International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 3 (2015) pp. 7469-7484

http://www.ripublication.com

System-on Chip Test Scheduling using Multi-Objective Particle Swarm Optimisation Algorithm

Angappan Natarajan and M.C.Bhuvaneswari

Assistant Professor (Sr. Gr.),
Department of Electrical and Electronics Engineering,
PSG College of Technology, Peelamedu,
Coimbatore 641004, Tamilnadu, India
+91 9842077436 ann@eee.psgtech.ac.in
Associate Professor
Department of Electrical and Electronics Engineering,
PSG College of Technology, Peelamedu,
Coimbatore 641004, Tamilnadu, India
+91 9442625926 mcb@eee.psgtech.ac.in

Abstract

System-On Chip (SoC) is a methodology in which a complete system can be designed and accommodated in a single chip. SoC has many numbers of deeply embedded cores. The testing of SoC will be a complex and time consuming process so as to test more number of cores. When all the cores are completed testing, testing of SoC completes. The testing process of SoC consists of wrapper optimisation, Test Access Mechanism (TAM) optimisation and test scheduling. Once TAM and test wrapper are determined, the major challenge for system integrator is test scheduling. SoC test time minimization, resource conflicts due to sharing of TAMs, precedence constraints among tests, test power constraints, thermal profile during tests, and scheduling are the different factors that may be considered during the testing of SoC. Test Scheduling refers to the order in which the cores in SoC are tested. Test scheduling problem can be expressed using floor plan. Sequence pair is used for floor plan representation. In this paper, Testing time and idle pin-time product in Floor plan are considered as two objective functions for minimization. The optimal test time and idle pin-time product are obtained using Weighted Sum Multi-Objective Particle Swarm Optimisation (WSMOPSO) algorithm. The algorithm is tested with ITC'02 benchmarks circuits. For the given TAM partition, the test time and the idle pin-time product are calculated and are compared to the test scheduling based on single objective PSO algorithm.

Keywords: System-on Chip, wrapper, scheduling, floor plan, sequence pair, PSO

1. INTRODUCTION

Increasing development in VLSI technology leads to the implementation of a system with billion transistors in a single chip, as a collection of multiple cores, called as System on Chip. These large numbers of components makes the testing process of SoCs a complex process. Testing process should take less time so as to meet the time to market requirements as soon as possible. Testing time for such SoCs are more so that it affects the time-to-market issue. Testers involved in the testing process are costlier. Also, testers have constraint on the power handling capability.

SoC can be tested by serial or parallel method. In serial method, each core of SoC is tested sequentially. This takes more testing time. In parallel testing method, at the same instant of time many cores are tested. This type of testing causes more power dissipation which may lead to damage of SoC. However this method has some disadvantages, it is favoured for testing SoC because this method takes less testing time.

The input circuits considered for testing are ITC'02 Benchmark circuits. ITC'02 SoC benchmark circuits are a set of benchmark circuits officially presented at International Test Conference (ITC'02) in October 2002 in Baltimore, MD, USA. These benchmark circuits are designed with special focus of modular plug-and-play testing of core based SoCs. The set of benchmark currently contains twelve data sets. The Bench mark detail for a circuit contains number of input, number of output, number of scan chains and test patterns. These input details are considered for wrapper optimization, TAM optimization, and SoC test scheduling.

X.Chaun-Pei et al. [1] proposed an algorithm for the test scheduling of SoC with power taken as a constraint using the Particle Swarm Optimization (PSO). Iyengar et al. [2] and Chakrabarty [3] have employed TAM optimization for various numbers of TAMs, based on test bus & test data. The test cores are assigned to TAM width using Integer Linear Program (ILP) algorithm [4], [5]. The concept of effective TAM optimization based on ILP algorithm is explained in these papers.

Erik Larsson and ZeboPeng, [6] have described about the test parallelism which is used as a technique to reduce the test application time for SoC. In this work, Parallel Testing has been conducted under test power constraints. Julien Pougetet.al.[7] have described a test scheduling technique with the objective to minimize the test application time while considering multiple conflicts. The conflicts that have been considered are due to cross core testing, module testing with multiple test sets, hierarchical conflicts in SoCs where cores are embedded, sharing TAM, test power limitations. They have implemented this in ITC'02 Benchmark Circuits.

