Journal of Theoretical and Applied Information Technology
28th February 2013. Vol. 48 No.3
© 2005 - 2013 JATIT & LLS. All rights reserved·

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TEST BUS ASSIGNMENT OF SYSTEM ON CHIP BY USING CULTURAL PARTICLE SWARM OPTIMIZATION

[1]ZHONGLIANG PAN, [2]LING CHEN
[1]Department of Electronics, School of Physics and Telecommunications Engineering,
South China Normal University, Guangzhou 510006, China.
Email: panz@scnu.edu.cn

ABSTRACT

The system on chip (SoC) must be tested in order to insure the correctness of functions. The test of SoC involves the test of cores and the test of interconnects among cores. The test set of cores must be transported to the inputs of cores through the test access architecture in SoC. Therefore, in order to make the total test time to be minimal, it is necessary to determine an assignment of cores to test access architecture. In this paper, a new method is presented for test bus assignment of system on chip, which is based on cultural particle swarm optimization. In this method, the feasible solutions of test bus assignments are represented by the individuals. The better schemes of test bus assignments are obtained by the evolution of the individuals. The implementation of the method consists of a population space, a belief space, and a communication protocol that describes the exchange mode of knowledge between the population space and belief space. The particle swarm optimization with selection strategy is used for carrying out the evolution of individuals in population space. A lot of experimental results for SoC benchmark circuits demonstrate that the proposed method in this paper can effectively obtain the test bus assignments in shorter time than conventional genetic algorithms.

Keywords: *System On Chip*, *Test Bus Assignment*, *Cultural Algorithm*, *Particle Swarm Optimization*.

1. INTRODUCTION

The recent significant advances in circuit design technology allow designer integrates a complete system into a single chip, the chip is called system on chip (SoC). In the design procedure of SoC, a large number of reusable building blocks and intellectual property blocks (called as cores) are used, typical cores include DSP cores, embedded memory, CPUs, etc [1]. In order to insure the correctness of core functions, each core in SoC must be tested sufficiently. The test of SoC is a complex and difficult task, it involves the test of cores and the test of interconnects among cores [2,3].

Using minimal time to test all cores in a SoC is a goal of SoC test. In this aspect, the test bus assignment of system on chip is a main problem, which determines both partition and assignment of total test bus width. Sehgal [4] et al investigated the dual-speed architectures of test access mechanism (TAM), and presented a heuristic algorithm for TAM optimization and test bus assignment. Iyengar et al [5,6] discussed the design and co-optimization of test wrapper and test access mechanism, and presented an approach to get the test bus sizing and the assignment cores to test buses. Rosen et al [7] investigated the problem of worst-case execution time, test bus assignment and system scheduling for the multiprocessor SoC architectures of real-time applications. Larsson et al [8] discussed the optimization of test data transportation mechanism for bus architecture, and presented an approach that uses additional buffers at each core and allows the test of cores asynchronously, therefore can reduce the test time.

For the test bus assignment with thermal constraints, Chunhua et al [9] presented a test bus assignment approach that satisfies the resource, power, and thermal constraints. Aghaee et al [10] used a temperature sensor to implement the temperature-aware test bus assignment. Jiang et al [11] proposed a thermal-aware test bus assignment method to eliminate the hot spots during the manufacturing tests of SoC.

For the hierarchical SoC, Harmanani et al [12] used simulated annealing approach to carry out the wrapper design, TAM assignment, and the minimization of overall test time. Sehgal [13] et al investigated the design method of hierarchy-aware test infrastructure, the wrapper, and TAM optimization.

In this paper, a new method is presented for the test bus assignment of system on chip, which is based on cultural particle swarm optimization. In this method, the feasible solutions of test bus assignments are represented by individuals, the better schemes of test bus assignment are obtained by performing the evolutionary operations of individuals.

