PAPER *Special Section on Test and Verification of VLSIs*

Scheduling Power-Constrained Tests through the SoC Functional Bus

Fawnizu Azmadi HUSSIN[†a)], ***Nonmember***, **Tomokazu YONEDA**[†b)], ***Member***, **Alex ORAILOĞLU**[††c)], ***Nonmember***, ***and*** **Hideo FUJIWARA**[†d)], ***Fellow***

SUMMARY This paper proposes a test methodology for core-based testing of System-on-Chips by utilizing the functional bus as a test access mechanism. The functional bus is used as a transportation channel for the test stimuli and responses from a tester to the cores under test (CUT). To enable test concurrency, local test buffers are added to all CUTs. In order to limit the buffer area overhead while minimizing the test application time, we propose a packet-based scheduling algorithm called *PAcket Set Scheduling (PASS)*, which finds the complete packet delivery schedule under a given power constraint. The utilization of test packets, consisting of a small number of bits of test data, for test data delivery allow an efficient sharing of bus bandwidth with the help of an effective buffer-based test architecture. The experimental results show that the methodology is highly effective, especially for smaller bus widths, compared to previous approaches that do not use the functional bus.

key words: *functional bus, functional TAM, power-constrained, packet-based scheduling, system-on-chip testing*

1. Introduction

System designers are adapting to the system-on-chip (SoC) design methodology because of its efficiency compared to the traditional system-on-board approach. The main benefit of the SoC approach is that it can drastically shorten the design cycle by allowing pre-designed cores and their associated test set to be reused. The IEEE 1500 standard [1] supports this test reuse methodology by standardizing a test wrapper.

The use of SoC design methodology introduces several new problems and challenges in testing [2]. First, the cores that are embedded deep inside the silicon chip require a Test Access Mechanisms (TAM) for test data transportation. Several TAM architectures have been proposed such as TestRail [3], Virtual TAM [4], and TAMs based on transparency [5]. Second, the SoC's core-based design requires a mechanism to isolate the cores during test. This is achieved by the use of core wrappers [1], [3]. Third, the cores can either be tested sequentially at the cost of longer test application time, or in parallel at the cost of larger area overhead and power dissipation. In this regard, various test scheduling solutions have been proposed [6]–[22]. The core test scheduling approaches proposed by [6]–[15] rely on a dedicated TAM. This 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 core under test (CUT).

Regardless of how efficient the test schedule optimization, wrapper optimization, or TAM optimization algorithms are, the idea of adding a dedicated TAM for test data transportation by itself requires considerable area overhead. In addition, the long TAM wires increase the routing congestion. With small feature sizes below 90 nm, the long wires are highly potential spots of production defects. Defects in TAMs would prevent any cores from being properly tested, thereby affecting yield. To avoid relying on TAMs, the existing functional communication architecture should be used as an alternative to the extraneous TAM for testing purposes. The test scheduling methods proposed in [6]–[15] cannot be applied to this new problem, which has a single shared bus where each bus wire cannot be individually assigned according to the optimum test schedule. Cores can only be tested sequentially by using all functional bus bit width.

In order to maximally reuse the existing chip resources for testing, the authors in [16] described a method based on consecutive transparency, where test access paths are formed by creating transparent paths through existing functional connections between the SoC cores, thereby reusing most of the existing interconnects. However, the drawback of the proposed approach is that it is intrusive; sometimes, establishing the paths requires modification to the core internals, which might affect the critical paths of the cores.

Several test strategies which utilize the functional bus [17]–[22] have been proposed; they are further discussed in Sect. 2. However, these methods do not consider the test scheduling at the packet level. In our proposed approach, we explicitly schedule the transportation of data packets through the functional bus that carry the test vectors and responses. The test application times are obtained through complete packet-level simulations of the test data transportation. The proposed approach is unique because of the explicit packet transportation schedule, in addition to its ability to make use of the functional bus effectively.

We begin with a review of some related works in Sect. 2. In Sect. 3, a motivational example is given, followed

Manuscript received April 6, 2007.
Manuscript revised August 3, 2007.

[†]The authors are with the Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan.
[††]The author is with the Computer Science and Engineering Department, University of California San Diego, La Jolla, California 92093, U.S.A.
a) E-mail: fawniz-h@is.naist.jp
b) E-mail: yoneda@is.naist.jp
c) E-mail: alex@cs.ucsd.edu
d) E-mail: fujiwara@is.naist.jp

DOI: 10.1093/ietisy/e91–d.3.736

by a brief technical overview in Sect. 4. In Sect. 5, the support architecture design for the efficient utilization of the functional bus during testing is described. Section 6 elaborates the methodology to develop an efficient test schedule using the functional bus. In Sect. 7, we thoroughly evaluate our methodology experimentally. Finally, a brief set of conclusions is offered in Sect. 8.

2. Related Work

The author in [17] discussed the functional bus based test application for SoCs. When utilizing the embedded processor as tester [18]–[20], direct memory access (DMA) is used to transport test data from the external tester to the embedded memory [18], after which the test data are transported to the CUT through the addressable system bus [19]. In [20], the embedded processor is tested prior to core testing using software-based self-test [23]. The embedded memories are tested by either the processor or the embedded memory built-in-self-test (MBIST).

A hybrid TAM architecture which uses the existing functional bus in addition to extra TAMs was proposed in [8]. In this approach, the functional bus is converted into a bundle of TAMs, by adding some logic to make the bus wires controllable and observable to external testers. As a result, each functional bus wire can be individually controlled and assigned to cores for the test data transportation, similar to added TAM. To further minimize the test application time by optimizing TAM utilization, shared test vectors are broadcast to multiple cores.

In [21], the authors propose a test interface architecture between PCI buses and CUTs. The CUTs are tested using pseudo-random test vectors, generated by the embedded processor. In [22], a buffer interface between a functional bus and a CUT is proposed, while the control of test application is performed by a Finite State Machine (FSM) based controller. The hardwired controller has in [22] has two main weaknesses compared to our approach: (1) the area cost is proportional to the the volume of test data, and (2) the FSM-based test schedule is fixed, making it impossible to change. This flexibility is especially important during the hardware debugging stage. The test responses, on the other hand, are not transported through the bus to the test sink. Instead, they are compressed by local embedded multiple input signature registers (MISR), which could cause aliasing. Furthermore, the MISRs incur hardware overhead.

Due to the use of MISRs, the test application time of [22] should, in most cases, be shorter than our approach, which transports the test responses back to the tester. However, the use of software-based test program in our proposed approach has the advantage of being flexible, and does not incur hardware overhead other than the buffers. The role of the FSM-based test controller in [22] can be replaced by an external tester or an embedded processor. This is further discussed in Sect. 4

To differentiate the two types of test data transportation approaches, we define the following terminologies:

Definition 1: *Dedicated TAM* is a set of dedicated wires that are added to the SoC for the test data transportation between an ATE and all the SoC cores.

