PAPER Special Section on Test and Verification of VLSIs

Test Scheduling for Multi-Clock Domain SoCs under Power Constraint

Tomokazu YONEDA[†a)], ***Member***, **Kimihiko MASUDA**[†*], ***Nonmember***, ***and*** **Hideo FUJIWARA**[†b)], ***Fellow***

SUMMARY This paper presents a power-constrained test scheduling method for multi-clock domain SoCs that consist of cores operating at different clock frequencies during test. In the proposed method, we utilize virtual TAM to solve the frequency gaps between cores and the ATE. Moreover, we present a technique to reduce power consumption of cores during test while the test time of the cores remain the same or increase a little by using virtual TAM. Experimental results show the effectiveness of the proposed method.

key words: *multi-clock domain SoC, test scheduling, test access mechanism, power consumption*

1. Introduction

Today's SoCs embed hundreds of memory cores and several different types of logic cores obtained from various vendors. Moreover, multiple clocks operate at multiple frequencies in a single SoC. Testing of SoCs is a crucial and time consuming problem due to the increasing design complexity. SoCs are increasingly tested in a modular fashion because the system integrator in most cases has very limited knowledge about the structural content of the adopted core, and hence deals with it as a black box. Therefore he/she cannot develop the DFT structures and the corresponding test patterns for it. This is especially true if a core is a hard one or is an encrypted Intellectual Property block [1]. In order to enable modular test, each embedded core should be isolated from its surrounding circuitry. Zorian et al. introduced a generic test architecture that enables modular test for SoCs [1]. It consists of the following three components: 1) test pattern source and test response sink, 2) test access mechanism (TAM), and 3) wrapper. The TAM propagates test patterns for a core from test pattern source to the core, and furthermore propagates the responses from the core to test pattern sink. The wrapper provides an interface between TAM and core, and also provides functions for cores to switch the mode of the cores: 1) normal, 2) INTEST (to test cores), 3) EXTEST (to test interconnects between cores), and 4) BYPASS defined in IEEE std. 1500 [2]. The goal is to develop techniques for wrapper design, TAM design and test schedule that minimizes test application time under given constraints such as the number of test pins and power consumption. A number of approaches have addressed wrapper design [3]–[5] which are IEEE std. 1500 compliant. Similarly, several TAM architectures have been proposed such as *TestBus* [6], [7], *TESTRAIL* [8], *transparency based TAM* [9]–[11]. Moreover, many approaches for core-internal test scheduling problem have been proposed [3], [8], [12]–[16]. Recently, [17] proposed a test scheduling method to minimize the overall test time for core-internal logic and core-external interconnects.

However, these previous approaches are applicable only to single-clock domain SoCs that consist of embedded cores operating at the same clock frequency during test. Today's SoC designs in telecommunications, networking and digital signal processing applications consist of embedded cores operating with different clock frequencies. The clock frequency of some embedded cores during test is limited by its scan chain frequencies. On the other hand, other cores may be testable at-speed in order to increase the coverage of non-modeled and performance-related defects. Moreover, there also exists a frequency gap between each embedded core and ATE used to test the SoC. From these facts, we can say that the previous approaches have the following two problems: 1) when test clock frequency of a core is higher than that of ATE, the ATE cannot provide test sequences at the same speed of the test clock frequency of the core, and 2) when test clock frequency of a core is lower than that of ATE, testing of the core by lowering the frequency of ATE does not make use of ATE capability effectively. Therefore, it is necessary to develop a technique that can solve the above problems for the multi-clock domain SoCs.

Recently, virtual TAM based on bandwidth matching [18] has been proposed in [19] to increase ATE capability when the clock frequency of a core is lower than that of ATE. Xu and Nicolici extended the virtual TAM technique to the multi-frequency TAM design to reduce the test time for the single-clock domain SoCs in [20]. Moreover, a wrapper design for cores with multiple clock domains was proposed in [21]–[24] to achieve at-speed testing of the cores by using virtual TAM technique. However, the test scheduling problem for the multi clock domain SoCs is not addressed in these literatures.

