Particle Swarm Optimization Guided Multi-frequency Power-Aware System-on-Chip Test Scheduling Using Window-Based Peak Power Model

Rajit Karmakar*, Aditya Agarwal† and Santanu Chattopadhyay‡
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology Kharagpur, Kharagpur, 721302, India
*rajit@ece.iitkgp.ernet.in, †adityaagarwal.iitkgp@gmail.com, ‡santanu@ece.iitkgp.ernet.in

***Abstract*—This paper presents a multi-frequency test scheduling strategy for System-on-chip (SoC) under power constraint. While existing approaches consider either global peak or cycle-accurate power model, the proposed work considers an intermediate approach to reduce the power overestimation of global peak power model as well as the computational complexity of cycle-accurate power model. A Particle Swarm Optimization (PSO) guided test scheduling strategy has been integrated with our new window-based peak power model to reduce Test Application Time (TAT) over global peak power model. Experimental results show that further improvement in TAT can be achieved using multi-frequency test environment over single-frequency test approach.**

Keywords*—*System-on-Chip, Power-aware testing, Multi-frequency testing, Window-based peak power model

I. Introduction

Testing of different embedded cores with a common tester (ATE) has become a major challenge for System on Chip (SoC) test designers. Most of the System-on-Chip test scheduling problems [1-6] assume that the cores and the tester operate at a single frequency. However, in recent times multi-frequency operation of embedded cores has drawn the interest of many researchers. Test scheduling under multi-frequency environment has emerged to be a potential research problem.

Test power is another major concern during testing. Due to large number of switching activities in the scan cells, test mode power consumption is much higher than functional mode operation. Total power consumption at any instant by all the cores tested in parallel must not exceed a certain limit to ensure power safe testing. Power-aware test scheduling problem considers a power model of the cores and tries to schedule the cores respecting a certain power budget with the objective to minimize Test Application Time (TAT). Most of the works consider traditional global peak power model of the cores [2-5], where the maximum power consumption of a core during testing is considered as the power consumption throughout the testing of the core. It may generate a too restrictive schedule inhibiting potential parallel testing of some cores. On the other hand cycle-accurate power model [6] is the other power model which does the exact power sum at every instant of the schedule. However, the scheduling algorithm needs to handle large number of power values, hence, takes long time to generate schedule with reasonably good TAT. However, all these approaches are only capable of testing cores operating at single frequency.

This work is partially supported by the research project No. 9(5)/ 2010-MDD dt. 23/1/2011, sponsored by the DeitY, Govt. of India

It may be noted that power consumption is directly proportional to the frequency. With the facility of having the capability of multi-frequency testing, scan-shifting for a power hungry core can be done at lower frequency to minimize its power consumption. Although, it may increase the test time of the core, at the same time it encourages parallel scheduling of multiple cores due to its reduced power consumption. On the other hand, for less power hungry cores, shifting can be done at faster frequency without violating the power budget, thus minimizing the test time. However, as ATE is capable of sending data at a particular frequency, some architectural modification is needed to synchronize the data rate between the cores and the ATE. Wrapper-design for cores with multiple clock domains has been proposed in [9]. It can handle cores with different operating frequencies for different scan chains embedded within it. Several other approaches, using virtual TAM for bandwidth matching and test data rate synchronization between ATE and the cores operating at different frequencies have been presented in the literature [10-12]. A test data multiplexer (TDM) / test data demultiplexer (TDdeM) based approach has been used in [10] to fulfill the gap between the frequency of ATE and the cores. A dynamic reconfigurable multi-port ATE based multi-frequency test scheduling strategy has been proposed in [13]. However, all these works consider fixed average or peak power consumption of the core during scheduling.

In this paper we have proposed a new simple multiplexer based architecture capable of sending test data at different frequencies from ATE to the cores. A new window-based peak power model has also been introduced. Instead of taking cycle-accurate power, the peak power value over a time-window has been taken to approximate the core power over that interval of test time. The process, though introduces some inaccuracy in the power model, works well for most of the test cases. Moreover, the reduction in the number of power values also aid us in designing a Particle Swarm Optimization (PSO) [14] based search procedure to evolve better test schedules than many of the contemporary SoC testing approaches. Experimental results show that, with a small architectural modification for multi-frequency test environment, we can obtain better TAT than single frequency testing. It can also be noted that our window-based peak power model is able to improve the TAT over global peak power model without any significant compromise in the CPU time. Rest of the paper is organized as follows. Section II presents window-based peak power model. Multi-frequency test infrastructure is presented in Section III. . Section IV illustrates the test scheduling procedure. Experimental results and discussions are noted in Section V. Section VI draws the conclusion of the paper.

