Power-Conscious Microprocessor-Based Testing of System-on-Chip

Fawnizu Azmadi Hussin[1], Tomokazu Yoneda[1], Alex Orailoglu[2], and Hideo Fujiwara[1]

[1]Grad. School of Information Science, Nara Institute of Science and Technology, Kansai Science City, 630-0192, Japan
fawnizu@ieee.org, {yoneda, fujiwara}@is.naist.jp
[2]Computer Science and Engineering Department, University of California San Diego, La Jolla, CA 92093
alex@cs.ucsd.edu

Abstract In this paper, we are proposing a core-based test methodology that utilizes the functional bus for test stimuli and response transportation. An efficient algorithm for the generation of a complete test schedule that efficiently utilizes the functional bus under a power constraint is described. The test schedule is composed of a set of test vector delivery sequences in small chunks, denoted as packets. The utilization of small packet sizes minimizes required buffer sizes while optimizing the functional bus utilization. The experimental results show that the methodology is highly effective compared to previous approaches that do not use the functional bus. The strong results of the proposed approach are particularly highlighted when small bus widths are considered, an important consideration in current SOC designs where increasingly larger bus widths pose routing and reliability challenges.

Keyword Functional bus, Microprocessor-based, Power-Constrained, Test Packet, Test Scheduling

1. Introduction

The System-on-Chip (SOC) design methodology offers considerable benefits, which can be identified in the two distinct perspectives of shortening the design cycle, and delivering reusability of pre-designed cores and their associated test set. The IEEE 1500 standard supports test reuse methodology by standardizing a test wrapper [1].

A lot of work has been already performed to address relevant issues on SOC testing. Broadly, prior work in the area can be categorized as belonging to either of the two classes of (1) the delivery mechanism for the test data and (2) the scheduling of the core tests with various objectives and constraints. In regards to the test delivery mechanism, several types of test access mechanism (TAM) have been proposed such as TestRail [2], TestBus [3], and Virtual TAM [4]. All these techniques propose the addition of TAM architectures to the SOC in order to support the test application strategies they have developed.

The test scheduling approaches hitherto proposed in [5,6] can be classified as TAM-based. In all of these cases, the test scheduling schemes utilize a TAM similar to [2,3,4] to deliver test data to the cores under test. An extraneous TAM is consistently added to the SOC for the sole purpose of delivering the test vectors from external automatic test equipment (ATE) to the module under test.

In all SOCs, a functional communication network is readily available, and presents an alternative to the extraneous TAM for testing purposes. In addition, in most SOCs, embedded processors exist which could potentially be used to replace or complement the ATE. For simplicity, we will refer to the communication network as a *functional bus*; the communication network may not necessarily be in the form of a straightforward bus topology.

While there exists plentiful work in the literature on TAM-based testing, only two research groups [7,8] have previously addressed functional bus-based test scheme. In [7], the authors propose an interface between the PCI bus and the modules under test. In [8], a buffer interface between the bus and the modules under test is proposed, while the control of test application is performed by a Finite State Machine based controller.

In this paper, we illustrate our approach which utilizes the functional bus to test all modules in the SOC. In the process, we show how our approach greatly simplifies the test program, one of the primary strengths and differentiators of our proposed methodology. Such simplification is attained through the support of an efficient test architecture, which includes appropriate timing control circuitry.

We begin with some motivation in section 2. In section 3, the support architecture design for the efficient utilization of the functional bus during testing is explained. Section 4 elaborates the methodology to develop an efficient test schedule using the functional bus. In section 5, we thoroughly evaluate our methodology experimentally. Finally, a brief set of conclusions is offered in section 6.

2. Motivation

Let us look at some of the possible scenarios regarding packet based test delivery utilizing the functional bus. To ease description, let us denote each of the small test data units as a test packet.

Round-robin ordering would be a reasonable first attempt at scheduling the test delivery because of its fair allocation of the bus as shown in Figure 1a, where the x-axis represents time. Each packet will go through two separate stages of transfer (illustrated by Figure 2). First, it is delivered from processor to the buffer through the functional bus. On the second stage, it is transferred from the buffer into the scan chains.