Definition 2: *Functional TAM* refers to the existing SoC's functional interconnects which are transformed and reused for the test data transportation.

In this paper, we illustrate our power-constrained SoC testing approach which utilizes the functional TAM for the test data transportation. In order to take advantage of the functional TAM, we approach the problem from two angles, namely, a support architecture design framework and an algorithmic framework. 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 a simplification is attained through the support of an efficient test architecture, which includes appropriate timing control circuitry.

3. Motivation

Let us look at some of the possible scenarios regarding packet based test delivery utilizing the shared functional TAM. To ease description, let us denote each of the small test data units as a *test packet*. Figure 1 illustrate a sequence of events when test packet are transported between a tester and two CUTs, *Core A* and *Core B*, which are interfaced to the functional TAM through dedicated local buffers *Buffer A* and *Buffer B*, respectively, in Fig. 2. *Bus* and *Core A/B* represent the activities on the functional TAM and at the CUT, respectively. Once a packet carrying test vector data, v_t, (labeled V) is received by the local buffer, the test response packet (labeled R) from a previous test vector, v_{t-1} is returned in the next time slot.

The *Round-robin* packet delivery schedule in Fig. 1 is a reasonable first attempt at scheduling the test delivery be-

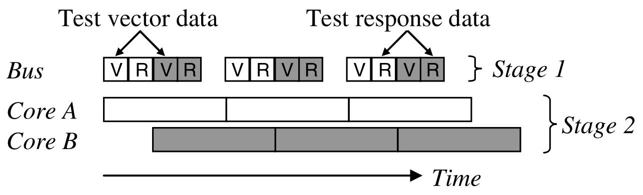


Fig. 1 Test data transportation using packet-based delivery on the functional TAM. A R packet carrying the test response data is returned to tester after every V packet, which carries test vector data.

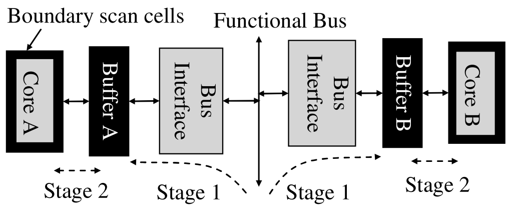


Fig. 2 Buffer-based test architecture enables parallel test application while utilizing a shared functional TAM.

cause of its fair allocation of the bus. Figure 3 (a) shows a similar delivery pattern for three unidentical CUTs m_1, m_2, and m_3. For each CUT m_i, each packet will go through two separate stages of transfer (illustrated by Fig. 2). First, it is delivered from a tester to the buffer through the functional TAM (labeled *Bus* in Fig. 3, where each time slot represents both V and R packets). On the second stage, the packet is transferred from the buffer into the scan chains (labeled m_i in Fig. 3 to illustrate the concurrent activities of all CUTs m_i in stage 2). A test response packet is returned to the tester after every successful reception of a test vector packet by the buffer.

Stage 1 of the subsequent test packet for a CUT can only begin, to avoid buffer overflow, after stage 2 of the previous test packet for that CUT 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 3 (b) shows CUT 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 the 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. CUT 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.

The problem can be remedied by increasing the packet size for m_1 and m_3, as in Fig. 3 (b). 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 sizes for all cores, but the minimum packet size for each core is constrained by the core with the smallest packet size (i.e. m_2). Further reduction in packet sizes for m_1 and m_3 would reintroduce the problem illustrated in Fig. 3 (a). The

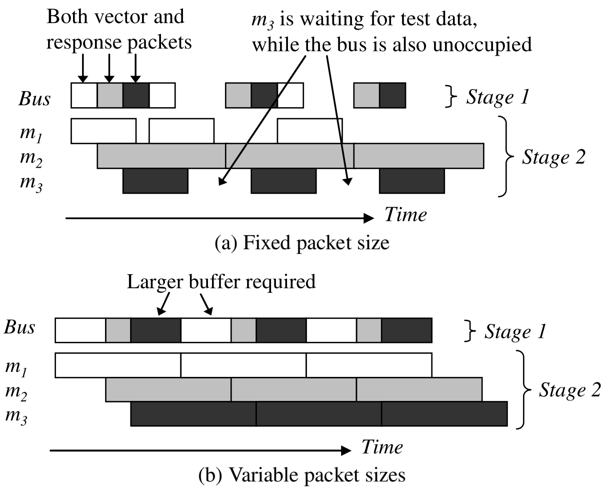

Both vector and response packets
m_3 is waiting for test data, while the bus is also unoccupied
Bus
Stage 1
m_1
m_2
m_3
Stage 2
Time
(a) Fixed packet size
Larger buffer required
Bus
Stage 1
m_1
m_2
m_3
Stage 2
Time
(b) Variable packet sizes

Fig. 3 Effect of repetitive packet delivery sequence for different packet sizes.

two scenarios illustrated by Fig. 3 above are not the only optimization problems that have to be solved to make core-based testing using the functional TAM attractive. 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 the buffer interface between the bus and the core, which is considered in this paper. There are several different variables that can affect test scheduling—bus frequency, bus width, scan frequencies, number of scan chains, and volume of test data for each core. All these variables contribute to the efficiency of the test schedule.

4. Technical Overview

Even though a functional TAM and a dedicated TAM may be similar in many ways, the underlying issues that need to be considered are completely different. We can broadly categorize them into two:

- Support architecture for test data delivery
- Algorithmic framework for efficient test scheduling

A functional TAM differs from a dedicated TAM because every functional TAM wire is connected to every embedded core; it cannot be committed to multiple cores simultaneously like a dedicated TAM by assigning subsets of TAM wires to different cores. Without a buffer-based test support mechanism (Fig. 2) similar to the test buffer in [21], [22], only one core can be tested at a time. Furthermore, in the conventional approach of dedicated TAM-based SoC testing, test control timing, including scan and capture clock generation, is provided by the external ATE. Therefore, the synchronization of test vector availability at the scan input and scan/capture clock generation is trivial since the ATE retains full control of all the event sequences for every core under test. When utilizing the functional TAM, this control timing needs to be performed on-chip, posing a research challenge, which we subsequently address in this paper.