978-1-4799-4006-6/14/$31.00 ©2014 IEEE

II. Window-Based Peak Power Model

Suppose a certain wrapper configuration of a core C_i has l wrapper chains with wrapper scan-in lengths $SI_1, SI_2, SI_3....SI_l$ and wrapper scan-out lengths $SO_1, SO_2, SO_3....SO_l$. If the core is tested with p test patterns, total test time (T) can be calculated as [7],

$$T = (1+max(SI_i, SO_i))*p+min(SI_i, SO_i), 1 \leq i \leq l \quad (1)$$

The length of test stimuli is equal to the sum of primary inputs (PI), Bidirectional lines $(Bidir)$ and number of scan cells (SC) while the length of test responses is given by the sum of primary outputs $(PO), Bidir$ and SCs.

Cycle-accurate power values can be computed for all sets of wrapper configurations of each core. However, from the cycle accurate power profiles it can be observed that for most of the time instants, there is not significant variation in the power profile in the neighborhood of it. This has motivated us to make the power model coarse-grained via windowing.

In window-based peak power model, we partition the total test time of a core into some smaller sized time intervals (windows) and consider a single peak power in that interval to represent the power value for that interval. A single global peak-power model introduces high amount of overestimation compared to the actual instantaneous power values. The window-based power model has the capability to reduce the amount of overestimation. The accuracy depends on the window-size to a large extent. In our formulation the length of the window interval has been kept flexible and can be tuned according to the requirements and computational resources available. Obviously schedule generation time depends on the length of the window interval. Very small window interval leads to the cycle accurate power model, while large window-size incorporates more false power values in the schedule. Figure 1 shows the concept of window-based peak power model. It shows the cycle-accurate power values, global peak power of 15 units and the window-based model with window-size 10. For example, for the time interval 0 to 10, the peak value is 8 and thus, it has been used as the power value for the entire window. From the figure it can be observed that the amount of overestimation of power is much less in the window-based model compared to the global peak power model.

For the example in Figure 1, the number of power values to be remembered is only 5 in the window-based model, compared to 50 for the cycle-accurate case. For the full 50 cycle operation, overestimation in global peak power model is 353 (computed by summing up the differences between the global peak of 15 and the instantaneous power values), while in window-based model, it reduces to 143.

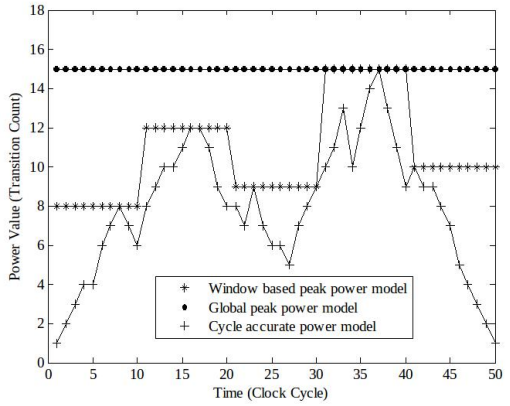

Fig. 1. Comparison of window-based peak power model with other power models

III. Multi-frequency Test Infrastructure

In modern days SoCs, most of the cores have the facilities of being tested at multiple frequency levels. This allows us to shift the test data from ATE to the scan chains at different possible frequencies. Different shift frequencies of a core produce different power profile of the core. Power consumption of a core can be described by the following equation (2).

$$p = \frac{1}{2} C V_{DD}^2 \alpha f \quad (2)$$

Where, C is the output capacitance, V_{DD} is the supply voltage, α is the number of switching activities in the scan chains and f is the shifting frequency of the scan chains. As power is directly proportional to the shifting frequency, an increase in the shift frequency increases the power by the same proportion. However, the test time of the core reduces by same ratio. Similarly for low frequency operation although the power consumption reduces, test time of the core increases. These variations of the test time and power profile with shift frequency of cores introduce lots of flexibilities in the schedule. A test controller needs to select a suitable operating frequency of a core among the available frequency ranges to get a better schedule.