Vikram Iyengar and Krishnendu Chakrabarty, [8] have described a technique for carrying out the TAM and wrapper optimization in conjunction. They provide an algorithm to construct the wrapper that reduces the testing time of cores. Before this work neither TAM nor wrapper can be optimized at a time. VikramIyengaret.al.[9] represented a heuristic technique for TAM wrapper co-optimization and demonstrated that technique and its stability for several industrial SoCs. Laung-Terng Wang

et.al.,[10] in their book described about various SoC test architectures. Different types of TAM and wrapper and the different modes of operation of wrapper are explained in this book. The concepts of test rail and test bus architecture are described in this book. Laurent Muller et.al.,[11] have published their work about floor planning representations and algorithms. The concept of floor plan and sequence pair is explained and the steps to obtain floor plan from sequence pair are explained in [12], [13].

Based on the literature review, the test scheduling has been implemented for optimal test time with power as a constraint. In this paper, the test scheduling of SoC is done using multi objective evolutionary algorithms for optimal test time and idle pin-time product. This paper is organized as follows. In section 2, A SoC test architecture is described. In Section 3, Justification for SoC Test Scheduling problem that can be represented as a Floor plan and the role of sequence pair is given. The use of MOPSO Algorithm for optimising SoC Test time and idle pin-time product is illustrated in section 4. The result & discussion and conclusion are given in section 5 and 6.

2. SOC TEST ARCHITECTURE

Test mechanism may be of one of the kinds boundary scan testing (IEEE 1149.1, IEEE 1149.6, IEEE P 1687), core-based testing (IEEE 1500). Architecture for testing embedded core based SoC is shown in Fig. 1 [14]. It consists of three parts namely, Test Pattern Source and Sink, Test Access Mechanism (TAM), Core Test Wrapper.

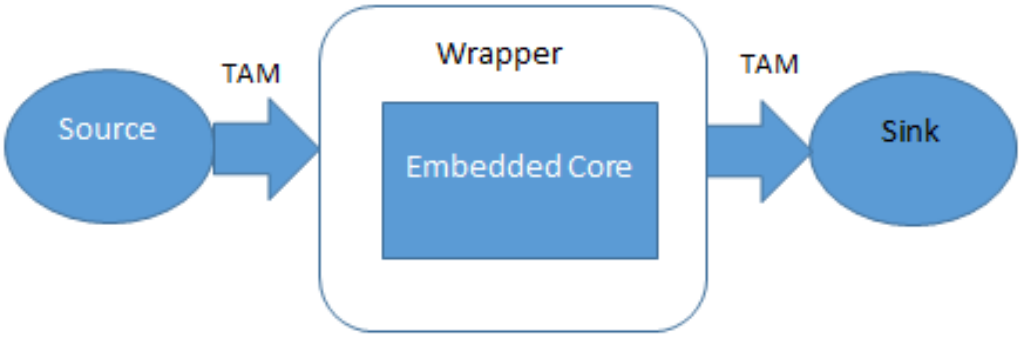

Fig.1 SoC Test Architecture

- **Test Pattern Source and Sink**. The test pattern source generates the test stimuli for the embedded cores and test pattern sink compares the obtained responses to the expected responses.[10]
- **Test Access Mechanism (TAM)**. The TAM transports test patterns. It is used for the on-chip transport of test stimuli from the test pattern source to the core under test and also for the transport of test responses from the core under test to a test pattern sink.

- **Core Test Wrapper**. The core test wrapper forms the interface between the embedded core and its environment. It connects the terminals of the embedded core to the rest of the integrated circuit and to the TAM.

2.1 Testing Process

The SoC testing process includes TAM optimisation, Wrapper optimisation and Test Scheduling shown in Fig.2. Test Scheduling refers to the order in which the cores of SoC are tested.

Fig. 2 Test Process Flow

3. FLOORPLAN REPRESENTATION USING SEQUENCE PAIR

The test scheduling problem can be represented as a floor plan. The Sequence pair is used to represent the floor plan [11]. For example {124536, 326145} is a sequence pair. It has positive and negative loci. The first half is positive loci and the second half is the negative loci. From this sequence pair grid representation is obtained. From the grid graph horizontal Constraint (G_H) graph and vertical Constraint (V_H) graph are obtained. G_H contains a source (s) and a sink (t) placed horizontally and V_H contains source and a sink placed vertically. The width of the module is given as node length for i^{th} module and for nodes‘s’ and‘t’, the width is taken as zero. In this paper, the width is taken as time taken to test the particular core and number of test pins in G_H and G_V respectively. The longest path of the G_H and G_V from source to sink is calculated.

