International Journal of Computer and Communication Technology

Volume 4 | Issue 1 Article 7

January 2013

Soft Computing Approach To Automatic Test Pattern Generation For Sequential Vlsi Circuit

Monalisa Mohanty
Lecturer,DRIEMS,Cuttack, mohanty_monalisa@yahoo.co.in

S. N. Patnaik
Asst.Professor, ECE Department, DRIEMS, Cuttack, India, patnaiksn@gmail.com

Follow this and additional works at: https://www.interscience.in/ijcct

Recommended Citation

Mohanty, Monalisa and Patnaik, S. N. (2013) "Soft Computing Approach To Automatic Test Pattern Generation For Sequential Vlsi Circuit," *International Journal of Computer and Communication Technology*: Vol. 4 : Iss. 1 , Article 7.
DOI: 10.47893/IJCCT.2013.1167
Available at: https://www.interscience.in/ijcct/vol4/iss1/7

This Article is brought to you for free and open access by the Interscience Journals at Interscience Research Network. It has been accepted for inclusion in International Journal of Computer and Communication Technology by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

Soft Computing Approach To Automatic Test Pattern Generation For Sequential Vlsi Circuit

Monalisa Mohanty [1], S.N.Patanaik[2]
[1] Lecturer,DRIEMS,Cuttack, [2] Prof.,HOD,ENTC, DRIEMS,Cuttack
[1]mohanty_monalisa@yahoo.co.in, [2]patnaiksn@gmail.com

Abstract -Due to the constant development in the integrated circuits, the automatic test pattern generation problem become more vital for sequential vlsi circuits in these days. Also testing of integrating circuits and systems has become a difficult problem. In this paper we have discussed the problem of the automatic test sequence generation using particle swarm optimization(PSO) and technique for structure optimization of a deterministic test pattern generator using genetic algorithm(GA).

Keywords: test pattern generator, design, particle swarm optimization, genetic algorithm, optimization.

1. INTRODUCTION

In digital integrated circuits, especially in sequential circuits, the automatic test pattern generation(ATPG) can be realized by using deterministic and simulated-based algorithms. Generally the deterministic algorithms are time-wasting because a lots of backtracking is required. On the other hand simulated based algorithms does not require any backtracking but they have lower fault coverage because of lacking structure information.

To improve the test efficiency and decrease the test time, we are going for PSO-based ATPG. The whole test generation algorithm includes four parts: initialization, test generator, fault simulator and test set compaction. Firstly initializes sequential circuit using PSO; secondly generates the test sequence of the target fault using PSO test generator, thirdly the test sequence simulates all other faults and deletes the detected faults and finally compacts the test set.
Further structure optimization of a deterministic test pattern generator (TPG) is achieved by genetic algorithm(GA) approach. The TPG is composed of a linear register and a non-linear combinational function that can invert any bit in the generated patterns. Consequently, any arbitrary test sequence can be produced. This type of TPG is suitable for on-line built-in self-test (BIST) implementations where a set of deterministic test patterns is required. In order to reduce the no. of gates in the BIST structure, a genetic algorithm (GA) is employed.

2. THE ATPG ALGORITHM

Here the problem of automatic test pattern generation for sequential circuits described at gate level and we are using the single stuck-at fault model. The ATPG algorithm based on PSO includes the following five phases:

Phasel : initialize the object sequential circuit using PSO algorithm;

Phase2: select a fault from fault table of to be detected sequential circuit as the target fault;

Phase3: generate a test sequence for target fault using PSO algorithm;

Phase4: simulate the other all faults using the generated test sequence;

Phase5: compact the test set.
The flow chart of the test generation process is described in Fig. 1.