Figure 1 represents the time a packet takes in stage 1 (labeled *bus*) and stage 2 (labeled m_i) by rectangles with the same shading. Stage 1 of the subsequent test packet for a module can only begin, to avoid buffer overflow, after stage 2 of the previous test packet for that module has been completed. Furthermore, since stage 1 uses a common bus, only one test packet can be in stage 1 at any given time.

Stage 1 and stage 2 are also referred to as test delivery and test application, respectively, for the test packet.

Figure 1a shows m_3 idle, waiting for test data because the test packet for m_3 cannot be delivered until the test packet for m_2 has been delivered. However, the test packet for m_2 cannot be delivered until test application of the previous packet of m_2 has been completed. Consequently, m_3 is starved for test data and at the same time the bus remains idle while waiting for m_2 to complete test application even though m_3 needs test data. An analogous situation holds for m_1. m_2, on the other hand, always receives its test data in a timely manner at the expense of starving m_1 and m_3.

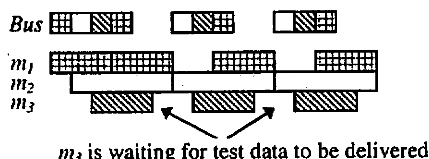

Figure 1a: Round-robin scheduling

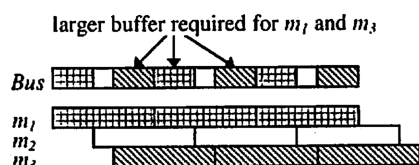

Figure 1b: Packet size and buffer size variations

The problem can be remedied by increasing the packet size for m_1 and m_3, as in Figure 1b. However, this quick fix implies that larger buffer spaces are required for m_1 and m_3 to store the larger packet sizes. We can reduce packet size for all modules, but the minimum buffer size is constrained by the smallest packet size of m_2. Further reduction in packet sizes for m_1 and m_3 would reintroduce the problem illustrated in Figure 1a.

An additional challenge stems from the fact that packet sizes cannot be arbitrary, because data delivery is conducted through a discrete number of bus wires. Figure 2 shows one possible buffer interface between the bus and the module. There are several different variables that can affect the test scheduling: bus frequency, f_b, scan frequencies, f_{sc1}, and f_{sc2}, bus width, number of scan chains, and volume of test data for each module.

Even though a functional bus and a TAM may be similar in many ways, the underlying issues that need to be considered are completely different. We can broadly categorize them into support architecture for test data delivery and algorithmic framework for efficient test scheduling.

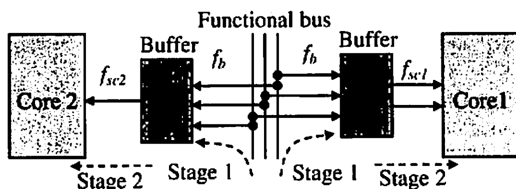

Figure 2: Bus-buffer-module interface

3. Architecture Design Framework

The buffer consists of four main components as shown in Figure 3. The input register latches data from the bus. Upon registering a full status bit for the input register, the top of the stack copies the data from the input register if its status bit indicates that it is empty. After copying, the input register status bit is cleared, preparing it for the next cycle of data from the bus. The stack will subsequently go through fall-through stages which will bring the data to the lowest empty slot.

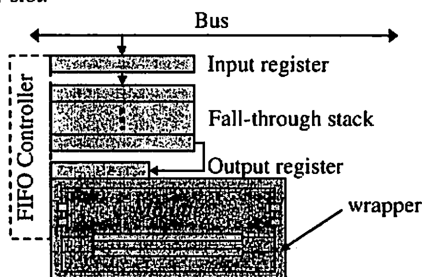

Figure 3: Buffer architecture

The output register is composed of s_m bits, where s_m is the number of wrapper scan chains for module m, possibly differing from the bus width. The output register is interfaced directly to the scan chain inputs. The output register is designed to support this mismatch in bus width and wrapper scan chains. Therefore, it can be easily adapted to any number of scan chains regardless of the bus width.