2. FORMULATION OF TEST BUS ASSIGNMENT PROBLEM

2.1 Test Access Architecture

There are many types of cores in a SoC. In order to apply the test set of a given core, access to its peripheries is necessary, therefore this needs a peripheral access architecture around a core to provide both the observability to the core inputs and the controllability to the core outputs. This access architecture is called test access mechanism (TAM). The test access mechanism links test source (e.g. ATE) to core inputs, and links the core outputs to the test sink (e.g. BIST). Here, the BIST is the built-in self-test, the ATE is the automatic test equipment. The TAM is shown in Fig.1 and Fig.2.

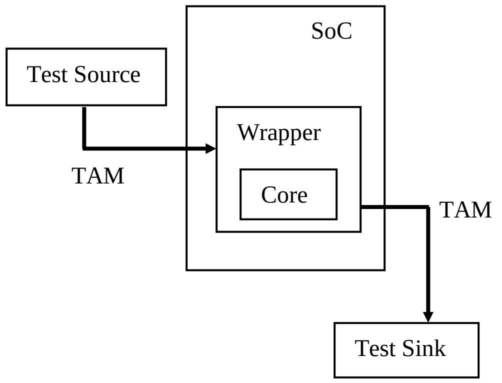

Fig.1 Test Source And Test Sinks.

In Fig.1, the core has a wrapper. The wrapper serves as the interface between the core and the TAM. The wrapper is a thin shell around the core, it supplies the switch between normal functional access and test access via the TAM. A wrapped core is a core with wrapper. A core without a dedicated wrapper is called unwrapped core.

In Fig.2, there are five cores with wrapper. These cores share a TAM. The test vectors are generated in the test source, the test responses are captured in the test sink. The TAM bridges the test source and core, as well as core and test sink. The TAM can transport the test vectors and test responses of cores by the SoC pins.

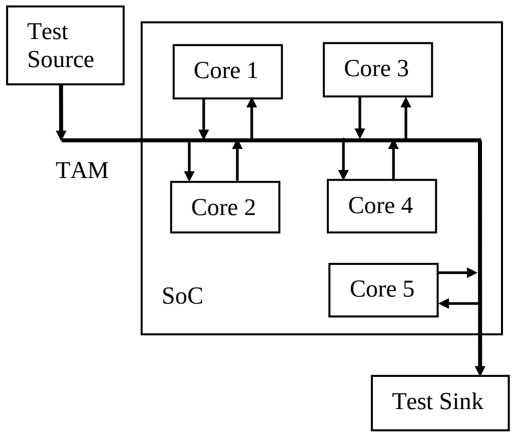

Fig.2 An Example Of Test Access Using

An advantage of wrapped core is that the wrapper can isolate the core during test. In general, a wrapper is in one of three modes: normal mode, internal core test mode, and external test mode. The normal mode is used when the core performs the function of core. The external test mode is used when the interconnection test among several cores are performed. The internal core test mode is used when the test of a core is performed, where the wrapper is put in internal core test mode, and a set of TAM wires is used for the transportation of both test vectors to the core and the test responses from the core.

2.2 Test Bus Assignment

Let K be the number of cores, i.e., there are K cores in a SoC. Let L be the number of TAMs in a SoC, suppose these TAMs are TAM_j (j=1 to L). Let the width of TAM_j be d_j (j=1 to L). The total width D of test bus is the summation of all d_j (j=1 to L), the $D = d_1 + d_2 + \cdots + d_L$.

For the i-th core in a SoC, define following three parameters: P_i , S_0 and S_i. The P_i is the number of test vectors of the i-th core. The S_0 is the length of the longest wrapper scan-out chain, the S_i is the length of the longest wrapper scan-in chain. Suppose the test time for the i-th core assigned to TAM_j is $T_i(d_j)$ cycles. The $T_i(d_j)$ can be expressed by the equation (1).

$$T_i(d_j) = \min(S_i, S_0) + P_i \cdot (\max(S_i, S_0)+1) \quad (1)$$

Suppose the x_{ij} is a variable, which is used for determining the assignment of cores to TAMs according to following mode: The $x_{ij}=1$ if the *i*-th core is assigned to TAM_j, the $x_{ij}=0$ if the *i*-th core do not assign to TAM_j. Thus, the time needed to test all cores on TAM_j is $\sum_{i=1}^{K} T_i(d_j)\cdot x_{ij}$.