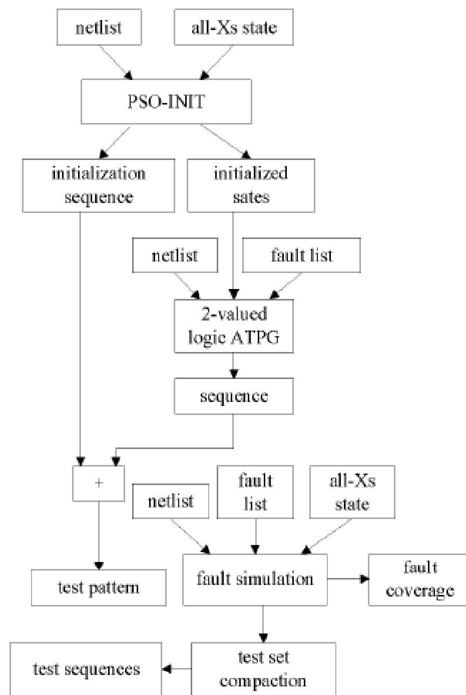


Figure 1. ATPG flow based on PSO

As the inputs, outputs and all signal lines are discrete values that may be zero or one, we use the two-binary coded discrete PSO. Its evolving equations can be described as follows:

$V_{ij}(t+1)=V_{ij}(t) + rand(p_{ij}(t)-x_{ij}(t)) + rand(g_{ij}(t)-x_{ij}(t))$ ……(1)

$Sig(V_{ij}(t+1))=1/(1+exp(-V_{ij}(t)))$ …….(2)

$X_{ij}(t+1)=\{0, rand \geq Sig(V_{ij}(t+1))$
1, others ……..(3)

In above equations $X_i(x_{i1},x_{i2},..., x_{in})$ is present position of the *ith* particle. $V_i(v_{i1} v_{i2}... ,V_{in})$ is actual flying velocity of the *ith* particle, $P_i(p_{i1}, p_{i2}, , p_{in})$ is the best position that the *ith* particle has experienced, $G_i(g_{i1}, g_{i2}, ... g_{in})$ is the best position that all particles have experienced in population, *rand* is a random between zero and one, *t* expresses the *tth* iteration.

PHASE 1: Initializing the Object Sequential Circuit

For sequential circuits, initialization can be achieved through the use of a global reset signal which is connected to all state elements, otherwise generating an initialization sequence that is able to initialize all the flip-flops to a known value.

The initialization methods can be classified into two kinds: functional initialization and logical initialization. In functional initialization there exists a sequence that brings the circuit to a known state no matter the initial state. The logical initialization is simulating circuit with a 3-valued logic simulator that is able to correctly compute the known final state starting from an all-X state.

The goal is initializing all flip-flops in objective sequential circuit. When the particle flies to the best position, there will have flip-flops initialized. Given an individual, the fitness function computes its closeness to the goal and should guide the PSO toward those regions of the search space where more likely the optimum solution is located. The fitness function is defined as the number of flip-flops that a sequence can initialize i.e. the following expression:

fitness (i, j) = num $(x_{i,j})$*- num* $(gbest_{j-1})$ *……(4)*

Where *fitness (i, j)* is the number of flip-flops that the *ith* particle of *jth* iteration can initialize by itself; $num(x_{ij})$ is the number of flip-flops that have determined state when looking the *ith* particle of *jth* iteration as the
input; $num(gbest_{j-1})$ is the number of flip-flops that have certain state when looking the best particle of *jth* iteration as the input.

PHASE 2. Selecting a Target Fault

The goal of this phase is selecting a fault from fault table as target fault. The initial sequence population is randomly generated. Assuming the population size is N . The length of initial sequence is equal to the minimum of flip-flops in some route multiplied to the number of objective circuit
inputs. We simulate all faults of fault table using every sequence of the initial sequence population.For each fault, we have an evaluation function. The total number of
sequences that can activate the fault is defined as the evaluation function, which is namely the following expression:

fitness $(f_i)=\sum_{s=1}^{N} g_s(\alpha_s, f_i)$ *…….(5)*

Where f_i denotes fault and α_s denotes sequence. When α_s can excite fi I the function value of $g_s(\alpha_s,f_i)$ is 1, or else its function value is 0. We select the fault that has the maximum evaluation function as the target fault, and we add all sequences that can activate the target fault into set A.

PHASE 3. The Test Sequence Generation

The goal of the phase is to generate a test sequence for the target fault. We use the PSO algorithm to modify the population and generate new individuals. Each sequence is an individual and the initial population is composed of Num sequences of A that is generated in last phase. Only the target fault is considered in this phase. The evaluation function is associated to each sequence and evaluates how close each individual is to the final test sequence that can detect the target fault.