The test data is serially shifted out from the bottom of the stack, and shifted into the output register. The First-In-First-Out (FIFO) buffer controller keeps track of the number of bits being serially shifted into the output buffer, n_{si}, and the number of bits being serially shifted out of the bottom stack, n_{so}. When $n_{si}=s_m$, the FIFO controller generates a scan clock to scan in the contents of the output buffer into the scan chain, whereupon new data is shifted into the output buffer. When $n_{so}=w_b$, the FIFO controller generates a signal to initiate a fall-through action to fill in the bottom of the stack with new data.

The FIFO controller also keeps track of the number of scan clocks already generated. When this number is equal to the longest scan chain in the module, $max(l_{m,i})$, a capture clock is generated. The FIFO controller can be implemented using three modulo counters, i.e., MOD s_m, MOD w_b and MOD $max(l_{m,i})$ as illustrated in Figure 4. The required input for this circuit is clk_{in}, which is the product of s_m and the scan frequency, f_m.

The proposed buffer architecture offers two distinct advantages. First, the test application at the module operates asynchronously with respect to the availability of test data in the buffer. Because of the asynchronous scan and capture clock generation by the FIFO controller, the buffer can accommodate unpredictable delivery time of the test vectors, thus handling the synchronization issue. As a result, the scan clock and the bus clock can be decoupled. Such decoupling enables the proposed test mechanism to utilize a bus frequency higher than the scan frequency. Such a capability is lacking in a TAM-based approach because TAM wires are connected directly to the scan chains.

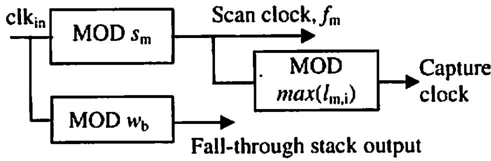

Figure 4: FIFO controller

The second advantage is that the buffer allows the test data to be delivered in chunks of any arbitrary multiple of bus width. This flexibility proves to be quite useful in optimizing the test schedule, in addition to minimizing the buffer area overhead.

4. Algorithmic Framework

The development of the algorithmic framework addresses two main objectives: minimization of the total required buffer size and maximization of bus utilization while ensuring that all modules receive the test data in a timely manner. In order to satisfy these twin objectives, the buffer size and the test delivery sequence need to be optimal.

The framework consists of two hierarchical steps. First step (described in Sections 4.2 and 4.3) is the grouping of modules which can be tested simultaneously under a maximum power constraint. In the second step (defined in Sections 4.4 and 4.5), for each group of modules, the optimum number of packets for every module is determined. Each of these packets is then scheduled for delivery through the functional bus.

In this section, the algorithmic framework is discussed in terms of the two hierarchical steps above. We start by defining a set of nomenclature useful in describing the methodology.

4.1 Terminology

Definition 1: A **test packet** is composed of a number of bits of test data delivered to a module by the processor, in one burst transfer through the functional bus.

Definition 2: **Packet size (p_m)** denotes the number of bits of test data in a test packet. Packet size is typically an integer multiple of the bus width.

Definition 3: A **packet set** is composed of a series of packets delivered to all modules $m_i \in M_G$, where M_G is the set of all modules in the SOC that are tested *simultaneously*. Several identical packet sets can be cascaded to form a **packet schedule**, which includes all packets for all modules m_i to complete the test for M_G.

Definition 4: A module m_i is said to have a **split ratio** of k, if k packets are scheduled for module m_i in one packet set. In other words, it means that module m_i will have k times the number of packets of the smallest modules with a split ratio of one. The module is also called a split-k module.

Definition 5: The **scan rate (R_m)** is the speed at which the test vectors are loaded into the scan chains and the test responses are shifted out of the scan chains in bits per second (bps). For a module which has s_m scan chains and scan frequency f_m, its scan rate is $R_m = s_m \times f_m$.

Definition 6: A **test group** consists of a subset of modules in an SOC that are tested simultaneously.