The total time to test all cores is expressed by the equation (2).

$$T=\max_{j}\sum_{i=1}^{K} T_i(d_j)\cdot x_{ij}\,,\quad 1\le j\le L \qquad (2)$$

Therefore, the test bus assignment can be formulated as following problem: For a SoC which has *K* cores and *L* test buses with widths $d_1, d_2, \cdots, d_L$, determine an assignment of cores to test buses such that the total test time T is minimal.

In general, the following two constraints are added to the test bus assignment:

$$\sum_{j=1}^{L} x_{ij}=1,\ 1\le i\le K \qquad (3)$$

$$\sum_{j=1}^{L} d_j = D \qquad (4)$$

The equation (3) shows that every core is assigned to exactly one TAM_j. The equation (4) shows that the summation of all TAM_j widths is *D*.

Furthermore, the test bus can be fork out into a set of smaller test buses in order to reduce total test time. These smaller test buses can be used to transport, in parallel, test data to the cores with smaller test data. This test bus subdivision is able to reduce test time, especially when several small cores with small test widths are assigned to a wide test bus. For the cores with larger test data, the undivided parts of test bus are still assigned to them. Therefore, the following constraint is added to the test bus assignment:

$$d_j\le\gamma\,,\quad 1\le j\le L \qquad (5)$$

The γ is a positive constant. The inequality (5) shows that the bits width of each TAM_j is at most γ.

3. TEST BUS ASSIGNMENT BY CULTURAL PARTICLE SWARM OPTIMIZATION

In this Section 3, a new method for the test bus assignment of system on chip is presented, the method is based on the principle of cultural algorithms and particle swarm optimization.

3.1 Structure Of Cultural Algorithms

The cultural algorithms use a basic set of knowledge sources. Each knowledge source is related to the knowledge observed in various social species. These knowledge sources are combined in order to guide the behavior of the individuals [14,15]. The cultural algorithms are based on some theories which try to model cultural as an inheritance process. The cultural algorithms consist of a population space, a belief space, and a communication protocol.

The population space consists of a set of possible solutions (i.e., individuals) for a given problem. Through time, such individuals can be replaced by some of their offspring. The belief space is used to store of the knowledge that can be acquired by the individuals, the information in belief space must be available for the individuals, so that the individuals can use it to modify their behavior.

There is a communication protocol (i.e., link) between the population space and belief space. The protocol gives the type of information to be exchanged between the two spaces, the rules about the individuals that can contribute to belief space, and the way that belief space can influence to the new individuals in population space.

In order to update the belief space, the experiences of the individuals in population space are added to belief space by using the *Acceptance* function. The knowledge in the belief space is used to guide the evolution of the individuals in population space by the *Influence* function.

There are two main aspects in the implementations for the test bus assignment of system on chip by using cultural particle swarm optimization: (1) The representation of individual, and the fitness of an individual; (2) The implementation steps of the algorithm. The two aspects are given in detail in the following two Sections 3.2 and 3.3.

3.2 Representations Of Individuals

For the test bus assignment of system on chip, an individual is a potential solution of the problem, it is an integer array of size *K*+*L*. It consists of following two parts. The first part is the assignment of cores to TAMs, it is an array of *K* integers with the *i*-th element representing the bus number for which the *i*-th core is assigned. The second part of an individual is the total width distribution of TAMs, which is an array of L integers with the j-th element representing the width of the TAM_j, where the sum of these widths is the total width of all TAMs.

The fitness of an individual is defined by 100/(1+T), the T is the test time given in equation (2). When one of constraint equations (3), (4), and (5) is used, the fitness of an individual is 0 if one of constraints has been violated, else the fitness is 100/(1+T).