However, it should be noted that ATE can be operated only at a single frequency, while the cores can be tested at different frequency levels. This frequency mismatch between cores and the ATE can be resolved by bandwidth matching between ATE and cores. The cores, operating at higher frequencies should be provided with more resources than normal frequency operation condition, while the low frequency cores require less resource to fulfil the bandwidth requirement criteria.

A. Multi-frequency architecture

To control the test data rate between ATE and the cores operating at different frequency level, TAM architecture needs to be modified. We propose a multiplexer based architecture that coordinates between ATE and cores, to select proper resources. Figure 2 shows the MUX based multi-frequency architecture. For the sake of simplicity we have assumed that a core can operate at five frequency levels ($f/4$, $f/2$, f, $2f$ and $4f$), while the ATE can send data only at a single frequency f. Let, n be the number of TAM wires required to test the core if both the ATE and the core operate at frequency f. Suppose a core operates at frequency $4f$, total bandwidth (BW) required to test the core is $BW = n \times 4f$. So, the ATE, which sends data at a rate of f, has to allocate $4n$ numbers of TAM lines to test the core to fulfil the bandwidth criteria. A test controller decides the operating frequency of individual cores and generates a three bit signal, which chooses the appropriate number of TAM resources required to test the core. A 4:1 and a 2:1 MUX are used for this purpose. Test controller generated signals and corresponding variation in required TAM, test time and power values have been noted in Table I.

TABLE I. VARIATION OF DIFFERENT TEST PARAMETERS WITH OPERATING FREQUENCY OF CIRCUIT UNDER TEST

Controller Signal			Core Frequency	ATE Frequency	TAM Required	Test Time	Power
0	×	×	f	f	n	t	p
1	0	0	$f/4$	f	$n/4$	$4t$	$p/4$
1	0	1	$f/2$	f	$n/2$	$2t$	$p/2$
1	1	0	$2f$	f	$2n$	$t/2$	$2p$
1	1	1	$4f$	f	$4n$	$t/4$	$4p$

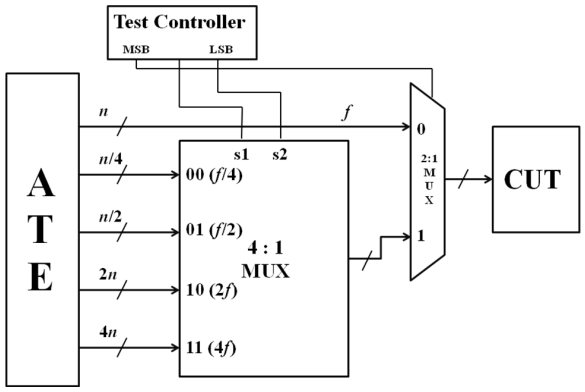

Fig. 2. Multi-frequency architecture

Unlike [10] we do not use test data multiplexer (TDM) or test data demultiplexer (TDdeM) to overcome the frequency gap between ATE and core. Instead our simple MUX based approach can dynamically select the operating frequency of the core during scheduling and allocate the resources. However, it may be noted that the test controller is restricted by the maximum bandwidth of $BW_{max} = W_{max} \times f_{ATE}$ (f_{ATE} is the frequency of ATE, i.e. f). Total allocated resources to all the cores tested parallelly is also limited to W_{max}.

IV. TEST SCHEDULING

A. Problem Formulation

Suppose a SoC with N cores $C_1, C_2, C_3....C_N$ is to be tested with a maximum of W_{max} TAM resources and a maximum power limit P_{max}. Each core can be tested at any of the five frequency levels ($f/4, f/2, f, 2f$ and $4f$). The test scheduling problem is to choose appropriate oprating frequencies of individual cores and allocate TAM resources and test times to them so that, the total test application time (TAT) is minimized. Also the power consumed by the SoC at each instant of time during test has to be less than P_{max}.