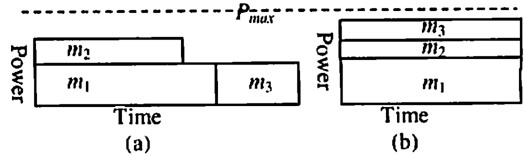

Figure 5: Power-constrained testing

4.2 Effect of Test Frequency on Power Dissipation and Test Application Time

In typical SOC testing, due to the design characteristics such as heat dissipation and current carrying capacity of wires, a limit is imposed on power dissipation that a circuit can tolerate without causing permanent damage to the chip. An illustrative example in Figure 5a shows that module m_3 cannot be tested together with m_1 and m_2 without exceeding P_{max}. However, as shown in Figure 5b, if the rectangles for m_2 and m_3 can be reshaped while keeping the area inside the rectangles constant, all modules can be tested concurrently resulting in shorter total test application time (TAT). The validity for this shape transformation has been discussed in [9].

4.3 Algorithm for Forming Test Groups

The ability to greatly simplify the test program when using the systematic packet scheduling methodology (Section 4.5) is one of the primary differentiators of our test methodology. Therefore, when grouping the modules, we utilize a method that supports this novel aspect to ensure that it can be fully exploited.

A test group is formed by scheduling the module with the longest test time first. When scheduling the next module into the same group, its frequency is reassigned to one of the discrete frequencies, smaller than the maximum test frequency. The smallest frequency that will not cause the module test time to exceed the test time of the first module in the group is selected as it meets the twin goals of not exceeding the maximum frequency while approaching it maximally within the preset flip-flop quantity constraint for the clock divider circuit. When the largest unscheduled module cannot fit the current group within the power constraint, a module that brings total power dissipation for the group closest to the power limit is chosen. This is repeated until no module can fit in, upon which, the same procedure is repeated to create a new group.

4.4 Minimizing Buffer Size

As illustrated in Section 2, splitting test data into smaller test packets reduces the buffer requirement. In principle, there are two ways a packet delivery schedule can be developed. We can specify the complete list of start times for every packet in order to maximize bus utilization and to minimize the time that a module is waiting for test data.

Alternatively, we can specify the start time for a properly chosen subset of packets and then iteratively repeat the same sequence of delivery until all packets are scheduled. The second approach, which we adopt, has the advantage of a shorter test program. In the next section, we take a look at this second approach which we call *packet set scheduling*.

Assuming each packet size is p_{mi}, the buffer size requirement can be specified as [(packet size in bits) –

(number of bits scanned in during the delivery period of the packet)], or

$$B_{m_i} = (p_{m_i} \times w_b) - (\frac{p_{m_i}}{f_b} \times R_{m_i}) \quad (1)$$

where, $R_{mi} = s_{mi} \times f_{mi}$, the scan rate (bps)

f_b = bus frequency (Hz)

p_{mi} = packet size in multiples of w_b (bits)

Equation (1) holds under the assumption that the next packet is delivered only when the previous packet has already been scanned in completely and the scan in operation can commence once the first bit of data arrives in the buffer. The total buffer size, B_{total}, for all modules $m_i \in M$, where M represents all modules in the SOC, is

$$B_{total} = \sum_{m_i \in M} B_{m_i} \quad (2)$$

4.5 Algorithm for Packet Set Scheduling

The packet scheduling algorithm consists of three steps. The first step consists of determining how to split the test packet for each module so that the individual packet sizes are roughly equal. In the second step, once the split ratio has been identified, packet size is determined by solving a set of linear equations. In the third step, a sequence of packet delivery schedules is systematically formed.

Step 1: Let us consider an SOC which has n modules to be tested simultaneously. In the first step of the algorithm, all modules with scan rates smaller than the average scan rate for all modules are considered to have a split ratio of one. Let $k < n$ be the number of split-1 modules. Before proceeding, let us define a terminology.

Definition 7: Assuming that the bus delivery rate is sufficiently high, a packet set is considered to be in **perfect-fit** if (i) it does not have modules that are waiting for test data, (ii) there are no two consecutive packets delivered that belong to the same module and (iii) the number of packets between adjacent split-1 packets are equal. Furthermore, all three conditions need still hold when two adjacent perfect-fit packet sets are cascaded, except possibly for the initial or final legs of test application.