3.3 Implementation Steps Of Algorithm

In the following, the particle swarm optimization (PSO) is used for the evolution of the individuals of population space in cultural algorithms. The particle swarm optimization is an evolutionary computation technique that is based on observations of the social behavior of animals such as bird flocking, fish schooling, etc [16,17]. The PSO is initialized with a population of random solutions (i.e., individuals). The individuals in the population are called as particles. The trajectory of each particle in search space is adjusted by dynamically altering the velocity of each particle, according to its own flying experiences and the flying experiences of other particles in search space. Here, for the particle swarm optimization being used in this paper, an improved strategy is used, which is to add selection operation into the PSO in order to accelerate the convergence speed.

The implementation steps of cultural particle swarm optimization for the test bus assignment of system on chip is given in Algorithm 1.

Algorithm 1

Step 1. Generate initial individuals in population space;

Step 2. Evaluate initial individuals in population space, i.e., compute the fitness of individuals in the population space;

Step 3. Initialize the belief space, i.e., produce the initial individuals in the belief space.

Step 4. For each individual in the population space, perform the evolutionary operations in the particle swarm optimization to produce new individuals (i.e., particles). Compute the fitness of the new individuals. Replace some old individuals with the new individuals, when the new individuals are better.

Step 5. Update the belief space with a lot of the accepted new individuals in the population space.

Step 6. If the stopping condition is satisfied, then the procedure is terminated, otherwise, go to the Step 4.

The main tasks of Algorithm 1 are the designs of population space, belief space, and the information exchange between the two spaces. The detail implementations are given as follows.

3.3.1 Design of population space

For the individuals in population space, the coding of individuals adopts the same approach given in the Section 3.2. The new individuals in population space are generated by the Algorithm 2.

Algorithm 2

Step 1. Let P(t) be a population consisting of M individuals.

Step 2. Perform the operations of particle swarm optimization to each of individual (i.e. particle) in the P(t), where each parent create an offspring.

Step 3. Make all offspring to form an offspring population S(t). Compute the fitness of individuals in the S(t).

Step 4. Perform selection operation, i.e., pick M individuals to be retained from both parent population P(t) and offspring population S(t). The new population P(t+1) is made up of the M individuals.

The implementation of the Step 2 in Algorithm 2 is as follows.

First of all, for each particle (i.e., individual), calculate its fitness. If the fitness is better than the best fitness (named as pBest) in history, then set current value as the new pBest.

Secondly, choose the particle with the best fitness of all the particles as the gBest.

Thirdly, for each particle, calculate its particle velocity according the following equation (6), and update particle position according equation (7).

Each individual is treated as a particle (a point) in the N-dimensional space. The i-th particle is represented as $X_i = (x_{i1}, x_{i2}, \cdots, x_{iN})$. Let $P_i = (p_{i1}, p_{i2}, \cdots, p_{iN})$ represent the best previous position (i.e., the position giving the best fitness) of the i-th particle.

The index of the best particle among all the particles in the population is represented by the symbol g. Let the V_i represent the rate of position change(velocity) for the particle i, the V_i is expressed by $V_i = (v_{i1}, v_{i2}, \cdots, v_{iN})$.

The velocity vector and position vector are updated according to following equations:

$$v_{ij}(t+1) = w\cdot v_{ij}(t) + c_1\cdot r_1\cdot (p_{ij}(t)-x_{ij}(t))$$

$$+ c_2\cdot r_2\cdot (p_{gj}(t)-x_{ij}(t)) \quad (6)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (7)$$

Where $1 \le i \le M$, $1 \le j \le N$. The i is the order number of particle, the j represents the j-th dimensional of particle, the t represents the evolutionary generation of the algorithm. The c_1 and c_2 are two positive constants, the r_1 and r_2 are two random numbers in the range [0 1]. The w is a weight, it is a positive constant.