Gridding is the process of encoding a packing into a sequence-pair. The following steps describe a procedure to transform a sequence pair to its floor plan [11]. Consider an n by n grid, where n is the number of modules. Let (G+, G-) be the sequence-pair produced by gridding. Modules x and x’ are related in exactly one of four ways:

$M^{aa}(x)$ = { x’ | x’ is after x in both G+ and G–}
$M^{ab}(x)$ = { x’ | x’ is after x in G+ and before x in G–}
$M^{ba}(x)$ = { x’ | x’ is before x in G+ and after x in Γ G–}
$M^{bb}(x)$ = { x’ | x’ is before x in both G+ and G–}
Let (G+, G–) be the sequence-pair produced by Gridding for a packing
If x’ $\varepsilon M^{aa}(x)$, then x’ is right of x in packing
If x’ $\varepsilon M^{ab}(x)$, then x’ is below x in packing
If x’ $\varepsilon M^{ba}(x)$, then x’ is above x in packing
If x’ $\varepsilon M^{bb}(x)$, then x’ is left of x in packing.

Whereas, the arrangement of the sequence-pair produced by gridding in shown in Fig. 3

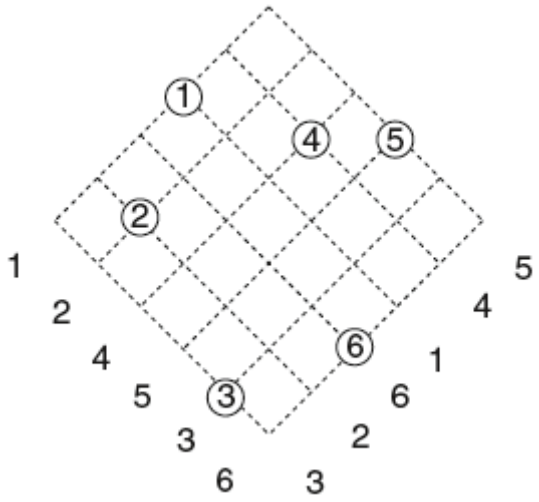

Fig. 3 Sequence-Pair Produced by Gridding

Label the horizontal grid lines and the vertical grid lines with module names along G+ and G– from top and from left, respectively. A cross point of the horizontal grid line of label i and the vertical grid line of label j is referred to by (i, j). Then, rotate the resulting grid counter clockwise by 45 degrees to get an oblique grid as shown in Fig.3. Place each module i with its center being on (i, i).

3.1 Horizontal and Vertical Constraint

On the basis of the preceding constraints, a horizontal constraint graph can be created with a source, a sink, and a node-weighted directed acyclic graph G_H (V, E), where, V is the set of nodes and E is the set of edges as follows:

- V: source s, sink t, and n nodes labelled with module names.
- E: (s, i) and (i, t) for each module i, and (i, j) if and only if module i is on the left of module j (horizontal constraint).
- Nodes weight: zero for s and t, width of module i for node i.

Given a sequence pair (G+, G–), the geometric relation of modules can be derived from the sequence pair as follows:

Rule 1 (horizontal constraint): Module i is left to module j if i appears before j in both G+ and G– (. . . i . . . j . . ., i . . . j . . .).

Rule 2 (vertical constraint): Module i is below module j if i appears after j in G+ and i appears before j in G– (. . . i . . . j . . ., i . . . j . . .).

Similarly, a vertical-constraint graph G_V (V, E) can be constructed on the basis of the vertical constraints and the height of each module. Both the horizontal and the vertical constraint graphs are acyclic in nature. If two modules i and j are in horizontal relation, then there is an edge between nodes i and j in G_H, and thus they do not overlap horizontally in the resulting floor plan. Similarly, if modules i and j are in vertical relation, they do not overlap vertically. Because any pair of modules is either in horizontal or vertical relation, no modules overlap with each other in the resulting floor plan.

The module locations can be obtained from the constraint graphs. The x coordinate of module i is given by the longest path length from the source s to node i in G_H. Similarly, the y-coordinate of module i can be computed on G_V. Consequently, the width and the height of the resulting floor plan can be computed by the longest path length between the source and the sink in G_H and G_V, respectively.

The longest path length computation on each node weighted directed acyclic graph, GH or GV, can be performed in O (n^2) time by applying the well-known longest path algorithm, and where n is the number of modules. In other words, given a sequence pair (G+, G–), the area-optimal packing can be obtained in quadratic time.

For the example sequence pair (G+, G–) (124536, 326145), the corresponding G_H and G_V can be constructed as shown in Fig. 4. The weight and the width (height) of each module are indicated in each node of G_H (G_V).

