4.2 Impact of GA Parameters

During the experiments, the impact of generic algorithm parameters on both the quality of a produced solution and convergence speed were also investigated. For the experiments, a *b02* circuit from ITC99 benchmark set was used, the results of which are summarized below.

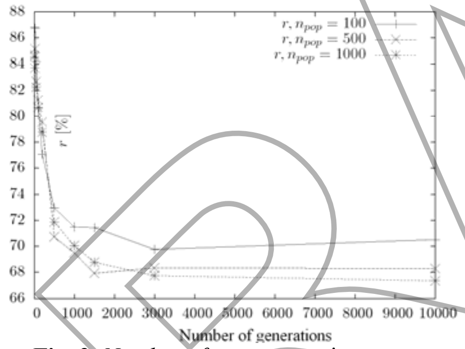

Fig. 2. *Number of generations* impact

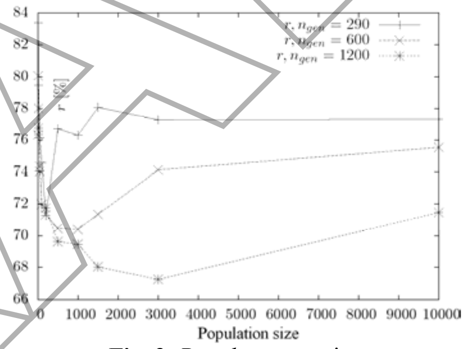

Fig. 3. *Population size* impact

In Fig. 2 (as in Fig. 3), average reduction values gained over 10 GA runs are presented on the vertical axis for various numbers of generations (population sizes) utilized during the runs. At the top, the impact of constant population size (100, 500, and 1000) is depicted in Fig. 2. It is evident that GA is able to converge relatively fast, so high quality results (i.e., those with small *r*) can be produced during the first several hundreds of generations using a relatively small population size. Similarly in Fig. 3, the impact of constant number of generations (290, 600, and 1200) is shown. It can be seen that the reduction grows with population size – after few oscillations, the value becomes stable if a bigger population size (e.g., 3000 or more) is utilized. Also, it can be seen the relation depends on the number of generations utilized.

4.3 Scalability of the Solution

For the described experiment below, a computational system composed of two 4-core Intel Xeon X5355 CPUs (i.e., 2x4 = 8 CPUs in total) running on 2,66 GHz was utilized. The main goal of the experiment was to verify experimentally the scalability of the solved task on a real multiprocessor system. Execution times, speedups and overheads related to the multiprocessor environment are demonstrated in Fig. 4 and Fig. 5. In Fig. 4, the execution time and the speedup are shown as a function of CPUs within the multiprocessor environment, while corresponding overhead is presented in Fig. 5. Because execution times related to actions (download of dynamic libraries, circuit verification, generation of look-up tables utilized during simulation, initial simulation, etc.) are included in the overhead, it is evident that pure communication overhead will be less or equal to the presented values.

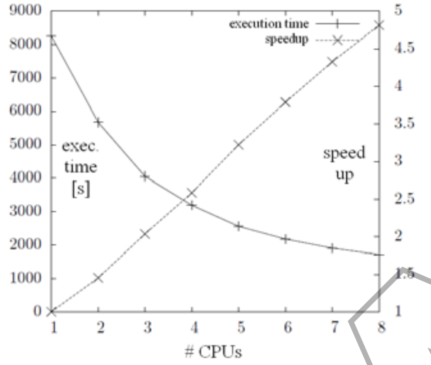

Fig. 4. Speedup

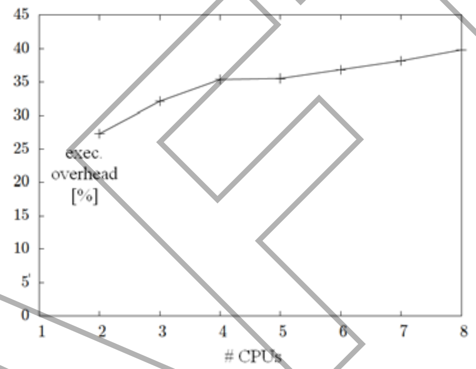

Fig. 5. Parallel execution overhead

4.4 Comparison with Other Approaches

In the paragraph, experimental results gained by our approach are compared with results of other published methods. It should be noted that comparison in an objective way is difficult because parameters of the methods differ a lot – e.g., circuits analyzed by the methods are mapped onto various platforms, various test pattern generators with different settings are used by the methods or the methods differ in the way they summarize achieved results. Moreover, some data were not available for some methods, so it was impossible to guarantee the equality of input conditions to experiments. In all experiments related to our optimization method, the circuits were mapped onto *AMI 0.5um library* by means of *Leonardo Spectrum* tool. While a test vector set is the only input to the optimizing procedure for combinational circuits, organization of registers in scan chains must also be taken into account if sequential circuits are processed (the circuits were modified to their *full scan* versions by *DFTAdvisor* tool; for simplification, one scan chain was utilized). Test vectors under the stuck-at-fault model were generated by *Flextest* tool. For each of the methods, mean values of the best results attained over 20 GA runs are presented.

In Fig. 6, results gained for a subset of ITC99 benchmarks are presented and compared to method A [11], method B [12] and method C [13]. It is evident in all cases that power consumption was reduced most by our method than by the others.