The implementation of the Step 4 in Algorithm 2 is as follows. Here, the parent population is P(t) and the offspring population is S(t). Rank the 2M individuals in the P(t) and S(t) in the descending order of their corresponding fitness, the first M individuals with higher fitness are chosen to constitute a new population P(t+1).

3.3.2 Design of belief space

There are three aspects for the design of belief space, i.e, the encoding of belief individual in belief space, the evolution of individuals in belief space, and the use of belief knowledge. The encoding of the individuals in belief space adopts the same encoding used in population space.

All the initial individuals in belief space are generated randomly. The evolutions of individuals in belief space are performed by selection, crossover and mutation operations in genetic algorithms.

For the belief knowledge, the situational knowledge and normative knowledge are used in this paper. The situational knowledge consists of a set of exemplars from the population. An exemplar consists of both a value for each environmental parameter and the fitness of the individual corresponding to the exemplar. The data structure of situational knowledge is represented by a list of exemplars. Let the number of exemplars be m, let $F(E_i)$ be the fitness of exemplar E_i. The situational knowledge is updated by adding the best individual in the population. The situational knowledge is reinitialized if the environmental changes are detected. These can be shown in the following equations:

If the $F(Z^t_{best}) > F(E^t_1)$ then $< E^{t+1}_1, E^{t+1}_2, \cdots, E^{t+1}_m >$ is $< E^t_{best}, E^t_1, \cdots, E^t_{m-1} >$;

Else if the environmental changes are detected

then $< E^{t+1}_1, E^{t+1}_2, \cdots, E^{t+1}_m >$ is $< Z^t_{best} >$;

Else

$< E^{t+1}_1, E^{t+1}_2, \cdots, E^{t+1}_m >$ is $< E^t_1, E^t_2, \cdots, E^t_m >$.

The t = 0,1, ···. The Z^t_{best} is the best individual in the population at time t. The E^{t+1}_1 and E^{t+1}_2 are the best and the second best exemplars. The situational knowledge represents exemplars or examples for other individuals to follow.

The normative knowledge is represented as a set of intervals, and each interval is viewed to be a promising range for good solutions of a parameter. These ranges provide guidelines for which individual adjustments can be made. When the update mechanism of normative knowledge is used, the interval range can move toward to the lower bound or the upper bound, the changes in the environments can be tracked. Therefore, the normative knowledge is able to provide guidelines or standards for individual behavior.

3.3.3 Design of communication protocol

The communication protocol between population space and belief space consists of two functions: *Acceptance* function and *Influence* function. The *Acceptance* function determines which individuals and their behaviors may impact the knowledge in belief space. The implementation of *Acceptance* function is as follows: If the individuals in population space have evolved μ generations, then the individuals with minimal fitness in belief space are replaced by the individuals with maximal fitness in population space, where the μ is a given positive integer.

The implementation of *Influence* function is as follows: If the individuals in population space have evolved λ generations, then a part of individuals (e.g. 16%) with lower fitness in population space are replaced by the same number of individuals being selected randomly in belief space, where the λ is a given positive integer.

Summarize above discussions in this Section 3, we give the detail implementation of test bus assignment of system on chip by using cultural particle swarm optimization. In the following

Section 4, the experimental results for test bus assignment are given.

4. EXPERIMENTAL RESULTS

We have implemented the test bus assignment of system on chip by using cultural particle swarm optimization given in Section 3 in Visual C++, and a lot of experiments for ITC'02 SoC Test Benchmark Circuits [18] are performed on personal computer with 3.0GHz and 512MB memory. Here we give the experimental results for the SoC benchmark circuits d695, p21241 and p93791.

The parameters used in the Algorithm 1 and Algorithm 2 are: the number of individuals in population space and belief space are 60 and 30, respectively; the maximal number of evolution iterations is 2000, the values of parameters w, c_1 and c_2 are 0.95, 0.90, 0.92 respectively for the equation (6). For the evolution of individuals in belief space, the parameters used in genetic algorithm are: the crossover rate is 0.95, one-point crossover is used, and the mutation rate is 0.0001.