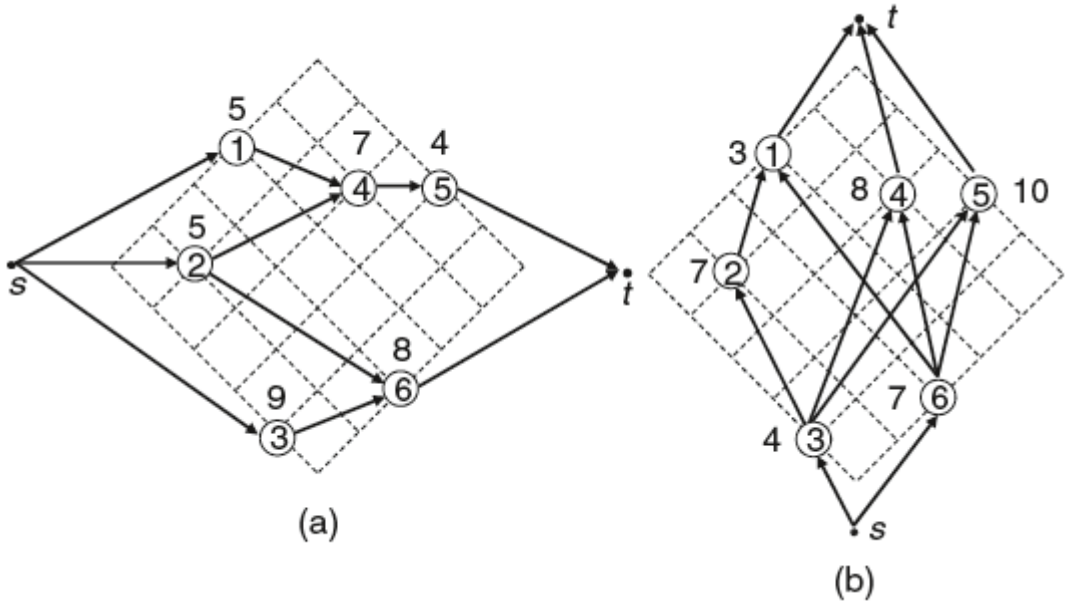

Fig. 4 (a) Horizontal Constraint for Given Module (b) Vertical constraint for Given module

This paper mainly concentrated in obtaining the optimal solution of total testing time and also in finding the minimum idle pin-time product in the floor plan.

Idle pin-time product in the floor plan can be calculated by calculating the total area in the floor plan minus total occupied area in the floor plan. The total area can be calculated by the multiplication of total TAM Width and the longest test time. The Idle pin-time product can be calculated from the difference between the total area and the sum of the individual rectangles as shown in Eq. 1.

$$\text{Idle pin-time product} = (\text{Total TAM width} * \text{Longest Test Time}) - \left(\sum_{i=1}^{n} Core\ TAM\ Width_i * Core\ Test\ Time_i\right) \quad (1)$$

With the test time and idle pin-time product, as the two objective functions, Weighted Sum Particle Swarm Optimisation (WSPSO) algorithm is used to get the optimal solution.

4. PARTICLE SWARM OPTIMISATION (PSO) AND WEIGHTED SUM PARTICLE SWARM OPTIMISATION (WSPSO) ALGORITHMS

PSO algorithm is an evolutionary optimisation technique developed by Eberhart and Kennedy in 1995 [15]. It is easy to implement compared to other evolutionary algorithms and also the parameters to be adjusted are very few. PSO conducts searches using a population of particles.

In this paper, the sequence pair is taken as particles. The particle has its own position and velocity. In order to modify the population and favour the best performing individual, PSO uses adaptable velocity vector for each particle, for which it shifts its position in each iteration. For i^{th} the position and velocity after time t is given as in Eq. 2, 3 [1].

$$v_{id}^{(t+1)} = wv_{id}^{(t)} + c_1 r_1 \left(p_{id}^{(t)} - x_{id}^{(t)}\right) + c_2 r_2 \left(p_{gd}^{(t)} - x_{id}^{(t)}\right) \quad (2)$$

$$x_{id}^{(t+1)} = x_{id}^{(t)} + v_{id}^{(t+1)} \quad (3)$$

Where w is inertia weight, c1 and c2 are two positive constants, and r1 and r2 are two random functions in the range [0, 1]. From the above two formulae the particles position and velocities are updated towards the global best (gbest). Each particle has its own best solution is called as local best (pbest). The best among all the solutions is called as global best (gbest).

4.1 Multi-Objective Particle Swarm Optimisation (MOPSO)

The relative simplicity of PSO and the fact that is a population based technique have made it a natural candidate to be extended for Multi-Objective Optimisation [16]. The existing MOPSO are based on either an aggregating approach or pareto based approach. The aggregating approach combines or aggregates all the objectives of the problem into a single one. The pareto based approach use leader selection techniques based on pareto dominance.