To the best of our knowledge, this paper gives a first discussion and a formulation of the core-internal test scheduling problem for multi-clock domain SoCs. We present a wrapper and TAM design for multi-clock domain SoCs and propose a test scheduling algorithm to minimize

Manuscript received April 9, 2007.
Manuscript revised August 3, 2007.

[†]The authors are with Nara Institute of Science and Technology (NAIST),Ikoma-shi, 630–0192 Japan.
[*]Presently, the author is with SHARP Corporation, Tenri-shi, 632–8567 Japan.
a) E-mail: yoneda@is.naist.jp
b) E-mail: fujiwara@is.naist.jp

DOI: 10.1093/ietisy/e91–d.3.747

test time under power constraint. In the proposed method, we use virtual TAM for each core to solve a frequency gap between each core and a given ATE while the approach in [20] uses a virtual TAM for each test bus (i.e., all the cores assigned to the same test bus must be tested at the same frequency). Therefore, the proposed method in this paper has more flexibility for the test scheduling. Moreover, we also use virtual TAM in order to reduce the power consumption of the cores during test while the test time of the cores remain the same or increase a little. Therefore, the proposed method is effective for the power-constrained test scheduling. Experimental results show the effectiveness of the proposed method.

The rest of this paper is organized as follows. We discuss the power consumption and multi-clock domain SoCs in Sect. 2. Section 3 shows a power-conscious virtual TAM technique. After formulating a test scheduling problem for multi-clock domain SoCs in Sect. 4, we present a power-constrained test scheduling algorithm in Sect. 5. Experimental results are discussed in Sect. 6. Finally, Sect. 7 concludes this paper.

2. Preliminaries

2.1 Power Consumption

Power consumption in CMOS circuits can be classified into two categories: static power and dynamic power. Static power dissipation is caused by leakage or other current drawn continuously from the power supply. On the other hand, Dynamic power dissipation is caused by output switching. For the current CMOS technology, dynamic power is the dominant source of power consumption. High average power consumption causes structural damage to the silicon, bonding wires or package. And if peak power consumption exceeds a certain limit, designers cannot guarantee that the entire circuit will function correctly. It is said that average power consumption is closely related to scan operation while peak power consumption is related to capture operation during test. In this paper, we only consider the power consumption during scan operation defined as follows.

First, the energy $E(k)$ consumed in the circuit on application of consecutive two test vectors (V_{k-1}, V_k) is defined as follows [25].

$$E(k) = 1/2 \cdot c_0 \cdot V_{DD}^2 \cdot \sum_i \cdot S_i(k) \cdot F_i \quad (1)$$

where c_0 is the circuit's minimum parasitic capacitance, V_{DD} is the power supply voltage, $S_i(k)$ is the number of switchings provoked by V_k at node i, and F_i is the number of fanout at node i. Let N be the number of clock cycles for scan operation. The total energy consumed in the circuit during the scan operation is defined as follows.

$$E_{total} = 1/2 \cdot c_0 \cdot V_{DD}^2 \cdot \sum_{k=1}^{N} \sum_i \cdot S_i(k) \cdot F_i \quad (2)$$

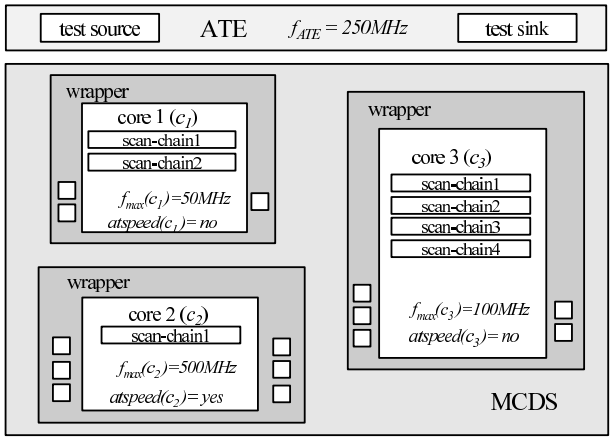


Fig. 1 Multi-clock domain SoC.