For the test bus assignment of system on chip, we also perform another experiments by using conventional genetic algorithms (CGA) in order to compare the method in this paper with CGA. The parameters used in the conventional genetic algorithm are: the maximal number of evolution generations is 2000, the population size is 90, crossover rate is 0.9; mutation rate is 0.001, the roulette wheel selection scheme and two-point crossover are used.

The experimental results are shown in Table 1 to Table 4. The column "*D*" shows the total width of test bus, the column "partition" shows both the number of bus partition and the value of each width. The column "CP" shows the results by using cultural particle swarm optimization in this paper, the column "CGA" shows the results by using conventional genetic algorithms. The column "Gen" shows the evolution generations being needed for obtaining a core assignment, the column "Time" shows the number of cycles being needed, the definition of the cycles is given in equation (1).

The core assignment gives the assignment scheme of core to bus, which is an array of elements with *i*-th element representing the bus number for which the *i*-th core is assigned.

Each method (the CP and CGA) is performed 20 times repeatedly for a SoC benchmark circuit. For the SoC benchmark circuits d695, p21241 and p93791. The results in Table 1 to Table 4 are obtained on personal computer with 3.0GHz and 512MB memory, the CPU time taken is less than one minute.

The SoC circuit d695 contains ten cores, of which two cores are combinational circuits and eight cores are sequential circuits, i.e., two combinational ISCAS'85 circuits: C6288 and C7552; eight sequential ISCAS'89 circuits: s838, s9234, s38584, s13207, s15850, s5378, s35932, s38417. The Table 1 and Table 2 show the results for SoC d695 with the number of partitions two and three respectively.

Table 1 The Results Of D695 For Two Width Partitions.

D	partition d_1 , d_2	CP		CGA	
		Gen	Time	Gen	Time
16	7, 9	71	40182	124	44390
24	6, 18	64	30514	117	35062
32	11, 21	87	25803	131	28704
40	8, 32	58	20461	93	22031
56	19, 37	161	17734	258	20076
64	41, 23	83	18019	137	19812

About the two width partitions of d695, the core assignments obtained for the total width 16, 24, 32, 40, 56 and 64 are (1 2 1 1 2 2 1 1 1 1), (2 1 1 1 2 2 1 1 1 2), (1 2 1 1 2 2 2 1 1 1), (1 2 1 1 2 2 1 1 2 2), (1 2 1 1 2 1 2 2 1 2), and (1 1 2 2 1 2 1 1 1 1), respectively.

Table 2 The Results Of D695 For Three Width Partitions.

D	partition d_1 , d_2 , d_3	CP		CGA	
		Gen	Time	Gen	Time
16	8, 6, 2	672	41056	1041	48131
24	2, 5, 17	386	28011	607	30514
32	9, 18, 5	337	21093	581	22157
40	18, 16, 6	371	18014	614	20143
56	5, 18, 33	429	13012	783	17905
64	17, 6, 41	304	13507	562	16816

About the three width partitions of SoC circuit d695, the core assignments obtained for the total width 16, 24, 32, 40, 56 and 64 are (1 1 3 3 1 2 2 3 3 1), (2 2 2 1 3 3 3 1 3 2), (3 3 3 3 1 2 2 3 3 2), (

2 2 3 3 2 1 2 3 3 1), (3 2 1 1 3 2 3 1 2 3), and (3 3 2 2 3 1 3 2 1 3), respectively.

The SoC circuit p21241 contains 28 cores, of which 6 cores are combinational circuits and 22 cores are sequential circuits. The Table 3 shows the results for SoC circuit p21241 with multiple width partitions, where these partitions contain two, five and six, respectively.