The test source/sink to a functional TAM can either be an external ATE or an internal programmable block. Unlike dedicated TAMs, the ATE cannot be connected directly to the functional TAM wires, but through a test interface port shown in Fig. 4. For example, AMBA [24] bus architecture provides a Test Interface Controller (TIC), which

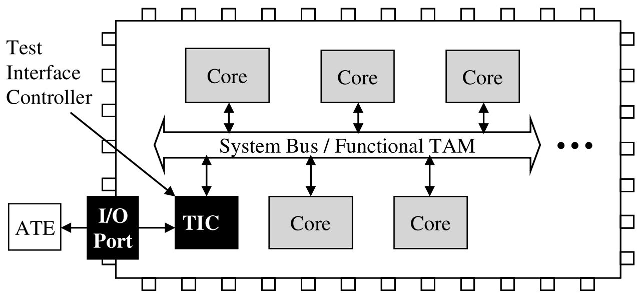


Fig. 4 Interfacing an ATE through a Test Interface Controller (TIC) port for a functional mode test data transfer through the functional TAM.

communicates to the ATE and the CUT using the functional read/write transactions. The TIC relays the vector data packets from the ATE to the CUT and vice versa for the response packets. Several issues need to be resolved in order to utilize the programmable core as a test source/sink, which among others include testing of the programmable core itself, and loading and offloading of the test data to the programmable core. An external ATE (connected through a TIC) is used as the test source/sink, therefore, these issues regarding the programmable core are not addressed in this paper.

Test scheduling that enables the reuse of the functional bus as functional TAMs involves three steps. First, breaking the test set into subsets capable of efficiently utilizing the bus. Second, scheduling all tests for every core with the objective of test time minimization, and third, generation of a test program which will execute in real-time by the tester to perform test application for all the SoC cores. In order for the benefits of utilizing the functional TAM for testing to outweigh its counterparts in the dedicated TAM approach, the test architecture needs to support the algorithmic framework, and vice versa. Otherwise, problems such as bus underutilization arise because of the required arbitration between different cores, and improper test schedules causing certain cores to starve of test data while other cores may be hogging the bus, resulting in prolonged test application time.

Test data overflow/underflow is a particularly serious potential problem, unless the necessary timing synchronization or interlock between the tester and the cores under test is provided. The test support architecture should be designed to also provide timing synchronization, while minimizing the hardware overhead.

5. Test Support Architecture

Buffer based test architecture was proposed by [22] in order to enable concurrency of core-based testing using a shared functional TAM. Our general test architecture with buffers similar to [22] is shown in Fig. 2 with the corresponding test delivery and test application timing diagram similar to Fig. 3.

Figure 5 shows the detailed architecture of the interface between the functional TAM and the core through the functional bus protocol interface. Both the functional connections (solid lines) and design-for-testability (DFT) connections (dotted lines) are shown. The components shown in solid black shading are the proposed buffer-based DFT architecture. Boundary cells (BC) are added to the core PI/POs in order to isolate the core during test. The wrapper scan chains are formed by chaining the input BCs, internal scan chains (ISC), and output BCs, in that precise order. Bidirectional I/Os are treated similarly to the ISCs since the BC's scan inputs, and not functional inputs, are used. Bidirectional I/Os are not shown in Fig. 5 to avoid clutter.

During the test application, the test data are delivered to the input buffer and then scanned into the scan chains. At the same time, the test responses are scanned out and stored

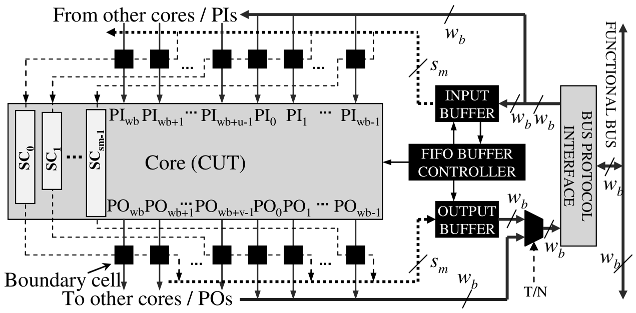
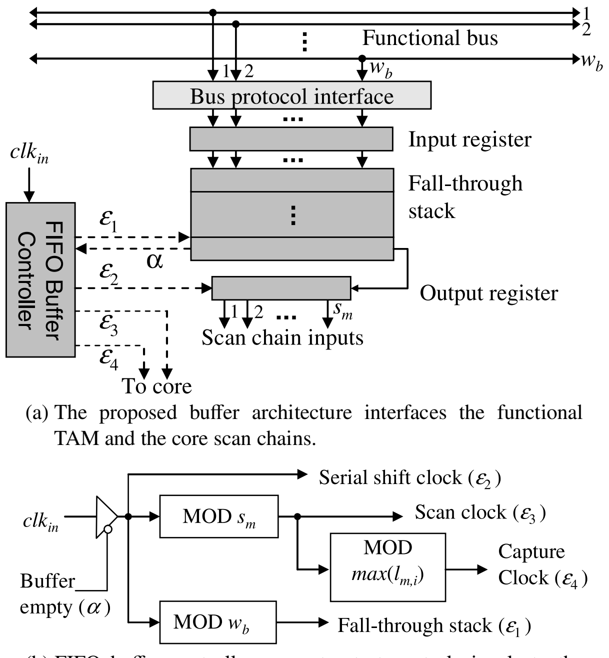


Fig. 5 Core test architecture with input and output buffers to temporarily hold the test vectors and test responses, respectively.



(a) The proposed buffer architecture interfaces the functional TAM and the core scan chains.

(b) FIFO buffer controller generates test control signals to the buffer and the CUT when test data are received.

Fig. 6 Test buffer architecture.

in the output buffer before being retrieved by the tester for analysis.

The buffer consists of four main components—input register, output register, fall-through stack, and FIFO buffer controller—as shown in Fig. 6 (a), illustrating the input buffer and the corresponding first-in first-out (FIFO) buffer controller. The output buffer (not illustrated) has identical structure as the input buffer but with reverse data flow. 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 the fall-through stages which will bring the data to the lowest empty slot.

The output register is composed of s_m bits, where s_m is the number of wrapper scan chains for core m, possibly differing from the bus width, w_b. It 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 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}. Serial shifting is clocked by the tri-stated clock signal (labeled ϵ_2 in Figs. 6 (a) and 6 (b)). When n_{si} equals s_m, the FIFO controller generates a scan clock ϵ_3 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} equals w_b, the FIFO controller generates a signal ϵ_1 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 core, $max(l_{m,i})$ for all i scan chains in core m, a capture clock ϵ_4 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 Fig. 6 (b). The required input for this circuit is clock clk_{in}, whose frequency value is the product of the number of wrapper scan chains, s_m, and the scan frequency, f_m. The same FIFO controller is used for both input and output buffers because of their inverse operation. This eases timing synchronization between the input and output buffers.

The proposed buffer architecture offers two distinct advantages. First, the test application at the core operates asynchronously with respect to the availability of test data in the buffer. When empty, the buffer disables the control signal generation by means of status signal α, which is an input to the FIFO controller. 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 dedicated TAM-based approach because TAM wires are connected directly to the scan chains.

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.