Rectangular 2-D bin packing has evolved as a popular method to solve the test scheduling problem for embedded cores [1]. Each core $C_i (1 \leq i \leq N)$ is represented by a set of wrapper configurations R_i. The test resource requirement of core C_i with j^{th} wrapper configuration operating at the same frequency as ATE (i.e. f_{ATE}), can be represented by a rectangle whose height and width represent allocated TAM width (w_{ij}) and the corresponding test time ($T(w_{ij})$) respectively. However, actual test time and allocated TAM width is decided based on the operating frequency of the core selected by a test controller. To get a schedule for the full SoC, the rectangles are to be packed into a bin of fixed height (W_{max}) so that TAT (width of the bin) is minimized. Power constrained scheduling takes this problem to a 3-D bin packing problem, where power represents the third dimension. We have used the window-based peak power profile for each wrapper configuration of each core. Bin packing is NP-Hard [1]. In the following we have presented a PSO based approach to solve the scheduling problem.

B. Test Schedule Generation

The process consists of the following components.

- Generation of test rectangle ($TR_i, 1 \leq i \leq N$) for each core.
- Selection of one test rectangle for each core.
- Selection of operating frequency for each core and calculation of frequency factor $FF_{C_i} = \frac{f_{C_i}}{f_{ATE}} (1 \leq i \leq N)$
- Modification of each test rectangle according to the corresponding frequency factor.
- Scheduling the modified test rectangles ($MTR_i, 1 \leq i \leq N$).

We have used the $Design\_Wrapper$ algorithm [8] to generate different wrapper configurations for each core. For a core, the wrapper configurations corresponding to only the pareto-optimal TAM width [8] are noted. Corresponding window-based power profiles for all possible operating frequencies are determined. Selection of one test rectangle per core has been performed using PSO.

1) Particle Swarm Optimization Formulation: PSO is a population based evolutionary technique designed by Eberhart and Kennedy [14]. Each particle has its fitness value. Particles evolve over generations guided by self and group-intelligence, and also via their inertia.

a) Particle Structure: Let the number of cores in the SoC be N, the maximum number of rectangles for any core be M and the number of frequency values be F. Let $B = \lceil \log_2 M \rceil$ and $K = \lceil \log_2 F \rceil$. A particle consists of $B \times N + K \times N$ number of bits. First $B \times N$ bits select the rectangle indices for the cores, while next $K \times N$ bits select their corresponding frequencies. The decimal equivalent of first B bits identifies the test rectangle selected for the first core, second B bits for the second core, and so on. The decimal equivalent of each K-bit value of remaining $K \times N$ bits selects a frequency. For example, for $K = 3$ and $F = 5$, values are in the range 0 to 4 ($0 \leftarrow f/4, 1 \leftarrow f/2, 2 \leftarrow f, 3 \leftarrow 2f$ and $4 \leftarrow 4f$). Figure 3(a) shows a simple particle with $N = 4$ and $B = 4$, $F = 5$. In this case test rectangles 9, 2, 8 and 13 are selected for cores 1, 2, 3 and 4 and their corresponding frequencies are $f/2, 2f, 4f$ and f respectively. For the initial generation, particles are generated randomly; however care has been taken to ensure that the indices generated for a core are always within the total number of rectangles of it and also the bandwidth ($w_{ij} \times f_{C_i}$) allocated to any core C_i is within the limit of maximum allowable bandwidth value of $W_{max} \times f_{ATE}$. Fitness of a particle is equal to the total test

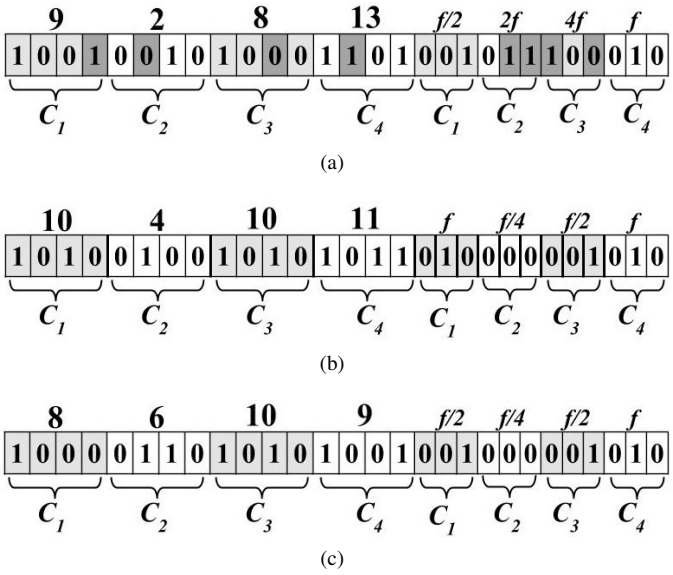

Fig. 3. Particle structure; (a) Initial Particle, (b) Local Best Particle, (c) Evolved Particle

time (TAT) of the SoC after scheduling the test rectangles using the procedure noted in Section IV-B-2.