Lemma 1: Let n = number of modules to be tested simultaneously, k = number of split-1 modules and $r > 1$ be a divisor of k. If r is to be the next larger split ratio, then the number of split-r modules must equal $d \times k/r$, for some positive integer d, in order to form a perfect-fit packet set.

Proof: In order to ensure satisfaction of condition (ii) of Definition 7, a minimum of k packets from the remaining n-k modules need to exist for scheduling in between k split-1 packets. Therefore, $n-k \geq k/r$. In order to have an even utilization of bus time, as outlined in condition (iii), we need to schedule the same number of packets in between the delivery of split-1 packets. Therefore, the number of packets needs to be $d \times k$. If each candidate module contributes r packets, then the number of modules required is $(d \times k)/r$. This forms d subgroups of (k/r) split-r modules which make up the split-r group ■

To determine the split-r modules, we iteratively check for all possible values of r, starting with the smallest. Let R_{avg} be the average scan rate of split-1 modules. For the remaining modules with split ratio value unassigned, if there exist k/r modules m_i that fulfill $R_{mi} < 1.5r \times R_{avg}$, then all the k/r modules are assigned split ratio values of r. This process is repeated when identifying the next k/r subgroups of split-r modules. As a result, d subgroups of k/r modules are assigned the split ratio of r.

If no subgroup could be found for the current value of r, this process is repeated for the next larger value of r until $r = k$. Otherwise, the remaining $(n-k-d \times k/r)$ modules are assigned a split ratio of $2k$ to form the split-$2k$ group. The complete packet set schedule can be represented by equation (5), assuming k and q modules for split-1 and split-$2k$ groups, respectively.

Step 2: Once the split ratios are determined, the next step is to determine the packet size for each module. Equation (3) describes the scan in time of a test packet, where w_b is the bus width, p_{mi} and f_{mi} are the packet size and scan frequency, respectively, for module m_i.

$$T_{m_i,p} = w_b \cdot p_{m_i} / f_{m_i} \quad (3)$$

In Figure 6, each $p^g_{i,j}$ represents a test packet where,

g = module number from split-i group
i = split ratio for module g
j = packet number for module g, and $j \leq i$

To preclude introduction of gaps between the test applications of two consecutive packets of a module as illustrated by Figure 1a, the packet TATs multiplied by the corresponding split ratio must be identical as illustrated by Figure 6. Equation (4) describes the packet TAT as illustrated by Figure 6, where r and $2k$ are the corresponding split ratios for each module. Packet size, p_{mi}, and buffer size, B_{mi}, for each module m_i can be calculated by solving equations (1), (2) and (4) simultaneously. A unique solution can be obtained for every value of B_{total}.

Step 3: Equation (5) shows the sequence of packet delivery for one packet set that fulfils the perfect-fit condition in Definition 7. In equation (5), the odd lines show the schedule delivery for q split-$2k$ packets followed by d split-r packets. The even lines show the schedule delivery for the subsequent q split-$2k$ packets followed by a single split-1 packet from one of the k modules.

To retrieve the test response, our methodology simply continues with a burst read every time a burst write is performed when a test packet is delivered. This approach requires minimal overhead on the control algorithm specified by the packet set in equation (5). This differs from the TAM approaches, where test data delivery and response retrieval are performed simultaneously at the expense though of doubling the number of physical pins on the chip.

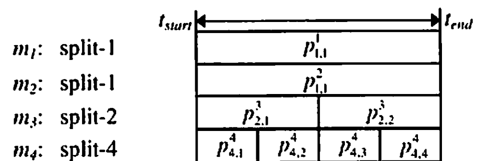

Figure 6: Packet TAT and split ratio

$$\frac{w_b \cdot Pm_1}{fm_1} = \cdots = \frac{w_b \cdot Pm_k}{fm_k} = r \cdot \frac{w_b \cdot Pm_{k+1}}{fm_{k+1}} = \cdots = r \cdot \frac{w_b \cdot Pm_{k+dk/r}}{fm_{k+dk/r}} = 2k \cdot \frac{w_b \cdot Pm_{k+dk/r+1}}{fm_{k+dk/r+1}} = \cdots = 2k \cdot \frac{w_b \cdot Pm_{k+dk/r+q}}{fm_{k+dk/r+q}} \quad (4)$$