On the other hand, the multiplexer at the output buffer (Fig. 5) introduces one additional gate-delay, compared to the standard IEEE 1500 architecture. However, the impact on functional timing between the functional bus and the core is minimized because the buffers are added in parallel to the functional paths. Another drawback is that gated scan clock is used to disable the scan operation when the buffer is empty, which might affect the delay characteristics of the clock tree.

Due to the DFT architecture, which interfaces the core directly to the functional bus, the proposed methodology is not applicable to embedded cores that are not directly accessible from the bus. One possible way of mitigating this is by introducing bypass interconnects between the core and the bus; this issue is out of the scope of this paper therefore not discussed.

In Sect. 6, the scheduling of test vectors and responses transportation for the embedded cores is discussed. It is assumed that the FIFO buffers and controllers are fault-free, therefore not the target of testing. The test of these DFT architectures can be either done in an integrated fashion, or independent of the core tests (i.e. in priory). In this paper, the latter is assumed.

6. Packet Delivery Scheduling Algorithm

In this section, the issues related to packet delivery scheduling discussed in Sect. 3 are addressed. The packet delivery schedule that minimizes the test application time is developed with two objectives:

1. Minimization of the total required buffer size.
2. Maximization of bus utilization.

The above objectives are sought while at the same time ensuring that all cores receive the test data in a timely manner. In order to satisfy these twin objectives, the buffer size for each core and the test delivery sequence need to be optimal.

The scheduling algorithm consists of two hierarchical steps. The first step (described in Sects. 6.2 and 6.3) is the grouping of cores which can be tested simultaneously under a maximum power constraint. In the second step (defined in Sects. 6.4 and 6.5), for each group of cores, the optimum number of packets (and the corresponding packet size) for every core is determined. Each of these packets is then scheduled for delivery through the functional TAM.

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.

6.1 Terminology

Definition 3: A *test packet* is composed of a number of bits of test data delivered to a core by the tester, in one burst transfer through the bus.

Definition 4: Due to the delivery of test packets through the w_b-bit wide functional TAM, the number of bits of test data that makes up a test packet is typically $p_m \times w_b$, where p_m is denoted as the *packet size*.

Definition 5: A *test group* consists of a subset of cores in an SoC that are tested simultaneously.

Definition 6: A *packet set* is composed of a series of packets delivered to all cores $m_i \in M_G$, where M_G is a test group. Several identical packet sets can be cascaded to form a *packet schedule* consisting of all packets for all cores m_i

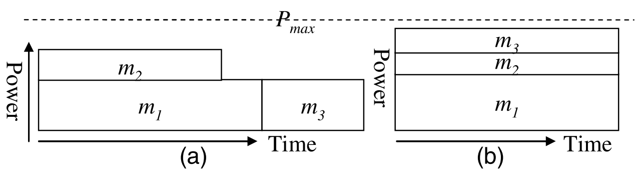
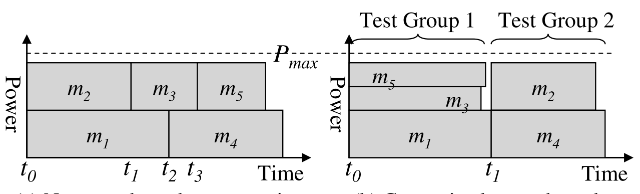


Fig. 7 Power-constrained scheduling with variable test frequencies allows better power utilization to minimize the test application time.



(a) No group boundary constraint (b) Constrained group boundary

Fig. 8 Forming non-overlapping test groups by reducing the test frequencies of cores m_3 and m_5.

to complete the test of M_G. Figure 1 shows a packet set {*Core A, Core B*} repeated three times to form part of a test schedule.

Definition 7: A core m_i is said to have a *split ratio* of k, if k packets are scheduled for core m_i in one packet set. In other words, it means that core m_i will have k times the number of packets of the smallest cores with a split ratio of one. The core is also called a *split-k core*.

Definition 8: The *scan rate* (R_m) is the speed at which the test vectors are loaded into the wrapper scan chains and the test responses are shifted out of the wrapper scan chains in bits per second (*bps*). A core with s_m wrapper scan chains and f_m scan frequency has a scan rate of $R_m = s_m \times f_m$.

6.2 Scan Frequency Reductions

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 Fig. 7 (a) shows that core m_3 cannot be tested together with m_1 and m_2 without exceeding the maximum power dissipation, P_{max}. However, as shown in Fig. 7 (b), if the rectangles for m_2 and m_3 can be reshaped while keeping the area inside the rectangles constant, all cores can be tested concurrently, resulting in shorter total test application time. This power-time rectangle's shape transformation by means of changing the scan frequency has been used and discussed by [13] in their dedicated TAM-based SoC scheduling methodology.

The implementation of the frequency divider is out of the scope of this paper. However, since the use of multi-frequency clocks in IC design is commonplace, we assume that such frequency divider implementation is possible, without causing timing violating. In the proposed algorithm (Sect. 6.3), the choice of frequency values are constrained; these values can be selected based on the actual clock frequencies that can be made available on-chip.

6.3 Forming Non-overlapping Test Groups

The distinctive characteristic of our test scheduling methodology (Sect. 6.5) is that it produces a delivery schedule for only a very small subset of test data packets of every core that can be cycled to produce a complete test data delivery schedule. The small delivery schedule means that a small and simple test program is needed to enumerate the start time for the delivery of each packet. The existing functional TAM approach [22] requires that the delivery time of each data packet to all CUTs be individually specified. For large SoCs, the test program (or the control circuit as proposed by [22]) can be very large.

Therefore, when grouping the cores, we utilize a method—forming non-overlapping test groups (Fig. 8 (b))—that supports this novel aspect to ensure that it can be fully exploited. The grouping in Fig. 8 (b), for an SoC with five CUTs, requires two different test packet delivery schedules which start at time t_i, compared to four for Fig. 8 (a).

A test group is formed by scheduling the core with the longest test time first. When scheduling the next core into the same group, its frequency is reassigned to one of the discrete frequencies smaller than the maximum scan frequency. The smallest frequency that will not cause the core test time to exceed the test time of the first core 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 core cannot fit the current group within the power constraint, a core that brings total power dissipation for the group closest to the power limit is chosen. This is repeated until no core can fit in, upon which, the same procedure is repeated to create a new group. The process is repeated until all cores are assigned to one of the test groups.

6.4 Buffer Sizes

Assuming that each packet starts to be loaded into the scan chains as soon as it arrives at the buffer (i.e. zero buffer loading latency), the required buffer size for a core m_i to store the packet can be specified as [(*packet size in bits*) – (*number of bits loaded into the scan chains during the delivery period of the packet*)], or

$$B_{m_i} = p_{m_i} \cdot w_b - \frac{p_{m_i}}{f_b} \cdot R_{m_i} \tag{1}$$

where p_{m_i} = packet size, w_b = bus width, f_b = bus frequency, and R_{m_i} = scan rate.