b) Evolution of a particle: In a PSO formulation, evolution of a particle is guided by three factors − its own intelligence, global (swarm) intelligence, and the inertia factor. A particle always remembers its history about its best structure over generations. This is called the local best $(pbest)$ of the particle. In a particular generation, the particle with the best fitness value is the global best $(gbest)$ of the generation. For the initial generation, $pbest$ of each particle is initialized to itself, while the $gbest$ of the generation is the best one of the first generation. In the successive generations, new particles are created using the $replace$ operator noted next.

The $replace$ operator attempts to align a particle with its $pbest$ and the $gbest$ particles, with some probability. For the sake of this alignment, the $replace$ operator is applied at each individual position of a particle. In the first part of the particle, for bit position i, the bit is replaced by the corresponding bit of $pbest$ particle with probability α. After the operator has been applied for $pbest$, the same is done with respect to $gbest$ with probability of replacement, β. After both the replacement operators have been applied to all bit positions for a core, a consistency check is performed. If the new rectangle number for the core becomes larger than the total number of rectangles available for the core, the rectangle number is reverted back to its value in the original particle. Similar replacement operations and consistency checks are also done on the frequency part of the particle. In our experimentation, we have kept both α and β values at 0.1. Figure 3 shows an example alignment of a particle towards its local best. The dark shaded bits of the initial particle (Fig. 3(a)) are replaced by the bits in the same position as the local best particle (Fig. 3(b)) to get the evolved particle (Fig. 3(c)).

2) Scheduling of test rectangles: The algorithm takes as input the rectangle set TR_i, $(1 \leq i \leq N)$ corresponding to the particle, the maximum TAM width W_{max}, and the maximum power limit P_{max}. However, these rectangles cannot be directly used for scheduling due to bandwidth matching problem under multi-frequency environment. The rectangle set TR_i, $(1 \leq i \leq N)$ is modified to a new set of rectangles MTR_i, $(1 \leq i \leq N)$ according to the frequency f_{C_i} of each core C_i. The height w_{ij} of TR_i is multiplied by FF_{C_i} to get the height of MTR_i, while the width of MTR_i can be calculated by dividing the width of TR_i $(i.e.\ T(w_{ij}))$ by FF_{C_i}. The power profiles of the cores corresponding to the modified rectangle set are modified accordingly, keeping the total energy during testing same for both TR_i and MTR_i.

Scheduling is performed using this new set of rectangles MTR_i, $(1 \leq i \leq N)$, honouring the constraints that at no instant of time, total TAM width requirement exceeds W_{max}, and the instantaneous power value does not exceed P_{max}. The resulting total test time (TAT) is the fitness of the particle. At any point of time, the algorithm maintains the following data structures to arrive at a decision about scheduling the next core.

1) ***Break_Point_List (BP):*** A set of time instants at which the power requirement of the schedule has changed from its value in the previous instant. The next core can be scheduled at any time of the break-points, $bp_k \in BP$.
2) ***Available_TAM_Width_Info (ATW):*** A set with cardinality same as BP. The value atw_k is equal to the total free TAM width available at break point instant bp_k.
3) ***Power_Tracker (PT):*** This is also a set with cardinality same as BP and holds the total power consumed by the already scheduled cores at the corresponding break point instant.

As the till unscheduled cores get scheduled, the list BP, ATW and PT also get updated. The bin packing procedure also needs to prioritize the next unscheduled rectangle to be selected for packing (scheduling). For this purpose the rectangles (MTR) are sorted on their area values (TAM width (w) × test time (T)) in a descending order. The break-point list BP is scanned from the minimum to the maximum value.

To make the schedule compact, we try to utilize any available TAM resource and power budget at every break-point. Hence, for the break-point bp_k, the algorithm scans the unscheduled rectangle list to check for the largest rectangle that can be scheduled at bp_k. If none are feasible, the algorithm advances to the next break-point. Power requirements are also checked to ensure satisfaction of power limit at every break-point till the end of schedule for the current core. When rectangles corresponding to all cores have been scheduled, the maximum end time of testing of all cores gives the total test application time. The algorithm to produce the schedule is presented next.