Let T be the clock period during scan operation (i.e., a fixed value for T is used during scan operation). Then, the peak power P_{peak} corresponds to the highest energy consumed during one clock period divided by T as follows.

$$P_{peak} = \max_k(E(k))/T \quad (3)$$

The average power P_{ave} corresponds to the total energy divided by the scan shift time as follows.

$$P_{ave} = E_{total}/(N \cdot T) \quad (4)$$

The scan frequency f is defined by $f = 1/T$. Therefore, power consumption during scan operation is proportional to the scan frequency.

2.2 Multi-Clock Domain SoCs

This section describes the formal notation we use to model the multi-clock domain SoC under test. An example of an SoC is shown in Fig. 1 where each core is wrapped to ease test access. Test pattern source and test response sink are implemented off-chip as an ATE. We assumed that an SoC consists of scan-designed cores and each core has single clock frequency during scan operation (i.e., single clock frequency during test data propagation from ATE to the core and from the core to ATE). This assumption is practical even for cores with multiple clock domains. For example, in [21], [26], the authors use single clock frequency during scan in/out operations while multiple clocks are used during capture operation to test delay faults in the circuits with multiple clock domains. In multi-clock domain SoCs, the clock frequency during scan in/out operation for a core can be different from that for other cores because each core has its own scan chain design and the requirement and limitation for the scan in/out frequency can be different from others. In this paper, we consider multi-clock domain SoCs where each core has single clock frequency during scan in/out operation while the frequency during scan in/out operation can be different between cores. In the sequel of this paper, we use the term "test frequency" as the clock frequency during

scan in and out. The multi-clock domain SoC can be modeled as $MCDS = (C, P_{max})$ where:

P_{max} : maximum allowed power consumption;
$C = \{c_1, c_2, \ldots, c_n\}$ is a set of cores;
Each core $c_i \in C$ is characterized by

- $f_{max}(c_i)$: maximum test (scan) frequency;
- $power(c_i)$: power consumption at $f_{max}(c_i)$;
- $atspeed$: at-speed test requirement;
- $R_i = \{r_{i1}, r_{i2}, \ldots\}$ is a set of wrapper designs; Each wrapper design $r_{ij} \in R_i$ is characterized by :
  - $pin(r_{ij})$: number of pins required to test;
  - $cycle(r_{ij})$: number of clock cycles required to test;

For each core, a maximum test frequency and a power consumption at the frequency are given. Each core also has an information about the requirement of at-speed testing. $atspeed(c_i) = yes$ means that c_i must be scanned in/out at $f_{max}(c_i)$ (i.e., the test frequency affects test quality, and we cannot change it for test scheduling). $atspeed(c_i) = no$ means that c_i can be scanned in/out at $f_{max}(c_i)$ or lower frequencies (i.e., the test frequency does not affect test quality, and we can decrease it for test scheduling). Moreover, each core has a wrapper list that consists of possible wrapper designs for the core. Each wrapper design has the number of test pins and the number of clock cycles required to test the core with the wrapper design. The test time for c_i operating at $f_{max}(c_i)$ can be calculated as $cycle(c_i) / f_{max}(c_i)$.

3. Virtual TAM for Power Minimization

In multi-clock domain SoCs, there exist clock frequency gaps between the given ATE and cores during test. The frequency gaps between ATE and cores can be solved by using virtual TAM techniques [19] based on bandwidth matching [18]. Virtual TAMs are based on the following equation between the TAM width and operating frequency.