The output responses of propagation gates and flip-flops in fault free circuit and in fault circuit are different. So the evaluation function is defined as below.

$$H(s_j, f_i) = max(h (v_k^j, f_i)) \quad \ldots (6)$$

$$h(v_k^j, f_i) = c_1 \sum_{p=1}^{n_1} w_p p_d d_p (v_j^k, f_i) + c_2 \sum_{m=1}^{n_2} w_m d_m (v_j^k, f_i) \quad \ldots (7)$$

Where function $H(s_j, f_i)$ is associated with each sequence S_j ; v_k^j is the *kth* input vector of sequence S_j. n_1 and n_2 are the number of gates and flip-flops respectively. w_p is the weight of the *pth* gate, defined as the maximum number of gates on any path between the *pth* gate and a PO or PPO. The function $d_p (v_k^j, f_i)$ returns 1(0) if the value of the *pth* gate is different (equal) in the good and in the faulty circuit for fault f*i* . W_m is the weight of the *mth* flip-flop, defined as the maximum number of flip-flops on any acyclic path between the *mth* flip-flop and a PO. The function $d_m(v_k^j, f_i)$ returns 1 (0) if the value of the *mth* flip-flop is different (equal) in the good and in the faulty circuit for fault f_i. C_1 and C_2 are arbitrary constants.

PHASE 4. Fault Simulation

This phase determines whether the test pattern generated in the third phase can detects other faults of fault table.

Fault simulation has mainly three functions in test generation. The first is guiding the test pattern generation (TPG) process; the second is measuring the effectiveness of the test patterns; the last is generating fault dictionaries. So the phase can reduce the CPU time requirement and improve the test efficiency greatly.

PHASE 5. Test Set Compaction

Test set compaction is also called minimizing the test set .It decreases the number of the test patterns. Compacting test set is very important for reducing the cost of testing the large scale digital circuits by shortening the test application time. Small test set also reduces the test storage requirements. So test set compaction has very much importance to quicken the test process and to cut down the cost of testing.

If a test set can detect all faults of a given circuit, we call the test set as a complete test set of the circuit. In all complete test sets, that has the fewest patterns is called minimal complete test set. Any fault can be detected by a pattern of the minimal complete test set at least, and at any rate there a fault that other patterns cannot detect in all the faults that can be detected by a pattern of the minimal complete test set. In fact, test set compaction is to find the minimal complete test set of a given circuit.

3. GENETIC ALGORITHM FOR STRUCTURE OPTIMIZATION

We used GA optimization because of its intrinsic parallelism that allows working from a broad database of solutions in the search space simultaneously. Thus, the risk of converging to a local optimum is relatively low.

3.1 Encoding

The parameters of the TPG to be optimized were coded as integer values into three different chromosomes. With those three chromosomes we concurrently optimized the structure of the TPG, the order of the test patterns, and the bit order of test patterns. The first chromosome, which encodes the structure of n-bit TPG, looks like

$C_1 = t_1 i_1 t_2 i_2 \ldots t_n i_n$ ....(1)

where t_j (j = 1, 2, . . . , n) represents the type of the flipflop (either D or T) and i_j (j = 1, 2, . . . , n) represents the presence of the inverter on the input of the *j* -th flip-flop.

The second and third chromosome, which encode the order of the test patterns, and the bit order of test patterns, look like

$C_2 = a_1 a_2 \ldots a_m$ .... (2)

where m is the number of test vectors and a_j (j =1, 2, . . . ,m) is the label number of the test pattern from the pattern list, and

$C_3 = b_1 b_2 \ldots b_k$ .... (3)

where k is the number of flip-flops in the structure and b_j (j = 1, 2, . . . , k) is the label number of the bit order of test patterns.

3.2 Initial population

The initial population consisted of n chromosomes reproductions of the initial structure. To ensure versatile population some chromosomes were mirrored. The values on the left side (beginning) of the chromosome were mirrored to the right side (ending), while the values from the right side were mirrored to the left side; either type of registers or inverter presence or both values were mirrored in case of the first chromosome type. In case of other two chromosomes, their initial reproduction included

mirroring of orders between the beginning and the ending positions.