Equation (1) holds under the assumption that the next packet is delivered only when the previous packet has already been scanned in completely. Therefore, the total buffer size, B_{total}, is given by Eq. (2) for all cores $m_i \in M$, where M represents all cores under test in the SoC.

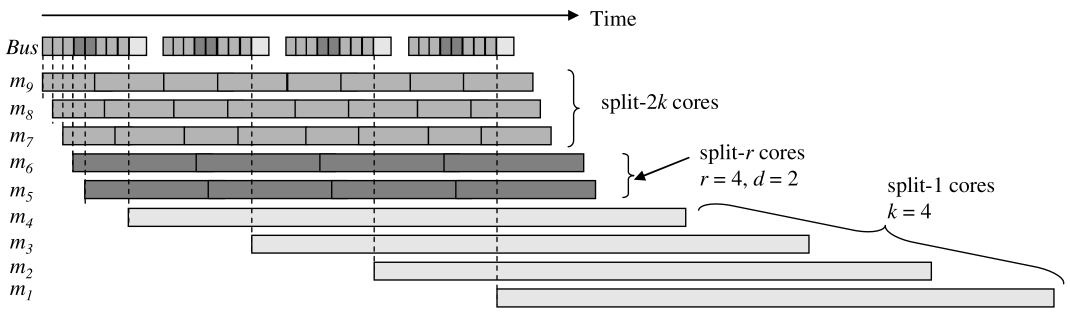


Fig. 9 Packet set scheduling algorithm. An example of a *perfect-fit* delivery sequence.

$$B_{total} = \sum_{m_i \in M} B_{m_i} \tag{2}$$

6.5 PAcket Set Scheduling (PASS) Algorithm

The packet set scheduling algorithm consists of three steps. First, to determine how to split the test packet for each core (i.e. finding the split ratios) so that the individual packet sizes are equal. If the packet sizes are not equal, the largest packet will become the constraint when minimizing the total buffer sizes as illustrated in Sect. 3. In the second step, once the split ratio has been identified, the packet sizes are determined by solving a set of linear equations. In the third step, a sequence of packet set delivery schedules is systematically formed.

6.5.1 Step 1

Let us consider a test group which has n cores to be tested simultaneously. In the first step of the algorithm, all $k < n$ cores with scan rates smaller than the average scan rate for all cores are considered to have a split ratio of one—the smallest split ratio. This is because other larger cores will be assigned split ratios of larger than or equal to one. Under the PASS scheme, the smallest possible number of packets is desirable when forming a packet set in order to minimize the complexity of the resulting test program. Before proceeding, we define a relevant terminology to aid the description of the algorithm.

Definition 9: 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 cores that are waiting for test data, (*ii*) there are no two consecutive packets delivered that belong to the same core 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.

Figure 9 shows a perfect-fit delivery sequence, where the test group consists of nine cores, m_1 to m_9. Between the four split-1 cores ($m_1 - m_4$), eight packets belonging to other cores ($m_5 - m_9$) are delivered (perfect-fit condition (*iii*)). In order to ensure the perfect-fit criteria are not violated when forming a perfect-fit packet set consisting of split-1 and split-r cores, for any value of $r > 1$ such that $k \bmod r$ is zero, the number of split-r cores must equal $d \times k/r$ for some positive integer d. This requirement is imposed in order to achieve an even utilization of bus time, as implied by condition (*iii*) of Definition 9, we need to schedule the same number of split-r packets in between the delivery of split-1 packets. This forms d subgroups of (k/r) split-r cores which make up the split-r group.

To determine the split-r cores, we iteratively check for all possible values of r, starting with the smallest. Let R_{avg} be the average scan rate of split-1 cores. For the remaining cores with split ratio value unassigned, if there exist k/r cores m_i that fulfill $R_{m_i} < \beta r \times R_{avg}$ for some constant β, then all the k/r cores are assigned split ratio values of r. The constant β gives a cut-off limit on the largest core to be assigned to split-r group. It limits the relative buffer sizes of split-r cores and split-1 cores. The value of $\beta = 1.5$ was chosen after thorough experimentation.

The process above is repeated when identifying the next k/r subgroups of split-r cores. As a result, d subgroups of k/r cores 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 equals k. Then, the remaining $q = (n - k - d \times k/r)$ cores are assigned a split ratio of $2k$ to form the split-$2k$ group.

Instead of assigning split ratio of $2k$ for the remaining cores, the same procedure for forming the split-r group can be extended to form other split groups. However, to minimize the algorithm complexity, we have chosen only three split ratio values (1, r, and $2k$) since they provide sufficiently good results (i.e. overall test application time) as illustrated in Sect. 7.

6.5.2 Step 2

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

$$T_{m_i,p} = \frac{w_b \cdot p_{m_i}}{f_{m_i}} \tag{3}$$

To preclude introduction of gaps between the test applications of two consecutive packets of a core (condition (*i*) of

Definition 9) as illustrated by Fig. 3 (b), the packet loading time (i.e. scan in time of a packet worth of test data from the input buffer into the scan chains) multiplied by the corresponding split ratio must be identical as shown by the illustrative example in Fig. 10. This is necessary to ensure that there are no gaps between the packets within the packet set and between adjacent packet sets.

Equation (4) describes the general packet loading times as illustrated by Fig. 10, where r and $2k$ are the corresponding split ratios for each core. Equation (4) assumes that the split-1, split-r, and split-$2k$ cores are labeled m_1 to m_k, m_{k+1} to $m_{k+dk/r}$, and $m_{k+dk/r+1}$ to $m_{k+dk/r+q}$, respectively. For example, given $k = 4, r = 4, d = 2$, and $q = 3$ as in Fig. 9, cores $\{m_1, m_2, m_3, m_4\} \in$ split-1 group, $\{m_5, m_6\} \in$ split-r group, and $\{m_7, m_8, m_9\} \in$ split-$2k$ group, respectively. Packet size, p_{m_i}, and buffer size, B_{m_i}, for each core m_i can be calculated by solving Eqs. (1), (2), and (4) simultaneously. For each value of B_{total}, a unique solution can be obtained.

$$\begin{aligned}
&\frac{w_b \cdot p_{m_1}}{f_{m_1}} = \cdots = \frac{w_b \cdot p_{m_k}}{f_{m_k}} \\
&= r \cdot \frac{w_b \cdot p_{m_{k+1}}}{f_{m_{k+1}}} = \cdots = r \cdot \frac{w_b \cdot p_{m_{k+\frac{dk}{r}}}}{f_{m_{k+\frac{dk}{r}}}} \\
&= 2k \cdot \frac{w_b \cdot p_{m_{k+\frac{dk}{r}+1}}}{f_{m_{k+\frac{dk}{r}+1}}} = \cdots = 2k \cdot \frac{w_b \cdot p_{m_{k+\frac{dk}{r}+q}}}{f_{m_{k+\frac{dk}{r}+q}}} \qquad (4)
\end{aligned}$$