4.2 General Algorithm for Weighted Sum Particle Swarm Optimisation (WSPSO) [16]

STEP 1: initialization

- Generate N particles at random
- Initialize the velocity of N particles to zero.

STEP 2: Repeat until a stopping condition is reached

- Evaluate the fitness of each particle using Eq. 4
- Determine the best vector pbest visited so far by each particle.
- Determine the best vector gbest visited so far by the whole swarm.
- Update the velocity and position of each particle.

The Weighted Sum PSO (WSPSO) uses a weighted aggregate approach to combine the objectives into a single objective. The fitness of the particle is evaluated using the function given by the Eq.

$$F(X) = \sum_{j=1}^{k} w_j \frac{f_j(X) - f_j^{min}}{f_j^{max} - f_j^{min}} \quad (4)$$

Where, $f_j(X)$ is the fitness of X with respect to the j^{th} objective. f_j^{max} and f_j^{min} are the maximal and minimal values of the j^{th} objective respectively. w_j is the weight associated with the j^{th} objective. For this problem a normalised weight vector is randomly generated [17] using Eq. 5 so that the search is widened in all directions instead of fixed directions.

$$w_j = \frac{rand_j}{\sum_{i=1}^{k} rand_i} \quad (5)$$

Optimisation is performed on F(X). The weights are randomly generated to enrich the searching directions and to obtain the solutions with good diversity. The *pbest* and *gbest* are allotted as in case normal PSO. Three parameters contribute to the updating of particle velocity.

- An internal contribution proportional to its previous velocity
- An exploratory contribution proportional to the vicinity of the particle to its personal best position
- An exploratory contribution proportional to the closeness of the particle to the global best position.

  The particle adjusts its velocity V_{ij} through each dimension j by referring to $pbest_j$ and swarm's best experience $gbest_j$ using Eq. 6 [16].

$$v_{ij} = wv_{ij} + C_1 R_1 (pbest_{ij} - p_{ij}) + C_2 R_2 (gbest_j - p_{ij}) \quad (6)$$

C1 is the cognitive constant depicting the relative influence of the particle's personal best position on its velocity, C2 is the social constant depicting the relative

influence of the global best position on its velocity and R1 and R2 are random numbers. The inertia weight w controls the momentum of the particle. A larger w pushes towards the global exploration. The weighting function is given by Eq.7

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{total}} * iter_{total} \quad (7)$$

Where w_{max} represents the initial weight and w_{min} refers to final weight. $Iter_{total}$ - number of iterations and $Iter_{curr}$- current iteration number.

Once the velocities of all particles are determined they are used to obtain the updated position of the particle through each dimension using Eq. 8.

$$P_{ij} = P_{ij} + V_{ij} \quad (8)$$

4.3. WSPSO Algorithm for Test Scheduling

STEP1: Give a set of Sequence Pair which represents the solutions.
STEP2: From the given set of sequence pair grid graph is obtained.
STEP3: From the grid graph find the Horizontal and Vertical Constraint graph.
STEP4: The longest path of the G_H which represents the total testing time needed to test the entire SoC system is calculated. The Idle pin-time product from floor plan can be calculated.
STEP5: Initialize the velocities of all particles as Zero.
STEP6: For all the particles calculate the Velocities using Eq. 5 and based on the new velocities the position of the particles is also updated using Eq. 8.
STEP7: Compare all the Particle best (P*best*) values with the Global best (G*best*). If the P*best* value is better than the G*best* then replace the G*best* with the P*best* else the same will remain.
STEP8: Repeat the steps 2 to 7 until the optimal solutions for the Total Testing Time and also for the Idle pin-time product are obtained. When the optimal solution is obtained the process gets stopped.