Algorithm Schedule-Rectangles

Input:
List of modified rectangles (MTR) to be scheduled
W_{max}, the maximum TAM width
P_{max}, the power limit

Var:
BP : A list of break points.
ATW : List of available TAM widths at each break point $bp \in BP$.
PT : List of total power values at each break point $bp \in BP$.
Begin
$BP \leftarrow 0$
$ATW \leftarrow 0$
$PT \leftarrow 0$
Sort list of rectangles on decreasing area
Mark all rectangles as unscheduled
while there exists unscheduled rectangles **do**
while all entries of BP not checked **do**
Let bp_k be the next entry of BP
$atw_k \leftarrow ATW[bp_k]$
$pt_k \leftarrow PT[bp_k]$
Check if any rectangle picked up in sorted order can be scheduled at bp_k with available TAM resource and power budget
if yes **then**
Update BP, ATW, PT and Rectangle List
Mark corresponding rectangle scheduled
else
Continue with next $bp_k \in BP$
end if
end while
end while
Return the maximum test end time among all rectangles.
End

V. Experimental Results

In this section we present results of experimentation with three most popular $ITC'02$ [15] benchmark SoCs. However, none of the previous works consider window-based peak power model. So, direct comparison of our work with other related works is not possible. To compare the quality of our PSO guided test schedule algorithm, we have first compared our work with some other global peak power model based single frequency test scheduling approaches [2-5]. For this purpose, the peak power values for cores are taken as those mentioned in [2]. Table II notes the power-aware test scheduling results for the largest $ITC'02$ benchmark $p93791$, corresponding to power limit 20000. While the works reported in [2-5] consider only this global peak power values, [6] have reported scheduling results for both global and cycle-accurate power models. Our results, noted under the column marked PSO GP are better than other techniques including the cycle-accurate model of [6], for most of the cases. This shows the quality of our test scheduling algorithm.

However, detailed information regarding test patterns is required to create the window-based peak power model. As $ITC'02$ benchmarks do not have any information on test patterns, we have randomly generated the test patterns. Total number of scan transitions in a cycle has been taken as the measure of power consumption by the core at that cycle. It may be noted that these power values become different from those mentioned in [2]. Table III shows comparative study between window-based power model and global peak power model as well as between single frequency testing and multi-frequency testing. To have a better understanding of the effect of different power models on TAT, we have compared the window-based power model results with global peak power model, keeping multi-frequency test environment as constant for both the cases. Similarly the comparison between multi-frequency approach and single frequency approach has been carried out under same window-based peak power profile. In Table III the global peak multi-frequency approach is marked as GPMF, while WPSF and WPMF indicate window-based peak power single frequency approach and window-based peak power multi-frequency approach respectively.

For our experiment we have considered the length of the window interval as 1000 clock cycles for benchmark $d695$ and $p22810$ and 5000 clock cycles for $p93791$ SoCs. In our case studies we have taken the population size of PSO as 2000, 3000, 5000 numbers of particles respectively for $d695$, $p22810$ and $p93791$. Simulation terminates if there is no improvement in solution over 1000, 1500, 2000 generations respectively.

It is clear from Table III that a significant improvement in TAT can be achieved by using multi-frequency test environment. It may be noted from column 3 that our multi-frequency approach is able to generate test schedule under low power constraint, while single frequency approach fails to do that. It is justifiable that under multi-frequency test environment, power hungry cores can be tested at low frequency to minimize their power consumption. A proper selection of operating frequency of each core actually helps to reduce TAT as well as encourage low power test scheduling. It may also be noted that our window-based peak power model performs reasonably better than global peak power model under same multi-frequency environment. CPU times needed for schedule generation of our approach are reasonably small, about 0.02 seconds on a 2.40 GHz Intel Core2 Duo processor having 3 GB main memory.