6.5.3 Step 3

In step one, the cores are assigned to either split-1, split-r, or split-$2k$ groups. Once the split ratios are determined, the complete packet set schedule that fulfills conditions (*ii*) and (*iii*) of Definition 9 can be systematically represented by Fig. 11, assuming k and q cores for split-1 and split-$2k$ groups, respectively. Each $p^g_{i,j}$ represents a test packet delivery followed by a response packet retrieval where,

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

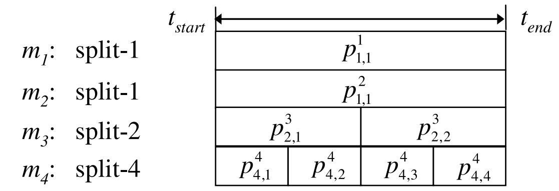


Fig. 10 Equating split ratio × packet loading time for all cores in a test group.

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

Fig. 11 Packet set delivery sequence for a test group with three split groups—split-1, split-r, and split-$2k$.

In Fig. 11, the odd rows (horizontal) show the schedule delivery for q split-$2k$ packets followed by d split-r packets. The even rows show the schedule delivery for the subsequent q split-$2k$ packets followed by a single split-1 packet from one of the k cores. The packet delivery ordering is from left to right and top to bottom.

The retrieval of the response packet is scheduled after every test packet delivery. This can be implemented using the write-read data transfer which requires a single address cycle. If it is not supported by the functional operation, an additional address cycle would be required to initiate the read cycle to fetch the response packet. This approach requires minimal overhead on the control algorithm.

7. Experimental Results

We have conducted experiments on several ITC'02 benchmark [25] circuits in order to verify the efficiency of the proposed algorithm. Since the power dissipation information is not available (except for h953 circuit) in the benchmark suite definition, we obtained power information for p93791 and p22810 from [10] and d695 from [12]. In order to analyze the efficiency of functional TAM utilization for test data delivery, a single shared functional bus is assumed to be connected to every core.

The effect of the frequency divider resolution on the test application time is shown in Fig. 12. In each plot, the bottom curve represents the test application 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 application time (top curves) are always higher as they incur 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 dedicated TAM-based test scheduling approaches. No direct comparison can be offered with previous functional TAM-based test schemes as the experimental results in [22] use four benchmark cir-

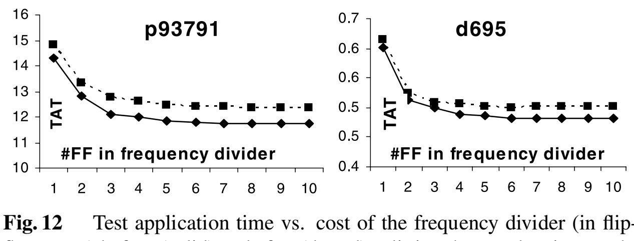


Fig. 12 Test application time vs. cost of the frequency divider (in flip-flop count) before (solid) and after (dotted) splitting the test data into packets.

Table 1 Experimental setup. Same frequency settings for dedicated TAM-based approaches and our PASSa approach, while PASSb uses higher bus frequency.

	Scan frequency	Bus frequency
Dedicated TAM-based	$f_s = F_s$	$f_b = f_s < F_b$
PASSa	$f_s = F_s$	$f_b = f_s < F_b$
PASSb	$f_s = F_s$	$f_b = 2 \times f_s < F_b$

Table 2 Test application time (h953). $B_{total} \leq 100 \times w_b$.

	Test Application Time (ms)				
Pmax	**[12]**	**[9]**	**[11]**	**PASSa**	**PASSb**
6.0E+09	1.22636	1.22636	1.22636	1.21633	1.21942
7.0E+09	1.19357	1.19357	-	1.21633	1.22314
8.0E+09	1.19357	-	1.19357	1.23164	1.21376

cuits which lack required comparison information such as information on test data and scan chain configurations that are needed. The authors of [22] subsequently proposed a method which uses a combination of a functional TAM and a dedicated TAM [8], which is subsequently analyzed and compared.

Table 1 shows the frequency information for dedicated TAM approaches and two variations of our approaches, *PASSa* and *PASSb*, with distinct bus frequencies. The scan frequency, f_s, is set to the assumed maximum, F_s = 100 MHz; therefore all the dedicated TAM-based TATs [9]–[12] are divided by 10^5 to convert from the number of clock cycles to time (millisecond). The bus frequency, f_b, for PASSb is double that of dedicated TAM-based and PASSa approaches (but less than the maximum bus operating frequency, F_b) to illustrate the benefit of our buffer-based approach.

In Table 2, the TATs for [9], [11], [12] are all equal at three P_{max} values for h953 circuit. In our approach, relatively similar results were obtained. No noticeable improvement was achieved when increasing the bus frequency (PASSb). These steady results were due to a single dominant core, m_1, that constrains the TAT minimization for this circuit.

Figure 13 shows plots of the TAT for different bus widths. For 64- to 128-bit bus, the TAT is constrained by the largest core; therefore, increasing bus widths has no significant effect on test application time. However, for bus widths between 12 and 48 bits, PASSa delivers improvements of 4.8% and 18.2% over [10] for both maximum power, P_{max}, values of 3,000 and 10,000 for p22810. PASSb 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 Fig. 13 (c) and 13 (d). In fact, our test methodology delivers marked improvements in reducing test application time for smaller bus widths.

For d695 (Table 3), our approach proves to be highly effective, even for the same bus frequency as [10], [12], 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 dedicated TAM-based approach requires quite elevated levels of TAM overhead in order to outperform our packet scheduling approach using the functional TAM.

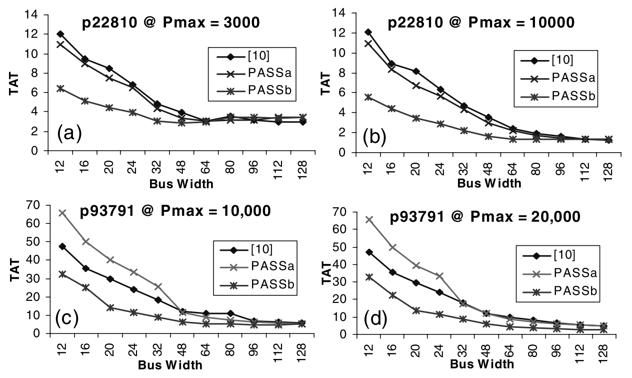


Fig. 13 Test application time (ms) vs. bus width for selected P_{max}. $B_{total} \leq 200 \times w_b$.