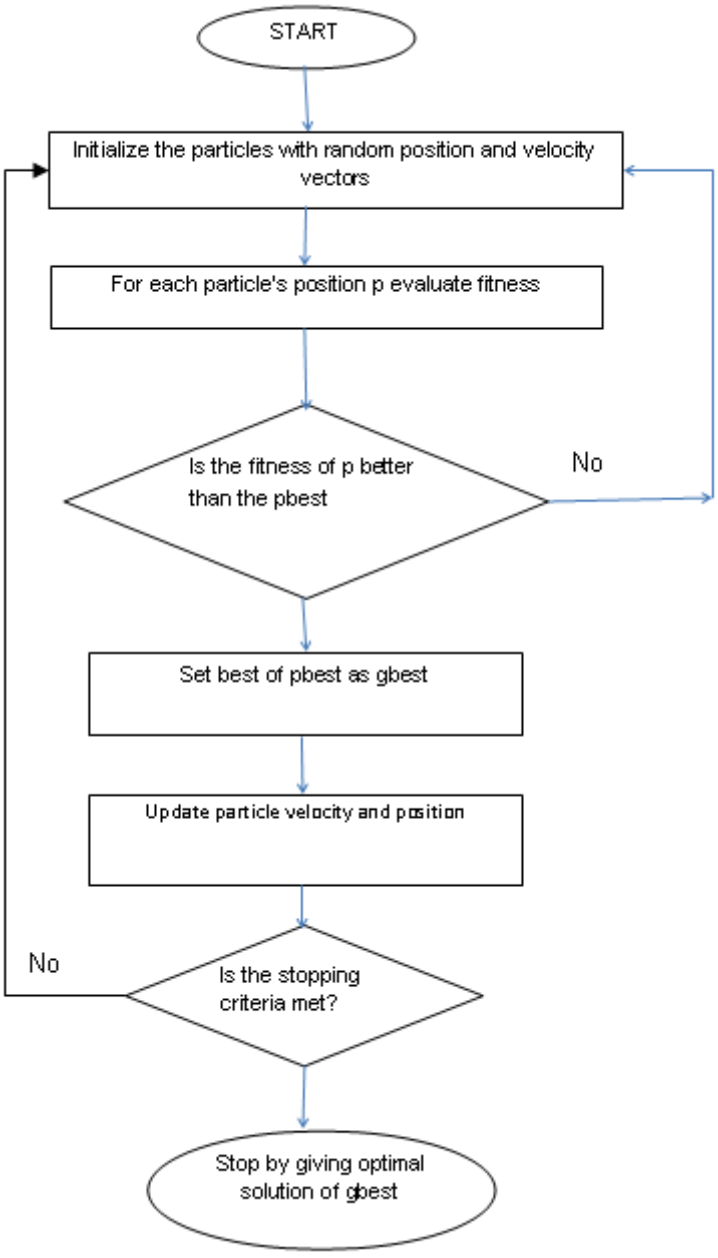

Fig. 5 Flowchart for WSPSO Algorithm

5. RESULTS AND DISCUSSION

The TAM Width of 32, 48, and 64 are considered for analysis. The TAMs are partitioned into either 4 or 5 partitions. This partition is given as input to the system. For the Number of partitions 4, the total TAM width is divided into the partition size of 5, 10, 15, and 2. Based on this, the cores are allocated to TAMs. The optimal testing time with single objective PSO and WSPSO algorithms are obtained. Also the

Idle pin-time product is calculated for individual core allocation. For odd number of partitions, the proposed WSMOPSO based scheme provides better results in terms of test time.

For all the cases the idle pin-time product found to be better when WSPSO algorithm is used. Tab. 1 to Tab. 5 show the Optimal Testing Time obtained for the Benchmark circuits D695, U226, H953, F2126, G1023 using Single Objective PSO algorithm and WSPSO respectively. Fig.6 shows the global best and particle best positions for one set of sequence pair for D695. Table 6 shows that for TAM width 64, WSPSO algorithm performs better in terms of test time and idle pin-time product.