Tab.3 The Results Of P21241 For Multiple Width Partitions

D	partition	CP		CGA	
		Gen	Time	Gen	Time
16	6, 10	117	430281	182	439714
24	8, 16	234	360454	328	368746
32	10, 22	96	319023	151	328709
40	5, 5, 10, 10, 10	492	201846	876	209731
56	12, 17, 16, 3, 1, 7	532	158731	892	160172

About the width partitions of SoC circuit p21241, the core assignments obtained for the total width 16, 24, 32, 40 and 56 are (2 1 2 2 2 2 2 1 1 1 2 2 1 2 2 2 2 1 1 1 1 1 1 1 1 2 1 1), (2 2 2 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 2 2 1), (1 2 2 2 1 2 2 2 2 1 2 2 1 2 1 1 2 1 1 1 1 1 2 2 2 2 2 1), (4 4 3 5 3 1 1 1 3 1 1 5 3 2 1 3 5 3 1 4 3 1 1 4 4 5 3 5), and (1 2 2 1 2 6 6 5 1 1 4 4 6 4 6 5 4 4 4 6 6 1 5 1 2 3 5 6), respectively.

The SoC circuit p93791 is a larger circuit, it contains 32 cores, of which 18 cores are combinational circuits and 14 cores are sequential circuits. The Table 4 shows the results for p93791 with the number of partitions two and three respectively.

Table 4 The Results Of P93791 For Width Partitions.

D	partition	CP		CGA	
		Gen	Time	Gen	Time
32	9, 23	302	896051	547	898706
40	17, 23	154	738604	271	749512
48	23, 9, 16	1305	597813	2494	599807
56	10, 23, 23	2491	516862	3805	520174
64	23, 18, 23	1806	461707	2913	470814

About the width partitions of SoC circuit p93791, the core assignments being obtained for the total width 32, 40, 48, 56 and 64 are (2 1 1 2 1 2 2 1 1 1 1 2 2 2 1 1 2 1 1 1 1 1 2 1 1 2 2 1 1 2 1 1), (1 2 2 1 1 2 1 1 2 1 1 1 2 2 2 2 2 1 1 1 1 2 2 1 1 1 2 2 2 2 1 1), (3 1 2 1 2 1 2 2 1 1 3 3 1 1 2 1 2 2 1 3 1 1 2 2 3 3 1 1 2 1 2 3), (1 3 2 3 2 3 3 3 3 3 1 2 2 2 2 3 2 1 2 3 3 3 2 1 1 3 3 3 2 2 2 2), and (1 1 3 1 2 3 1 1 1 3 2 3 1 1 2 2 2 3 2 1 3 1 2 3 1 2 3 1 2 1 1 2), respectively.

Summarize the above experimental results for ITC'02 SoC Test Benchmark Circuits, it is shown that the method proposed in this paper can effectively solve the test bus assignment of system on chip, and can obtain test bus assignments in shorter time than conventional genetic algorithms.

5. CONCLUSIONS

The test bus assignment has an affect on the total test time of system on chip. In this paper, a new method is presented for test bus assignment of system on chip, which is based on cultural particle swarm optimization. The feasible solutions of test bus assignment is represented by individuals, the better scheme of test bus assignment is obtained by the evolution of the individuals. The experimental results show that the method proposed in this paper can obtain test bus assignment scheme in shorter time than conventional genetic algorithms. In the further, some work needs to be done in building integrated model for the test bus assignment and many types of core tests.

ACKNOWLEDGEMENTS

This work were supported by National Natural Science Foundation of China (No.61072028), the Project of Department of Education of Guangdong Province (No.2012KJCX 0040), Guangdong Province & Chinese Ministry of Education Cooperation Project of Industry, Education and Academy (No.2009B090300339).

REFRENCES:

[1] H.Salamy, J.Ramanujam."An effective solution to task scheduling and memory partitioning for multiprocessor system-on-chip". *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.31, no.5, 2012, pp.717-725.

[2] A.B.Kinsman, N.Nicolici. "Time-multiplexed compressed test of SoC designs". *IEEE Trans. on Very* Large *Scale Integration (VLSI) Systems*, vol.18, no.8, 2010, pp.1159-1172.