Table 3 Test application time (ms) of d695. $B_{total} \leq 50 \times w_b$.

	[12]	**[10]**	**PASSa**	**PASSb**	**[12]**	**[10]**	**PASSa**	**PASSb**
P_{max}	Bus width, b_w = 32				b_w = 48			
1500	0.456	0.435	0.390	0.342	0.310	0.327	0.393	0.325
1800	0.443	0.425	0.390	0.206	0.299	0.321	0.261	0.190
2000	0.432	0.425	0.402	0.233	0.294	0.291	0.294	0.212
2500	0.432	0.418	0.402	0.206	0.290	0.291	0.276	0.153
	b_w = 64				b_w = 80			
1500	0.276	0.270	0.322	0.327	0.209	0.244	0.329	0.327
1800	0.245	0.239	0.207	0.196	0.205	0.188	0.192	0.191
2000	0.242	0.219	0.234	0.213	0.192	0.187	0.224	0.213
2500	0.237	0.219	0.206	0.153	0.192	0.187	0.173	0.153
	b_w = 96				b_w = 112			
1500	0.209	0.234	0.324	0.324	0.168	0.194	0.328	0.328
1800	0.181	0.188	0.190	0.189	0.150	0.188	0.192	0.191
2000	0.178	0.175	0.211	0.210	0.141	0.146	0.214	0.214
2500	0.158	0.173	0.153	0.152	0.141	0.140	0.147	0.152

Table 4 Performance comparison of several testing approaches (d695) [$*P_{max}$=2500; maximum hardware overhead constraint = 300† and 400‡ units].

Methodology	**TAT (ms)**	**Configurations**
Unconstrained [14]	11.0	64-bit dedicated TAM
Hierarchy Constraints [10]	21.9* 20.5	64-bit dedicated TAM
Test Set Sharing and Broadcasting [8]	26.1	64-bit functional bus
	20.4†	64-bit functional bus and 64-bit dedicated TAM (Total = 128 bits)
	18.5‡	
Proposed	18.8*	64-bit functional bus, $f_b = f_s$
	13.5*	64-bit functional bus, $f_b = 2 \times f_s$

In Table 4, some performance comparisons with several scheduling approaches is given. The second row shows the TAT for the dedicated TAM-based scheduling without considering power and hierarchy constraints [14]. The TAT is bounded by the lower bound of $T_{LB} = 10.2\,ms$ [14]. When the design hierarchy is considered as a constraint, the resulting TAT [10] is shown on the third row. On rows 4-6, the TAT of a test set sharing and broadcasting approach is given. When using only the functional TAM (fourth row), the TAT is 27% higher than the case of using only the dedi-

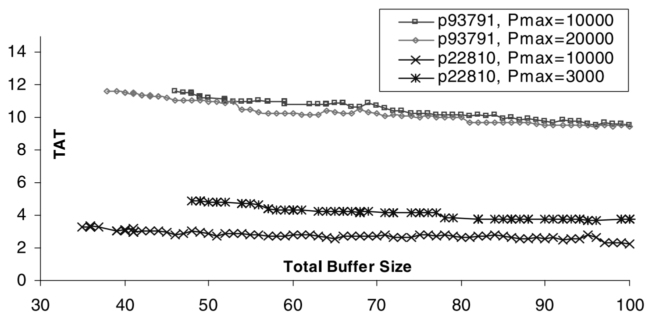


Fig. 14 Little reduction in the test application time (ms) as the total buffer sizes ($\times w_b$ flip-flops) are increased. Bus width, $w_b = 32$ bits.

Table 5 Average buffer sizes per core for the corresponding TAT of PASSa and PASSb in Fig. 13 and Table 3.

Circuit	p22810		p93791		d695			
P_{max}	3000	10000	10000	20000	1500	1800	2000	2500
PASSa	6.66	6.39	3.89	4.31	4.45	3.75	4.00	4.06
PASSb	5.95	6.40	5.03	5.23	4.71	3.78	3.87	3.87

cated TAM [10].

The author improves the performance by using a hybrid architecture with twice the previous bit width (rows 5-6) [8], but comparable hardware overhead due to the same bit width of dedicated TAM as [10]. The last row shows our proposed approach which uses only the functional TAM, with comparable performance when the functional TAM frequency is constrained by the scan frequency. The real advantage is illustrated when a higher functional TAM frequency (not exceeding the maximum functional frequency) is used for test data transportation, which is made possible by the frequency decoupling provided by the buffer architecture.

Figure 14 shows the trend in TAT under different buffer size utilization for the two circuits with the same power constraints as in Fig. 13. The buffer size represents the total size, in multiples of bus width, allocated to all cores in the circuit. It is interesting to note that increasing buffer size only reduces the TAT marginally. Therefore, buffer size can be reduced with only a small penalty on TAT. Table 5 shows the area cost in terms of average buffer sizes per core, averaged over $w_b = 32\ldots128$, for the corresponding TAT results reported in Fig. 13 and Table 3.

With the flexibility of bus frequency selections, unique to our proposed approach as a dedicated 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. This is illustrated by PASSb in Fig. 13 and Table 3.

8. Conclusion

The functional bus is an essential part of any SoC design. As the performance of SoCs increases, the bus speed has also been increased even though not at the same pace. There are many different bus architectures and protocols that provide various functionalities and performance. Therefore, this paper explains our research effort to take advantage of the existing functional bus for testing purposes.

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 cores under test, allocation of bus time slots for an efficient test data delivery schedule that maximizes bus utilization and that ensures that all cores 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 an 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.

Acknowledgements

This work was supported in part by Japan Society for the Promotion of Science (JSPS) under Grants-in-Aid for Scientific Research B (No.15300018) and for Young Scientists (B) (No.18700046). The authors would like to thank Prof. Michiko Inoue, Dr. Satoshi Ohtake and members of Computer Design and Test Laboratory in Nara Institute of Science and Technology for their valuable comments.

References

[1] E.J. Marinissen, R. Kapur, M. Lousberg, T. McLaurin, M. Ricchetti, and Y. Zorian, "On IEEE P1500 standard for embedded core test," J. Electron. Test., Theory Appl., vol.18, pp.365–383, Aug. 2002.

[2] Y. Zorian, E.J. Marinissen, and S. Dey, "Testing embedded-core based system chips," Proc. International Test Conference, pp.130–143, 1998.

[3] 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," Proc. International Test Conference, pp.284–293, 1998.

[4] A. Sehgal, V. Iyengar, M.D. Krasniewski, and K. Chakrabarty, "Test cost reduction for SoCs using virtual TAMs and lagrange multipliers," Proc. 40th Design Automation Conference 2003, pp.738–743.

[5] T. Yoneda and H. Fujiwara, "A DFT method for core-based systems-on-a-chip based on consecutive testability," Proc. Asian Test Symposium, pp.193–198, 2001.