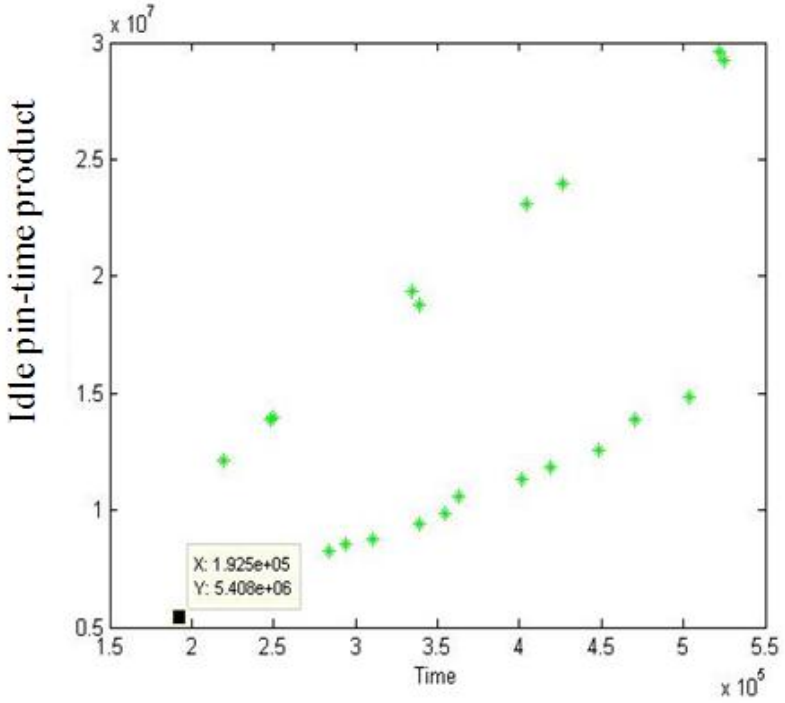

Fig.6. Global Best and Particle Best positions for D695

TABLE 1.Optimal Testing Time of D695 using Single objective PSO and WSPSO Algorithms

Total TAM Width	No. of Partitions	TAM Partitions	Single objective PSO		WSPSO	
			Optimal Testing Time (ms)	Idle pin-time product (ms * Pins)	Optimal Testing Time (ms)	Idle pin-time product (ms * Pins)
32	4	[5 10 15 2]	36024	806105	48974	764159
32	5	[1 1 1 1 28]	208221	5999520	191874	5119076
48	4	[7 16 23 2]	20446	952139	30376	1382288
48	5	[1 1 1 1 44]	212573	9428948	194381	8560273
64	4	[9 21 31 3]	21556	961425	24478	768864
64	5	[1 1 1 1 60]	151644	17016808	206119	11751379

TABLE 2.Optimal Testing Time of U226 using Single objective PSO and WSPSO Algorithms

Total TAM Width	No. of Partitions	TAM Partitions	Single objective PSO		WSPSO	
			Optimal Testing Time (ms)	Idle pin-time product (ms * Pins)	Optimal Testing Time (ms)	Idle pin-time product (ms * Pins)
32	4	[1 1 2 28]	112349	2244648	116468	5823532
32	5	[1 2 10 15 4]	106987	1632291	68197	966997
48	4	[2 16 23 7]	81220	2681322	59870	2202140
48	5	[1 2 15 23 7]	83269	2681322	72375	1976119
64	4	[2 21 31 10]	58737	4131290	67273	1430346
64	5	[1 2 21 31 9]	105753	4204084	75270	3003638

TABLE 3.Optimal Testing Time of H953 using Single objective PSO and WSPSO Algorithms

Total TAM Width	No. of Partitions	TAM Partitions	Single objective PSO		WSPSO	
			Optimal Testing Time (ms)	Idle pin-time product (ms * Pins)	Optimal Testing Time (ms)	Idle pin-time product (ms * Pins)
32	4	[3 10 15 4]	156673	3489523	122692	2046168
32	5	[1 1 1 1 28]	447031	12020503	487824	11079827
48	4	[5 16 23 4]	124062	3091278	122525	2213273
48	5	[1 1 1 1 44]	501689	22398557	447031	17795744
64	4	[5 21 31 7]	120469	1148585	120676	3475860
64	5	[1 1 1 1 60]	482527	1148484	441634	24119639

TABLE 4.Optimal Testing Time of F2126 using Single objective PSO and WSPSO Algorithms

Total TAM Width	No. of Partitions	TAM Partitions	Single objective PSO		WSPSO	
			Optimal Testing Time (ms)	Idle pin-time product (ms * Pins)	Optimal Testing Time (ms)	Idle pin-time product (ms * Pins)
32	4	[6 10 15 1]	335334	2664831	432666	1103247
32	5	[1 1 1 1 28]	2854026	3390299	2854026	536273
48	4	[8 16 23 1]	357088	4295770	385555	530978
48	5	[1 1 1 1 44]	2854026	536273	2904247	636715
64	4	[9 21 31 3]	335334	13116577	335334	4640145
64	5	[1 1 1 1 60]	2854026	6244325	2854026	536273

TABLE 5.Optimal Testing Time of G1023 using Single objective PSO and WSPSO Algorithms

Total TAM Width	No. of Partitions	TAM Partitions	Single objective PSO		WSPSO	
			Optimal Testing Time (ms)	Idle pin-time product (ms * Pins)	Optimal Testing Time (ms)	Idle pin-time product (ms * Pins)
32	4	[4 10 15 3]	26066	1064222	26076	407874
32	5	[1 1 1 2 27]	152812	8456725	147611	3848203
48	4	[6 16 23 3]	23961	1027798	34285	641928
48	5	[1 1 1 2 43]	116775	9711736	150445	6775146
64	4	[8 21 31 4]	22188	1152453	23182	654661
64	5	[1 1 1 2 59]	74449	3891402	115095	6141974