3.3 Genetic operators

In the selection process most fit chromosomes were selected for reproduction. In a two-point crossover chromosome mates were chosen randomly and with a probability p_c all values between two randomly chosen positions were swapped which led to the two new solutions. For example, considering two strings with crossover points on positions 1 and 4 see Fig. 6a. In the first chromosome, register type and inverter presence are considered as one indivisible block(ie, two values for one position in the chromosome).

Moreover, with some probability pr only the values of inverters in that swapping range were swapped. See Fig. 6b.The crossover in case of test patterns order and bit order of the test patterns was performed with the interchange of positions that store the ordered numbers within the range; for example within the range [2, 4] see Fig. 6c (positions with orders 3, 2, and 4 in the first chromosome are interchanged with orders 2, 4, and 3 of the second chromosome).

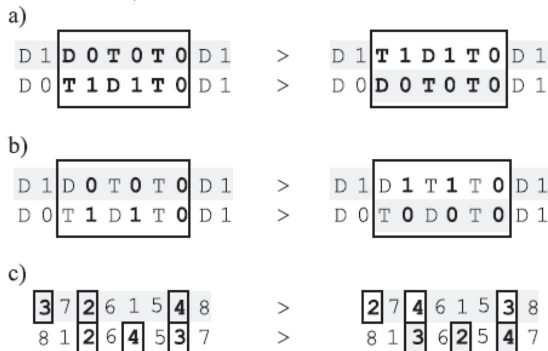


Fig. 6. Crossover operator: a) register type and inverter presence as one indivisible block, b) only the values of inverters are swapped, and c) interchange of positions that store the ordered numbers.

In the mutation process each value of the string mutated with a probability p_m. However, since a high mutation rate resulted in a random walk through the GA search space, p_m had to be chosen to be somewhat low. Three different types of mutation were applied (see Fig. 7):

- D/T-type change, where only flip-flop types were changed with some probability on each position in the chromosome (Fig. 7a);
- inverter change, where inverter presences were changed with some probability on each position in the chromosome (Fig. 7b);
- order change, where pattern orders and test bit streams order were changed after choosing the positions to be modified their values are interchanged (Fig. 7c).

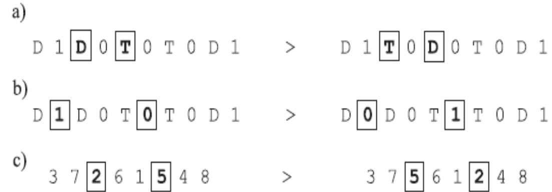


Fig. 7. Mutation operator: a) only flip-flop types are changed, b) inverter presences are changed, and c) pattern orders and test bit streams order are changed.

3.4 Fitness evaluation

After the recombination operators modified the solutions, the whole new population was ready to be evaluated. Here, the external evaluation tool (see Section 4.6) was used to evaluate each new string created by the GA.

3.5 Termination criteria

In our implementation the GA operated repetitively. When a certain number of populations had been generated and evaluated, the system was assumed to be in a non-converging state, the fittest member within all generations was taken to be the solution of the design problem.

3.6. EVALUATION TOOL

Operation of the *j*- th cell of the TPG register during one clock cycle can be expressed by the following equation:

$$Q_j = t_j q_j \oplus q_{j-1} \oplus i_j \oplus f_j$$
$$Q_1 = t_1 q_1 \oplus q_n \oplus i_1 \oplus f_1 \qquad \text{……..(4)}$$

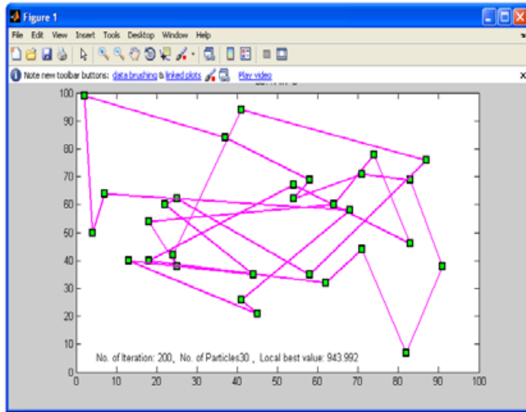
where q_{j-1} is the current state of the cell number $j-1$, q_j is the current state of the j -th cell, Q_j is the next state of the *j* -th cell, t_j is the coefficient determining type of the flip-flop in the *j* -th cell, ie, 0 for D-type flip-flop, and 1 for T-type flip-flop, ij is the coefficient determining whether there is an inverter at the input of the flip-flop in the *j* -th cell, ie, 0 for absence of inverter, and 1 for presence of inverter, and f_j is the value of the *j* -th output of the modification logic. Thus, the value of the *j* -th output of the modification logic is:

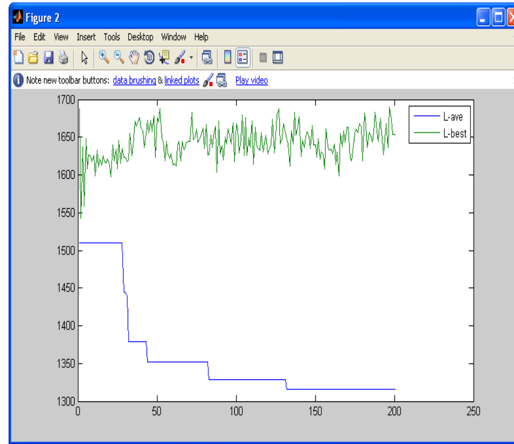
$$f_j = t_j q_j \oplus q_{j-1} \oplus i_j \oplus Q_j$$
$$f_1 = t_1 q_1 \oplus q_n \oplus i_1 \oplus Q_1 \qquad \text{……………(5)}$$

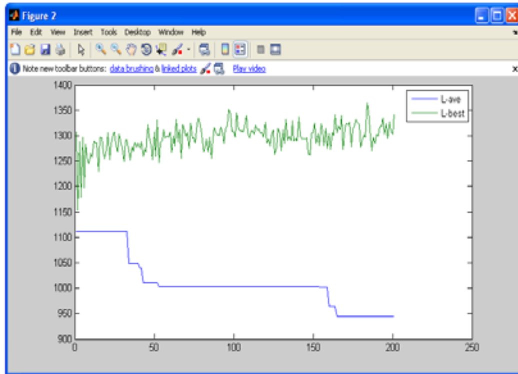
On the basis of these equations one can derive values of the outputs of the modification logic for each vector but last in the test sequence. In that way ON-set and OFF-set of the modification logic are defined.

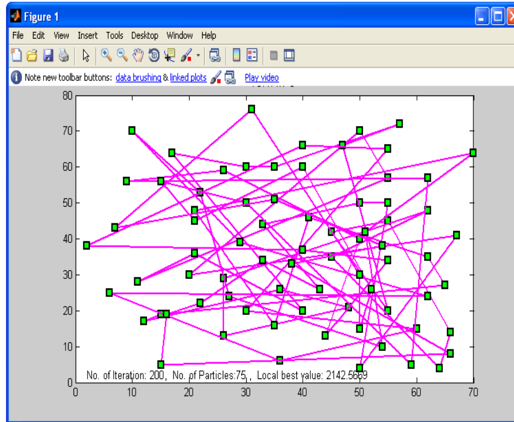
4. EXPERIMENTAL RESULTS

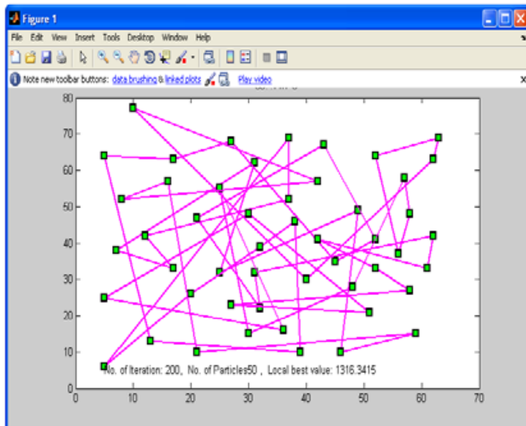
We implemented the test generation based on PSO algorithm containing all the techniques described above. We simulate some sequential circuits in MATLAB. The results are shown below:

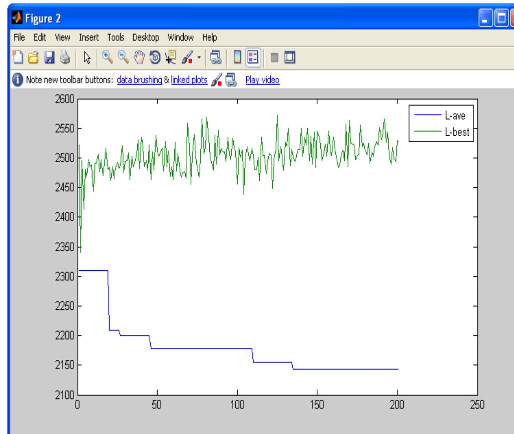












5.CONCLUSION

In this paper, a new approach for circuit diagnostic test Generation TPG structure optimization is proposed that combines PSO algorithm and GA. The initial population of PSO is generated randomly, and the population evolves towards better test patterns based on fitness function of every stage. Experimental results as demonstrated.

Also a new type of deterministic TPG is presented in the paper. It is based on a feedback shift register composed of D- and T-type flip-flops and inverters. It is also equipped with a modification logic that can invert any bit in any pattern generated by the register. A genetic algorithm which minimizes the area overhead of the TPG for the given deterministic test set is also described. The initial structure of the TPG is encoded and multiplied with some variations to form the initial population. The search for the optimal structure of the TPG is performed by selection, crossover, and mutation operators, while each solution is evaluated by the evaluation tool.

6. REFERENCES

[1] ***E. M. Rudnick, J. H. Patel, G.S. Greenstein, and T. M. Niermann, "Sequential circuit test generation in a genetic algorithm framework," Proc. Design Automation Conf, pp. 698-704, 1994.***

[2] ***Kennedy J,Ebrhart R C, "A Discrete Binary Version of the Particles Swarm Algorithm," In: Proc. 1997 Conf. on Systems, Man, and Cybernetics. Piscataway, NJ: IEEE Press, pp. 4104-4109, 1998.***

[3] ***Fulvio Corno, Paolo Prinetto, Maurizio Rebaudengo et al. "GATTO:A Genetic Algorithm for Automatic Test Pattern Generation for Large Synchronous Sequential Circuits." IEEE Transactions on Computer-aided Design of Interated Circuits and Systems, pp.991-1000, 1996.***

[4] ***F. Corno, P. Prientto, M. S. Reorda, G.. Squilero. "A new approach for initialization sequences computation for synchronous sequential circuits." Proceedings of IEEE International Conference on Computer Design, pp.381-386, 1997.***

[5] ***Ilker Hamzaoglu, Janak H. Patel, "Test Set Compaction Algorithms for combinational Circuits," IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, pp. 957-963, Aug, 2000.***

[6] ***Lu Changhua, Zhang Qibo, Jiang Weiwei. "A Global Optimization Algorithm for the Minimum Test Set of Circuits." IEEE International Conference on Robotics, Intelligent Systems and Signal Proceeding, :pp.1203-1207, Oct, 2003.***

[7] ***CORNO, F.—PRINETTO, P.—REBAUDENGO, M.—SONZA REORDA, M. : GATTO: a Genetic Algorithm for Automatic Test Pattern Generation for Large Synchronous Sequential Circuits, IEEE Transactions on Computer-Aided Design 15 No. 8 (1996), 943–951.***

[8] ***GARBOLINO, T.—HLAWICZKA, A. : A New LFSR with D and T Flip Flops as an Effective Test Pattern Generator for VLSI Circuits, Lecture Notes in Computer Science, vol. 1667, 1999, pp. 321–338.***

[9] ***BUSHNELL, M. L.—AGRAWAL, V. D. : Essentials of Electronic Testing for Digital, Memory and Mixed-Signal Circuits, Kluwer Academic Publishers, 2000.***

Monalisa Mohanty is currently working as a Lecturer, in Electronics and Telecommunication department at DRIEMS,Cuttack . She has completed her B.TECH from BPUT , Bhubaneswar and continuing her M.TECH from BPUT, Bhubaneswar. Her area of interest for Research are- Particle Swarm optimization(PSO), ATPG generation for sequential VLSI circuits and Ant Colony optimization(ACO).