[6] E. Larsson and H. Fujiwara, "System-on-chip test scheduling with reconfigurable core wrappers," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.14, no.3, pp.305–309, March 2006.

[7] A. Sehgal, V. Iyengar, and K. Chakrabarty, "SoC test planning using virtual test access architectures," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.12, no.12, pp.1263–1276, Dec. 2004.

[8] A. Larsson, E. Larsson, P. Eles, and Z. Peng, "SoC test scheduling with test set sharing and broadcasting," Proc. IEEE Asian Test Symposium, pp.162–169, 2005.

[9] 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," Proc. International Conference on Computer Aided Design, pp.100–105, 2003.

[10] J. Pouget, E. Larsson, and Z. Peng, "Multiple-constraint driven system-on-chip test time optimization," J. Electron. Test., Theory

Appl., vol.21, pp.599–611, 2005.

[11] C.P. Su and C.W. Wu, "A graph-based approach to power-constrained SoC test scheduling," J. Electron. Test., Theory Appl., vol.20, pp.45–60, 2004.

[12] 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," Proc. International Test Conference 2002, pp.74–82.

[13] T. Yoneda, K. Masuda, and H. Fujiwara, "Power-constrained test scheduling for multi-clock domain SoCs," Proc. Design, Automation and Test in Europe, pp.297–302, 2006.

[14] S.K. Goel and E.J. Marinissen, "SoC test architecture design for efficient utilization of test bandwidth," ACM Trans. Des. Autom. Electron. Syst., vol.8, no.4, pp.399–429, Oct. 2003.

[15] K. Chakrabarty, V. Iyengar, and M.D. Krasniewski, "Test planning for modular testing of hierarchical SoCs," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.24, no.3, pp.435–448, March 2005.

[16] T. Yoneda and H. Fujiwara, "Design for consecutive testability of system-on-a-chip with built-in self testable cores," J. Electron. Test., Theory Appl., vol.18, no.4/5, pp.487–501, Aug. 2002.

[17] P. Harrod, "Testing reusable IP — A case study," Proc. International Test Conference 1999, pp.493–498.

[18] C.A. Papachristou, F. Martin, and M. Nourani, "Microprocessor based testing for core-based system on chip," Proc. Design Automation Conference 1999, pp.586–591.

[19] S. Hwang and J.A. Abraham, "Reuse of addressable system bus for SoC testing," Proc. IEEE ASOC/SOC Conference 2001, pp.215–219.

[20] A. Krstic, L. Chen, W.C. Lai, K.T. Cheng, and S. Dey, "Embedded software-based self-test for programmable core-based designs," IEEE Des. Test Comput., vol.19, no.4, pp.18–27, July/Aug. 2002.

[21] J.R. Huang, M.K. Iyer, and K.T. Cheng, "A self-test methodology for IP cores in bus-based programmable SoCs," Proc. VLSI Test Symposium 2001, pp.198–203.

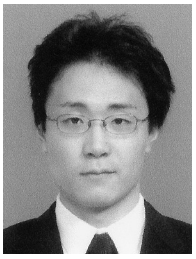
[22] A. Larsson, E. Larsson, P. Eles, and Z. Peng, "Optimization of a bus-based test data transportation mechanism in system-on-chip," Proc. 8th Euromicro Conference on Digital Systems Design, pp.403–409, 2005.

[23] L. Chen and S. Dey, "Software-based self-testing methodology for processor cores," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.20, no.3, pp.369–380, 2001.

[24] D. Flynn, "AMBA: Enabling reusable on-chip designs," IEEE Micro, vol.17, no.4, pp.20–27, July/Aug. 1997.

[25] E.J. Marinissen, V. Iyengar, and K. Chakrabarty, "A set of benchmarks for modular testing of SoCs," Proc. International Test Conference 2002, pp.519–528.

Fawnizu Azmadi Hussin is a Ph.D. student in the Computer Design & Test Laboratory at Nara Institute of Science and Technology. He obtained his B.Sc. in Electrical Engineering, specializing in Computer Design from the University of Minnesota, U.S.A. and subsequently his M.Eng.Sc. in Systems and Control from the University of New South Wales, Australia. His research interests are in VLSI design and testing, especially in the area of System-on-Chip (SoC), multiprocessor SoC, and Network-on-Chip (NoC) based SoC. Fawnizu is a member of the IEEE.



Tomokazu Yoneda received the B.E. degree in information systems engineering from Osaka University, Osaka, Japan, in 1998, and M.E. and Ph.D. degree in information science from Nara Institute of Science and Technology, Nara, Japan, in 2001 and 2002, respectively. Presently he is an assistant professor in Graduate School of Information Science, Nara Institute of Science and Technology. His research interests are VLSI CAD, design for testability, and SoC test scheduling. He is a member of the IEEE Computer Society.

Alex Orailoğlu received his S.B. Degree *cum laude* from Harvard University in Applied Mathematics and his M.S. and Ph.D. degrees in Computer Science from the University of Illinois, Urbana-Champaign. Alex Orailoğlu is currently a Professor of Computer Science and Engineering at the University of California, San Diego. His research interests include Embedded Systems and Processors, digital and analog test, fault tolerant computing, Computer-Aided Design, and nanoelectronics. Professor Orailoğlu serves in the technical, organizing and/or steering committees of the major VLSI Test, Design Automation, Embedded Systems and Computer Architecture conferences and workshops. He is an associate editor of the *Journal of Electronic Test: Theory and Applications*, of the *IEE Digital Systems and Design Journal*, and of the *Journal of Embedded Computing*. Dr. Orailoğlu has published 200 research articles. Dr. Orailoğlu is a Golden Core member of the IEEE Computer Society.

Hideo Fujiwara received the B.E., M.E., and Ph.D. degrees in electronic engineering from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively. He was with Osaka University from 1974 to 1985 and Meiji University from 1985 to 1993, and joined Nara Institute of Science and Technology in 1993. Presently he is a Professor at the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan. His research interests are logic design, digital systems design and test, VLSI CAD and fault tolerant computing, including high-level/logic synthesis for testability, test synthesis, design for testability, built-in self-test, test pattern generation, parallel processing, and computational complexity. He is the author of Logic Testing and Design for Testability (MIT Press, 1985). He received many awards including Okawa Prize for Publication, IEEE CS (Computer Society) Meritorious Service Awards, IEEE CS Continuing Service Award, and IEEE CS Outstanding Contribution Award. He served as an Editor and Associate Editors of several journals, including the IEEE Trans. on Computers, and Journal of Electronic Testing: Theory and Application, and several guest editors of special issues of IEICE Transactions of Information and Systems. Dr. Fujiwara is a fellow of the IEEE, a Golden Core member of the IEEE Computer Society, and a fellow of the IPSJ.