$$\left.\begin{array}{cccccccc}
p^1_{2k,1} & p^2_{2k,1} & \cdots & p^q_{2k,1} & p^1_{r,1} & p^{1+k/r}_{r,1} & \cdots & p^{1+(d-1)k/r}_{r,1} \\
p^1_{2k,2} & p^2_{2k,2} & \cdots & p^q_{2k,2} & p^1_{1,1} & & & \\
p^1_{2k,3} & p^2_{2k,3} & \cdots & p^q_{2k,3} & p^2_{r,2} & p^{2+k/r}_{r,1} & \cdots & p^{2+(d-1)k/r}_{r,1} \\
p^1_{2k,4} & p^2_{2k,4} & \cdots & p^q_{2k,4} & p^2_{1,1} & & & \\
\vdots & \vdots & & \vdots & \vdots & & & \\
p^1_{2k,2k-1} & p^2_{2k,2k-1} & \cdots & p^q_{2k,2k-1} & p^{k/r}_{r,r} & p^{k/r+k/r}_{r,r} & \cdots & p^{k/r+(d-1)k/r}_{r,r} \\
p^1_{2k,2k} & p^2_{2k,2k} & \cdots & p^q_{2k,2k} & p^k_{1,1} & & &
\end{array}\right\} \quad (5)$$

5. Experimental Results

In order to evaluate our methodology, we have conducted experiments on several ITC'02 benchmark [10] circuits. Power dissipation information only for h953 is available in the benchmark suite definition. We have additionally obtained power information for p93791 and p22810 from [11] and d695 from [5].

Figure 7 illustrates the effect of the frequency divider resolution (x-axis) on the test application time (y-axis). In each plot, the bottom curve is the test time after the test group has been formed under a power constraint. The higher frequency divider resolution allows us to achieve a shorter test application time. A significant reduction in test time can be achieved within the first four bits of clock divider resolution. The packet scheduling test time (top curve) is always higher as it incurs an additional overhead when splitting the test data into smaller packets.

In order to evaluate the performance of our test scheme, we need to compare with TAM-based test scheduling approaches. No direct comparison can be offered with previous functional test schemes as the experimental results in [8] used four benchmark circuits which do not have the required information such as information on test data and scan chain configurations that are needed.

Table 1 shows the frequency information for TAM approaches and two variations of our approaches, PS[a] and PS[b], with distinct bus frequencies[1].

Table 1: Scan and bus frequency settings

	Scan frequency	Bus frequency
TAM-based	$f_s = F_s$	$f_b = f_s < F_b$
PS[a]	$f_s = F_s$	$f_b = f_s < F_b$
PS[b]	$f_s = F_s$	$f_b = 2 \times f_s < F_b$

Figure 8 shows plots of the TAT for different bus widths. For 64- to 128-bit bus, the TAT is constrained by the largest module; therefore, adding bus widths has no significant effect on test application time. However, for bus widths between 12 and 48 bits, PS[a] delivers improvements of 4.8% and 18.2% over [11] for both maximum power, P_{max}, values of 3,000 and 10,000 for p22810. PS[b] is improved by 25.9% to 47.8% when test data delivery time is the limiting factor. Similar trends can be observed for p93791 in Figures 8c and 8d. In fact, our test methodology delivers marked improvements in reducing test application time for smaller bus widths.

In table 2, the TATs for [5,6,12] are all equal at three P_{max} values for h953 circuit. In our approach, relatively similar results were obtained. These steady results were due to a single dominant module, m_1, that constrains the TAT for this circuit.

For d695 (Table 3), our approach proves to be highly effective, even for the same bus frequency as [5,11], at all power levels for bus widths ranging from 32 to 80 bits. For 96-bit and wider buses, our methodology though fails to perform as well. It is interesting to note, however, that the TAM-based approach requires quite elevated levels of TAM overhead in order to outperform our packet scheduling approach using the functional bus.