TABLE II. Comparison Of PSO With Other Scheduling Procedure For $p93791$ For P_{max} = 20000 And Different W_{max}. (As Per Power Values Mentioned In [2])

p93791	P_{max} **= 20000**						
W_{max}	**MC [2]**	**Cycle Accurate [6]**		**GA [3]**	**ACO [5]**	**SFLA [4]**	**PSO GP**
		GP	**CA**				
16	1827819	1835416	1829232	1759656	1787856	**1660342**	1730385
24	1220469	1233680	1233716	1174517	187161	1203156	**1159733**
32	957921	932323	934069	886869	912503	993822	**871780**
40	821575	766353	769378	712053	-	-	**702252**
48	658132	640602	640615	600632	623013	611527	**583762**
56	549669	550636	539815	508947	573720	598228	**507406**
64	493599	485297	492463	450977	-	-	**444511**

TABLE III. COMPARISON OF DIFFERENT TEST SCHEDULING APPROACHES FOR DIFFERENT P_{max} AND W_{max} FOR $ITC'02$ BENCHMARKS (AS PER POWER VALUES GENERATED BY US)

d695	P_{max} = 1000			P_{max} = 1500			P_{max} = 2000			P_{max} = 3000		
W_{max}	**GPMF**	**WPSF**	**WPMF**	**GPMF**	**WPSF**	**WPMF**	**GPMF**	**WPSF**	**WPMF**	**GPMF**	**WPSF**	**WPMF**
16	41200	NS*	**40824**	40824	41981	**40477**	38722	41833	**38587**	38254	41771	**38144**
24	30288	NS	**30195**	27744	28132	**26964**	27979	28132	**26964**	27489	27489	**26486**
32	**21767**	NS	**21767**	21126	21476	**20638**	20640	21318	**20568**	20638	21136	**20472**
40	18695	NS	**18416**	17093	18023	**16962**	16915	17093	**16914**	**16701**	16962	**16701**
48	18549	NS	**17889**	14453	15898	**14348**	14388	14688	**14240**	**14016**	14310	**14016**
56	18509	NS	**17889**	12579	14797	**12365**	12324	12547	**12320**	12247	12155	**12155**
64	18323	NS	**17889**	12157	14680	**11720**	10856	10875	**10817**	10611	10817	**10571**
p22810	P_{max} = 6500			P_{max} = 8000			P_{max} = 10000			P_{max} = 12000		
W_{max}	**GPMF**	**WPSF**	**WPMF**	**GPMF**	**WPSF**	**WPMF**	**GPMF**	**WPSF**	**WPMF**	**GPMF**	**WPSF**	**WPMF**
16	383956	NS	**381990**	**381990**	433430	**381990**	**381990**	431966	**381990**	381990	431966	**381990**
24	**290834**	NS	**290834**	278641	292548	**271142**	274227	289845	**271142**	274227	288633	**271142**
32	**213318**	NS	**213318**	211202	222487	**206365**	206961	218613	**201162**	203404	218322	**201162**
40	174223	NS	**168529**	170372	177763	**167868**	168291	177763	**167868**	168210	177312	**167490**
48	147622	NS	**145962**	**145962**	151001	**145962**	145962	151001	**145102**	145662	150413	**145102**
56	140040	NS	**134266**	129574	129292	**125909**	125356	129292	**125278**	125356	129292	**123938**
64	**128399**	NS	**128399**	**116625**	116625	**116625**	114832	116625	**111401**	110450	116625	**109040**
p93791	P_{max} = 12000			P_{max} = 15000			P_{max} = 20000			P_{max} = 25000		
W_{max}	**GPMF**	**WPSF**	**WPMF**	**GPMF**	**WPSF**	**WPMF**	**GPMF**	**WPSF**	**WPMF**	**GPMF**	**WPSF**	**WPMF**
16	1555974	NS	**1500754**	1541789	1771797	**1500754**	1520095	1769673	**1479144**	1483264	1763891	**1479713**
24	1097268	NS	**1086643**	1086643	1178986	**1078119**	1078119	1178986	**1072684**	1074274	1178741	**1052881**
32	862866	NS	**845676**	862749	888064	**835715**	836302	888064	**835442**	836302	888038	**835442**
40	704919	NS	**704271**	700943	718005	**691604**	674746	710508	**664247**	666076	710508	**664247**
48	590197	NS	**589752**	588713	592200	**580268**	586362	592200	**581577**	582601	592200	**572976**
56	515930	NS	**513953**	513953	513658	**513658**	505525	512188	**498674**	503408	509565	**460304**
64	454672	NS	**448202**	442768	450523	**441430**	440796	449633	**440241**	440796	447244	**440241**