$$W_{ATE}^{c_i} \cdot f_{ATE} = W_{TAM}^{c_i} \cdot f(c_i) \quad (5)$$

where $W_{ATE}^{c_i}$ and $W_{TAM}^{c_i}$ are the ATE channel width and the TAM width assigned to core c_i, respectively, and f_{ATE} and $f(c_i)$ are the ATE channel frequency and core test frequency (= virtual TAM frequency) respectively. When $f(c_i)$ is higher than f_{ATE} shown in Fig. 2 (a), we insert a test data multiplexing (TDM) circuit between ATE outputs and the core inputs, and multiplex $\lceil f(c_i)/f_{ATE} \rceil \cdot m$ TAM wires at f_{ATE} into m virtual TAM wires at $f(c_i)$. On the other hand, when $f(c_i)$ is lower than f_{ATE} shown in Fig. 2 (b), we insert a test data de-multiplexing (TDdeM) circuit between ATE output and the core inputs, and de-multiplex n TAM wires at f_{ATE} into $\lfloor n \cdot f_{ATE}/f(c_i) \rfloor$ virtual TAM wires at $f(c_i)$. To observe test responses, we need to insert TDM/TDdeM between the core output and ATE inputs in the similar fashion.

In [19], it is observed that virtual TAMs have the following two characteristics. First, TDM and TDdeM are implemented using parallel-in/serial- out registers at the inputs of the cores and serial-in/parallel-out registers at the outputs of the cores. Therefore, the hardware cost is relatively low compared to the area of the core itself. Second, since the TDM and TDdeM used for implementation are placed next to the cores, only the original TAM wires are routed through the SoC. Thus, the routing cost is also low.

In this paper, we also utilize the virtual TAM technique to reduce power consumption of a core while the test time remains the same or increases a little. From the Eqs. (3) and (4), we observe that the power consumption of a core during test can be reduced by lowering its test frequency. However, this increases test time of the core proportionally to the power reduction ratio. Here, we insert TDdeM/TDM circuits between the ATE and the core. Then, more virtual TAM wires become available for the core, and test time can be reduced. In the best case, we can reduce the power

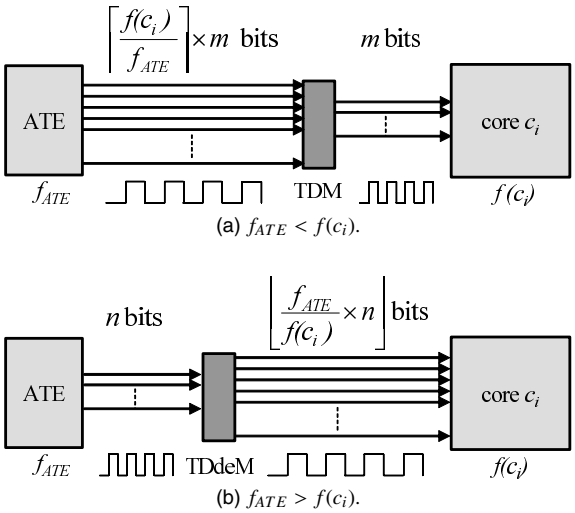


Fig. 2 Test data multiplexing/de-multiplexing.

Table 1 An example of power-conscious virtual TAM for core23 in p93791 (f_{ATE}=200 MHz).

TAM (bits)	$f(core23)$ =200 MHz VTAM (bits)	$f(core23)$ =200 MHz time (μs)	$f(core23)$ =100 MHz VTAM (bits)	$f(core23)$ =100 MHz time (μs)	test time increase (%)
1	1	9184.59	2	9185.95	0.01
2	2	4592.98	4	4754.04	3.51
3	3	3063.09	6	3177.19	3.72
4	4	2377.02	8	2408.74	1.33
5	5	1838.79	10	2006.89	9.14
6	6	1588.60	12	1607.39	1.18
7	7	1387.67	14	1569.79	13.12
8	8	1204.37	16	1205.54	0.10
9	9	1022.20	18	1205.54	17.94
10	10	1003.45	20	1203.19	19.91
11	11	972.90	22	1142.09	17.39
12	12	803.70	24	806.04	0.29
13	13	802.52	26	806.04	0.44
14	14	784.90	28	806.04	2.69
15	15	614.49	30	806.04	31.17
16	16	602.77	32	803.69	33.33

consumption for a core without an increase in test time by using the above power-conscious virtual TAM technique. In this paper, we assume that the power consumption of TDM/TDdeM circuits is negligible from the above observation in [19].