[3] Z.Chenchang, K.Hung. "An output-capacitor-free adaptively biased low-dropout regulator with subthreshold undershoot-reduction for SoC". *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol.59, no.5, 2012, pp.1119-1131.

[4] A.Sehgal, K.Chakrabarty. "Optimization of dual-speed TAM architectures for efficient modular testing of SoCs". *IEEE Trans. on Computers*, vol.56, no.1, 2007, pp.120-133.

[5] V.Iyengar, K.Chakrabarty, E.J.Marinissen. "Test wrapper and test access mechanism co-optimization for system-on-chip". *Journal of Electronic Testing: Theory and Applications*, vol.18, no.2, 2002, pp.213-230.

[6] V.Iyengar, K.Chakrabarty. "Test bus sizing for system-on-a-chip". *IEEE Trans. on Computers*, vol.51, no.5, 2002, pp.449-459.

[7] J.Rosen, A.Andrei, P.Eles, P.Zebo. "Bus access optimization for predictable implementation of real-time applications on multiprocessor systems-on-chip". *Proceedings of IEEE International Real-Time Systems Symposium*, Tucson (USA), Dec. 3-6, 2007, pp.49-60.

[8] A.Larsson, E.Larsson, P.Eles. "Optimization of a bus-based test data transportation mechanism in system-on-chip". *Proceedings of Euromicro Conference on Digital System Design*, Porto (Portugal), Aug.30 - Sept. 3, 2005, pp.403-409.

[9] Y.Chunhua, K.K.Saluja, P.Ramanathan. "Power and thermal constrained test scheduling under deep submicron technologies". *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.30, no.2, 2011, pp.317-322.

[10] N.Aghaee, H.Zhiyuan, P.Zebo, P.Eles. "Temperature-aware SoC test scheduling considering inter-chip process variation". *Proceedings of IEEE Asian Test Symposium*, Shanghai(China), Dec.1-4, 2010, pp.395-398.

[11] L.Jiang , X.Qiang, K.Chakrabarty, T.M.Mak. "Integrated test-architecture optimization and thermal-aware test scheduling for 3-D SoCs under pre-bond test-pin-count constraint". *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol.20, no.9, 2012, pp.1621-1633.

[12] H.Harmanani, R.Farah. "Integrated test scheduling, wrapper design, and TAM assignment for hierarchical SoC". *Proceedings of Midwest Symposium on Circuits and Systems*, Montreal (Canada), Aug.5-8, 2007, pp.1388-1391.

[13] A.Sehgal, S.K.Goel, E.Marinissen. "Hierarchy -aware and area-efficient test infrastructure design for core-based system chips". *Proceedings of Design, Automation and Test in Europe*, Munich (Germany), March 6-10, 2006, pp.1-6.

[14] L.Reyes, C.Zezzatti, C.Santillan. "A cultural algorithm for the urban public transportation". *Lecture Notes in Artificial Intelligence*, vol.6077, 2010, pp.135-142.

[15] J.Alami, A.E.Imrani, A.Bouroumi. "A multi-population cultural algorithm using fuzzy clustering". *Applied Soft Computing*, vol.7, no.2, 2007, pp.506-519.

[16] C.Sheng, H.Xia, C.J.Harris. "Particle swarm optimization aided orthogonal forward regression for unified data modeling". *IEEE Trans. on Evolutionary Computation*, vol.14, no.4, 2010, pp.477-499.

[17] P.Wang, L.Shi. "Inter-turn short circuit fault diagnosis of induction motors using the SVM optimized by bare-bones particle swarm optimization". *Journal of Theoretical and Applied Information Technology*, vol.45, no.2, 2012, pp.573-578.

[18] E.J.Marinissen, V.Iyengar, K.Chakrabarty. ITC SoC Benchmark Initiative. International Test Conference, Baltimore(USA),Oct.7-10, 2002, http: //www.extra.research.philips.com/ tc02socbenchm