* NS - No Schedule Possible

VI. CONCLUSION

In this paper we have presented a multi-frequency test scheduling approach using a new power model named window-based peak power model. The developed power model has been shown to reduce the power overestimation in the scheduling process. The model has been incorporated into a PSO based formulation to select test rectangles for cores and scheduling them using a 3-D bin-packing approach. Our multi-frequency test approach helps to reduce overall test application time as well as capable of low power test scheduling. However, with a small modification in the MUX based multi-frequency architecture, our scheduling procedure can be used for multi-clock domain SoC testing.

REFERENCES

[1] V. Iyengar, K. Chakrabarty, and E.J.Marinissen, "On using rectangle packing for soc wrapper/tam co-optimization," in *VLSI Test Symposium, 2002. (VTS 2002). Proceedings 20th IEEE*, 2002, pp. 253,258.

[2] J.Pouget, E.Larsson, and Z.Peng, "Soc test time minimization under multiple constraints," in *Test Symposium, 2003. ATS 2003. 12th Asian*, Nov 2003, pp. 312,317.

[3] C.Giri, S.Sarkar, and S. Chattopadhyay, "A genetic algorithm based heuristic technique for power constrained test scheduling in core-based socs," in *Very Large Scale Integration, 2007. VLSI - SoC 2007. IFIP International Conference*, 15-17 Oct 2007, pp. 320,323.

[4] X. L. Cui, X. M. Shi, and C. L. L. H. Li, "A shuffle frog-leaping algorithm for test scheduling of 2d/3d soc," in *Solid-State and Integrated Circuit Technology (ICSICT), 2012 IEEE 11th International Conference*, 29 Oct. - 1 Nov 2012, pp. 1,3.

[5] X. L. Cui, W. Cheng, and C. LI, "Effective soc test scheduling under peak power constraints based on the ant colony algorithm," in *IEEE International Conference on Test and Measurement (ICTM)*, 2010, pp. 189,192.

[6] S. Samii, M.Selkala, E. Larsson, K. Chakrabarty, and Z. Peng, "Cycle-accurate test power modeling and its application to soc test architecture design and scheduling," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions*, vol. 27, no. 5, May 2008, pp. 973,977.

[7] E. Marinissen, S. Goel, and M. Lousberg, "Wrapper design for embedded core test," in *Test Conference, 2000. Proceedings. International*, 2000, pp. 911,920.

[8] V.Iyengar, K.Chakrabarty, and E.J.Marinissen, "Test wrapper and test access mechanism co-optimization for system-on-chip," in *Test Conference, 2001. Proceedings. International*, 2001, pp. 1023,1032.

[9] X. Qiang and N. Nicolici, "Wrapper design for multifrequency ip cores," in *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 13, no. 6, 2005, pp. 678,685.

[10] T. Yoneda, K. Masuda, and H. Fujiwara, "Power-constrained test scheduling for multi-clock domain socs," in *Design, Automation and Test in Europe, 2006. DATE '06.*, vol. 1, March 2006, pp. 1,6,6–10.

[11] Z. Dan, H. Ronghua, T. Yoneda, and H. Fujiwara, "Power-aware multi-frequency heterogeneous soc test framework design with floor-ceiling packing," in *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium*, 27-30 May 2007, pp. 2942,2945.

[12] Z. Dan, U. Chandran, T. Yoneda, and H. Fujiwara, "Shelf packing to the design and optimization of a power-aware multi-frequency wrapper architecture for modular ip cores," in *Design Automation Conference, 2007. ASP-DAC '07. Asia and South Pacific*, 23-26 Jan 2007, pp. 714,719.

[13] Z. Dan, H. Ronghua, and H. Fujiwara, "Multi-frequency modular testing of socs by dynamically reconfiguring multi-port ate," in *Asian Test Symposium, 2007. ATS '07. 16th*, 8-11 Oct 2007, pp. 107,110.

[14] J.Kennedy and R.Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference*, Nov/Dec 1995, pp. 1942,1948.

[15] ITC'02 SOC Test Benchmarks. [Online]. Available: http://itc02socbenchm.pratt.duke.edu/