Figure 9 shows the trend in TAT (y-axis) under different buffer size utilization (x-axis) for the two circuits with the same power constraints as in Figure 8 and bus width, w_b = 32 bits. The buffer size represents the total size, in multiples of bus width, allocated to all modules in the circuit. It is interesting to note that increasing buffer size only reduces TAT marginally. Therefore, buffer size can be minimized with a small penalty on TAT. For all the experiments reported, the maximum total buffer size constraint is $100 \times w_b$ bits.

With the flexibility of bus frequency selections, unique to our proposed approach as a TAM-based approach is unable to utilize such flexibility, we can further improve the TAT while ensuring that nothing more than minimal bus widths are utilized.

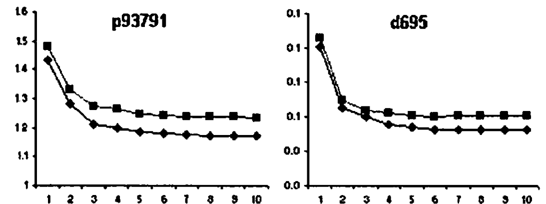

Figure 7: Frequency divider resolution

[1] The scan frequency, f_s, is set to the assumed maximum, F_s = 1.0 MHz; therefore all the TAM-based TATs are divided by 10^6 to convert from the number of clock cycles to time (second).

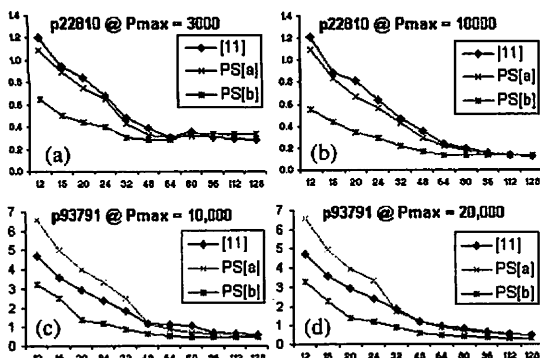

Figure 8: Power-constrained testing for different bus widths for p22810 (a,b) and p93791 (c,d)

6. Conclusions

The utilization of the functional bus for power-constrained core-based SOC testing entails a number of challenges. These include frequency and bit-width mismatch between the bus and the modules under test, allocation of bus time slots for an efficient test data delivery schedule that maximizes bus utilization and that ensures that all modules always have the test data that they need to continue testing simultaneously without exceeding the power constraint.

We have herein proposed an efficient methodology that overcomes all of these challenges through a test support architecture design framework and algorithmic design framework. The proposed methodology offers a solution that also minimizes the size of the test program.

The experimental data clearly showcases the benefits of the proposed methodology in reducing test application time especially for smaller bus widths, while also eliminating the need to add extraneous TAMs to the SOC solely for testing purposes.

Table 2: Power-constrained testing for h953

	Test Time (h953)				
Pmax	**[5]**	**[6]**	**[13]**	**PS[a]**	**PS[b]**
6.0E+09	0.12264	0.12264	0.12264	0.12163	0.12194
7.0E+09	0.11936	0.11936	-	0.12163	0.12231
8.0E+09	0.11936	-	0.11936	0.12316	0.12138

Table 3: Power-constrained testing for d695

	Bus = 32				Bus = 64			
Pmax	[5]	[11]	PS[a]	PS[b]	[5]	[11]	PS[a]	PS[b]
1500	45.56	43.54	**40.53**	**24.60**	27.57	26.97	**24.46**	**23.81**
1800	44.34	42.45	**37.00**	**18.90**	24.45	23.86	**18.80**	**17.05**
2000	43.22	42.45	**37.13**	**19.27**	24.17	21.94	**19.32**	**19.08**
2500	43.22	41.85	**37.34**	**18.83**	23.72	21.93	**18.83**	**13.45**
	Bus = 80				Bus = 96			
1500	**20.91**	24.37	23.69	23.79	**20.91**	23.43	23.33	23.22
1800	20.47	18.77	**17.23**	**16.92**	18.08	18.77	**16.98**	**16.89**
2000	19.21	18.69	**18.01**	**18.52**	17.83	**17.47**	18.06	17.84
2500	19.21	18.69	**15.30**	**13.34**	15.85	17.26	**13.77**	**13.66**
	Bus = 112				Bus = 128			
1500	**16.84**	19.40	23.96	23.75	**16.84**	19.40	23.33	23.10
1800	**14.97**	18.77	16.93	16.95	**14.90**	16.80	16.91	16.83
2000	**14.13**	14.56	17.97	18.68	**14.13**	14.47	17.86	17.92
2500	14.13	13.96	**13.63**	**13.64**	**12.99**	13.39	13.53	13.50