Table 1 shows an example of the power-conscious virtual TAM. In this example, we consider the wrapper design for core23 in p93791 from ITC'02 SoC benchmarks [27]. Columns "time" show the test time for the core when the clock frequency of ATE is 200 MHz. For each TAM width, we can achieve 50% power reduction by decreasing the frequency from 200 MHz to 100 MHz. On the other hand, test time increases by only a few percent in some cases while the other cases increase test time by 10 to 34%. We can observe similar trends for other cores in other SoC. Therefore, for power-constrained test scheduling, test time can be reduced if we select a test frequency for each core effectively.

4. Problem Formulation

We formulate the power-constrained test scheduling problem for multi-clock domain SoCs P_{mcds} that we address in this paper as follows.

Definition 1: P_{mcds}: Given a multi-clock domain SoC $MCDS$, the number of available test pins W_{max} and the clock frequency of ATE f_{ATE}, determine a wrapper design r_i^{test} and test frequency $f(c_i)$ for each core c_i and a test schedule such that

1. the total number of test pins used at any moment does not exceed W_{max},
2. the total power consumption used at any moment does not exceed P_{max},
3. each core satisfies at-speed test requirement (i.e., if $atspeed(c_i) = yes$, c_i must be tested at $f_{max}(c_i)$. Otherwise, c_i can be tested at frequencies lower than $f_{max}(c_i)$), and
4. the overall SoC test time is minimized. □

5. Scheduling Algorithm

This section presents a heuristic algorithm for P_{mcds}. The outline of the proposed algorithm is shown in Fig. 3.

Step 1: Testability Analysis

First, the algorithm checks whether there is a solution for a given problem instance (Line 1). For a core c_i such that $atspeed(c_i) = yes$, we cannot change the test frequency $f_{max}(c_i)$ and power consumption $power(c_i)$ during test. Therefore, there is no solution under the given P_{max} if $power(c_i)$ exceeds P_{max}. Moreover, we have no solution if the ATE cannot provide enough bandwidth for c_i to test at $f_{max}(c_i)$. Now, we summarize the conditions as follows.

For each $c_i \in C$ such that $atspeed(c_i) = yes$, if c_i cannot satisfy the following both two conditions, there is no solution and the algorithm exits. Otherwise, it moves to Step 2.

Procedure: $Schedule(MCDS, W_{max}, P_{max}, f_{ATE})$

```
   /* Step 1 */
 1: Do testability analysis;
   /* Step 2: Lower Bound Calculation */
 2: Compute lower bound T_LB^{c_i} on core test time for each c_i;
 3: Compute lower bound T_LB on SoC test time;
   /* Step 3: Determine Wrapper Design and Test Frequency */
 4: for each core c_i ∈ C do
 5:    Determine wrapper design r_i^test and test frequency f(c_i);
 6: end for
   /* Step 4: Test Schedule at Time 0 */
 7: Set W_0 = W_max, P_0 = P_max, S = C;
 8: while S ≠ φ do
 9:    Select c_i from S in the descending order based on T_LB^{c_i};
10:    if T_LB^{c_i} > T_LB/|C| AND pin(r_i^test) · f(c_i)/f_ATE ≤ W_0 AND
       power(c_i) · f(c_i)/f_max(c_i) ≤ P_0 then
11:       Schedule c_i at time 0 with wrapper design r_i^test and test
          frequency f(c_i);
12:       Update W_0, P_0 and set S = S − {c_i};
13:    else
14:       goto line 17;
15:    end if
16: end while
   /* Step 5: Remaining Power/Pin Distribution */
17: if P_0 > 0 then
18:    Do remaining power distribution;
19: end if
20: if W_0 > 0 then
21:    Do remaining pin distribution;
22: end if
   /* Step 6: Test Schedule for Remaining Cores */
23: while S ≠ φ do
24:    Select c_i from S in the descending order based on T_LB^{c_i};
25:    Find a start time s, wrapper design r_i^test and test frequency
       f(c_i) such that the end time of c_i is minimized;
26:    Schedule c_i at time s and set S = S − {c_i};
27: end while
```