Table 6. Best Results for Different Benchmark Circuits

Benchmark circuit	Using WSPSO				Using Single Objective PSO			
	TAM Width	No. of Partitions	Optimal Testing Time (ms)	Idle pin-time product (ms X pins)	TAM width	TAM partition	Optimal Testing Time (ms)	Idle pin-time product (ms X pins)
D695	64	4	24478	768864	48	4	20446	952139
U226	64	4	67273	1430346	64	4	58737	41312980
H953	64	4	120676	3475860	64	4	120469	1148589
F2126	64	4	335334	4640145	32	4	335334	2664831
G1023	64	4	23182	654661	64	4	22188	1152453

6. CONCLUSION

WSMOPSO algorithm is tested with the benchmark circuits D695 and U226 and the results of optimal values for both the Total Testing Time and the Idle pin-time product is obtained from Table 1 and Table 2. It is concluded that the results obtained for optimal testing time and idle pin-time product by using WSMOPSO is better than the results obtained using single objective PSO. The work can be extended to variants of MOPSO like Non Dominated Sorting MOPSO (NSPSO), Adaptive NSPSO and Hybrid NSPSO and compare the results.

7. REFERENCES

1. Chaun-Pei, X., Hu Hong-Bo, and Niu-Jao, 2009, "Test scheduling of SoC with Power Constrained based on Particle Swarm Optimization Algorithm", Third International Conference on Genetic and Evolutionary Computing, pp. 611-614.
2. Iyengar, V., Chakrabarty, K., and Marinissen, E. J., 2003, "Efficient Test Access Mechanism Optimization for SoC", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.22, pp. 635-643.
3. Chakrabarty, K.,2001, "Optimal Test Access Architectures for SOC", ACM Trans. on Design Automation of Electronics Systems, Vol. 6(1), pp. 26-49.
4. Chakrabarty, K., 2000, "Test Scheduling for core-based systems used mixed-integer linear programming", IEEE Trans. on Computer-Aided Design of Integrated circuits and systems, pp. 1163-1174.
5. Chakrabarty, K., 2000, "Design of System on Chip test architectures using Integer Linear Programming", proc. VLSI Symposium, pp.127-134.
6. Erik Larsson and ZeboPeng, 2005, "System-on-Chip Test Parallelisation under Power Constraints", Proc. Design Automation Test Eur., pp.138-144.
7. Julien Pouget, Erik Larsson and zebopeng, 2005, "Multiple Constraints Driven SOC Test Time Optimization", Journal Of Electronic Testing: Theory and Applications Vol 21(6), pp. 599–611.
8. Iyengar, V., Chakrabarty, K., and Marinissen, E. J., 2002, "Test wrapper and test access mechanism co-optimization for system-on-a-chip", Journal of Electronic Testing: Theory and Applications (JETTA), vol. 18, pp. 213-230.
9. Iyengar, V., Chakrabarty, K., and Marinissen, E. J., 2002, "Efficient Wrapper/TAM Co-optimisation for large SoCs", Proc. Europe Conference and Exhibition on Design, Automation and Test,
10. Laung-Terng Wang, Charles Stroud, Nur Touba, "System-On Chip Test Architectures: nanometre Design for Testability" Morgan Kaufmann, 1st edition, 2008
11. Laurent Muller, Sathiamoorthy Subbarayan, Thomas Antonson, 2004, "Floor-planning representations and Algorithms", Book, Chap.2.
12. Hiroshi Murata, Kunihiro Fujiyoshi, Shigetoshi Nakatake, and Yoji Kajitani, 1996 "VLSI Module Placement based on Rectangle-Packing by the Sequence-Pair", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.15.
13. Masaya Yoshikawa, HidekazuTerai, 2007, "Constraint-Driven Floor-planning based on Genetic Algorithm" Proc. of International Conference on Computer Engineering and Applications, pp. 147-151.
14. Zorian, Y, Marinissen,E. J., and Dey, S., 1999, "Testing embedded-core based system chips", IEEE Computer, Vol. 32(6), pp.52-60.
15. Kennedy, J., Eberhert, R.,1995, "Particle Swarm Optimisation", International Conference on Neural Networks, pp. 1942-1948.

16. Subhashini G.,2012, "Application of multi-objective evolutionary algorithms for task scheduling in heterogeneous distributed systems", Ph.D. thesis, PSG College of Technology, Coimbatore, India.
17. Parasopoulus, K. E., Vrahatis, M. N., 2002, "Particle Swarm Optimisation Method in Solving Multi-Objective Problems", SAC, pp. 603-607.