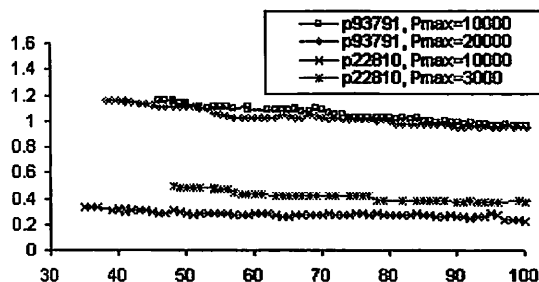

Figure 9: Buffer size vs. TAT

7. Acknowledgments

This work was supported in part by JSPS (Japan Society for the Promotion of Science) under Grants-in-Aid for Scientific Research B (No. 15300018) and in part by 21st Century COE (Center of Excellence) Program (Ubiquitous Networked Media Computing).

8. References

[1] E. J. Marinissen, R. Kapur, M. Lousberg, T. McLaurin, M. Ricchetti and Y. Zorian, "On IEEE P1500 Standard for Embedded Core Test", Journal of Electronic Testing: Theory and Applications, pp. 365-383, Aug. 2002.

[2] E. J. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg, and C. Wouters, "A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores", ITC 1998, pp. 284-293.

[3] T. Ono, K. Wakui, H. Hikima, Y. Nakamura and M. Yoshida, "Integrated and Automated Design-for-Testability Implementation for Cell-based ICs," ATS 1997, pp. 122–125.

[4] A. Sehgal, V. Iyengar, M. D. Krasniewski, and K. Chakrabarty, "Test Cost Reduction for SOCs Using Virtual TAMs and Lagrange Multipliers," In DAC 2003, pp. 738–743.

[5] Y. Huang, S. M. Reddy, W-T. Cheng, P. Reuter, N. Mukherjee, C-C. Tsai, O. Samman, and Y. Zaidan, "Optimal Core Wrapper Width Selection and SOC Test Scheduling Based on 3-D Bin Packing Algorithm", ITC 2002, pp. 74-82.

[6] Y. Xia, M. Chrzanowska-Jeske, B. Wang, and M. Jeske, "Using a Distributed Rectangle Bin-Packing Approach for Core-based SOC Test Scheduling with Power Constraints", ICCAD 2003, pp. 100-105.

[7] J-R. Huang, M. K. Iyer, and K-T. Cheng, "A Self-Test Methodology for IP Cores in Bus-Based Programmable SOCs", VTS 2001, pp. 198-203.

[8] A. Larsson, E. Larsson, P. Eles and Z. Peng, "Optimization of a Bus-based Test Data Transportation Mechanism in System-on-Chip", 8th Euromicro Conference on Digital Systems Design, Aug. 2005, pp. 403-411.

[9] T. Yoneda, K. Masuda, and H. Fujiwara, "Power-Constrained Test Scheduling for Multi-Clock Domain SOCs", Design, Automation and Test in Europe (DATE 2006), To Appear.

[10] E. J. Marinissen, V. Iyengar and K. Chakrabarty, "A Set of Benchmarks for Modular Testing of SOCs", ITC 2002, pp. 519-528.

[11] J. Pouget, E. Larsson, and Z. Peng, "Multiple-Constraint Driven System-on-Chip Test Time Optimization", Journal of Electronic Testing, Vol. 21, 2005, pp. 599-611.

[12] C-P. Su, and C-W. Wu, "A Graph-Based Approach to Power-Constrained SOC Test Scheduling", Journal of Electronic Testing, Vol. 20, 2004, pp. 45-60.