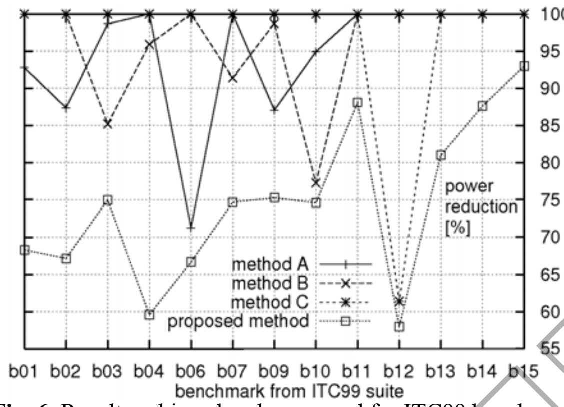

Fig. 6. Results achieved and compared for ITC99 benchmarks

In Fig. 7, results gained for a subset of ISCAS85/89 benchmarks are presented and compared to method A [14], method B [13] and method C [6]. It can be seen then (except in s27, s298, s641, s1488, c7552 circuits) that power consumption achieved by our method was reduced more than by the others.

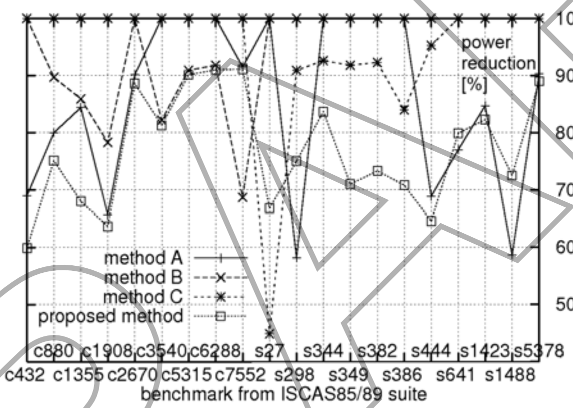

Fig. 7. Results achieved and compared for ISCAS58/89 benchmarks

5 Conclusions

In our research we analyzed methods which were used in modern approaches having power consumption reduction as their goal. It was recognized that all previous approaches were based on a separate analysis of test vectors and scan chain sequences. Based on this finding, the methodology merging these two possibilities together was defined, developed and implemented. It was also decided to verify the results on implementation platform instead of by means of Hamming distance between input test vectors.

Valuable experimental results were gained which indicate that our approach is better than previous methodologies.

Acknowledgements. This work was supported by the Grant Agency of the Czech Republic (GACR) No. *102/09/1668 – SoC circuits reliability and availability improvement*, by Research Project *MSM 0021630528 – Security-Oriented Research in Information Technology* and the grant *BUT FIT-S-10-1*.

References

1. Raghunathan, A.; Jha, N. K.; Dey, S.: High-Level Power Analysis and Optimization, Boston. ISBN 0-7923-8073-8, 175 pp. Kluwer Academic Publishers (1998).
2. Roy, K.; Prasad, S. C.: Low-Power CMOS VLSI Circuit Design. ISBN 0-471-11488-X, 359 pp. Wiley-Interscience publication (2000).
3. Nicolici, N.; Al-Hashimi, B. M.: Power-Constrained Testing of VLSI Circuits. ISBN 1-4020-7235-X, 178 pp. Kluwer Academic Publishers, (2003).
4. Debjyoti, G.; Swarup, B.; Kaushik, R.: Multiple Scan Chain Design Technique for Power Reduction during Test Application in BIST. In 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 191- 198 (2003).
5. Dabholkar, V., Chakravarty, S., Pomeranz, I., et al.: Techniques for Minimizing Power Dissipation in Scan and Combinational Circuits During Test Application. IEEE Trans. on Computer-Aided Design of Integrated Circuits, Vol. 17, No. 12, pp. 1325-1333 (1998).
6. Chakravarty, S., Dabholkar, V.: Minimizing Power Dissipation in Scan Circuits During Test Application. In: Proceedings of International Workshop on Low-Power Design (1994).
7. Pangrle, B., Kapoor, S.: Leakage power at 90nm and below [on-line]. EE Times Asia. Src: http://www.eetasia.com/ARTICLES/2005JUN/B/2005JUN01_POW_EDA_TA.pdf (2005).
8. Thompson, S., Packan, P., Bohr, M.: MOS Scaling: Transistor Challenges for the 21st Century. Intel Technology Journal, Vol. 19 (1998).
9. Marongiu, A. et al.: Analysis of Power Management Strategies for a Large-Scale SoC Platform in 65nm Technology. In Proceedings of the 11th Euromicro Conference on Digital System Designing Architectures, Methods and Tools, pp. 259-266 (2008).
10. Vranken, H., Waayers, T., Fleury, H., Aj.: Enhanced Reduced-Pin-Count Test For Full-Scan Design. In Proceedings of IEEE International Test Conference, pp. 738-747 (2001).
11. Almukhaizim, S., Makris. Y., Yang Y.-S., Veneris, A.: Seamless Integration of SER in Rewiring-Based Design Space Exploration. In: Proceedings of International Test Conference, pp. 1-9 (2006).
12. Babighian, P., Kamhi, G., Vardi, M.: PowerQuest: Trace Driven Data Mining for Power Optimization. In: Proceedings of Design, Automation & Test in Europe Conference & Exhibition, pp. 1-6 (2007).
13. Girard, P., Landrault, C., Pravossoudovitch, S.: Reducing Power Consumption During Test Application by Test Vector Ordering. In Proceedings of the IEEE International Symposium on Circuits and Systems, IEEE Computer Society, pp. 296-299 (1998).
14. Jelodar, M. S., Aavani, A.: Reducing Scan Base Testing Power Using Genetic Algorithm. In Proc. of 11th Iranian Computer Engineering Conference, Vol. 2, pp. 308-312 (2006).