Fig. 3 Outline of the proposed algorithm.

$$P_{max} \geq power(c_i), and \tag{6}$$

$$W_{max} \cdot f_{ATE} \geq \min_j\{pin(r_{ij})\} \cdot f_{max}(c_i) \tag{7}$$

Step 2: Lower Bound Calculation

The authors in [8] proposed an architecture independent lower bounds on core and SoC test time. In this step (Line 2-3), similar lower bounds are calculated for use in the later steps. First, we calculate a lower bound $T_{LB}^{c_i}$ on test time of each core c_i as follows.

$$T_{LB}^{c_i} = \frac{cycle(r_i^{max})}{f_{max}(c_i)} \tag{8}$$

where r_i^{max} is the wrapper configuration of c_i such that $pin(r_i^{max})$ is maximum and r_i^{max} satisfies the following condition.

$$pin(r_i^{max}) \cdot f_{max}(c_i) \leq W_{max} \cdot f_{ATE} \tag{9}$$

Then, we calculate a lower bound T_{LB} on SoC test time as follows.

$$T_{LB} = \max\left(\max_i\{T_{LB}^{c_i}\}, \frac{TotalData}{W_{max} \cdot f_{ATE}}\right) \quad (10)$$

where $TotalData = \sum_i pin(r_i^{min}) \cdot cycle(r_i^{min})$ and r_i^{min} is the wrapper configuration of c_i such that $pin(r_i^{min})$ is minimum.

Step 3: Determine Wrapper Design and Test Frequency
Main idea in this step (Line 4-6) is to lower the test frequencies of cores which are not required to test at-speed in order to increase test concurrency for power-constrained test scheduling in the next step. For each core c_i, we determine a wrapper design r_i^{test} and an integer frequency division factor m_{c_i} such that

(i) $T_{LB} \geq cycle(r_i^{test}) \cdot \dfrac{m_{c_i}}{f_{max}(c_i)}$,

(ii) $W_{max} \cdot f_{ATE} \geq pin(r_i^{test}) \cdot \dfrac{f_{max}(c_i)}{m_{c_i}}$,

(iii) $P_{max} \geq power(c_i) \cdot \dfrac{f_{max}(c_i)}{m_{c_i}}$,

(iv) $m_{c_i} = 1$ if $atspeed(c_i) = yes$,
otherwise, m_{c_i} is maximized, and

(v) $pin(r_i^{test})$ is minimized subject to (iv).

Test frequency $f(c_i)$ for c_i is determined as follows.

$$f(c_i) = \frac{f_{max}(c_i)}{m_{c_i}} \quad (11)$$

In this paper, we assume that the frequency division factor m_{c_i} is an integer. However, to simplify the hardware implementation we can limit m_{c_i} as two's exponent or user-specified frequency set.

Step 4: Test Schedule at Time 0
This step determines test schedule at time 0 (Line 7-16). First, we initialize the available power consumption P_0, available test pins W_0 at time 0 and the set of unscheduled cores S (Line 7). Then, we sort cores in the descending order based on $T_{LB}^{c_i}$ (Line 9). After that, we schedule a core c_i in the above order at time 0 with wrapper r_i^{test} and test frequency $f(c_i)$ (Line 11), and update the corresponding variables (Line 12). This process is repeated until all cores are scheduled or c_i cannot satisfy the conditions at Line 10. The condition $T_{LB}^{c_i} > T_{LB}/|C|$ can prevent us from scheduling cores with small amount of test data to time 0. Instead of scheduling such small cores at time 0, next step tries to reduce the test time of the cores scheduled in this step by distributing the remaining available power and test pins.

Figure 4 shows a current test schedule generated after Step 4. In Fig. 4 (a), the horizontal axis denotes the test time, and the vertical axis denotes the power consumption used in each test time. In Fig. 4 (b), the horizontal axis denotes the test time, and the vertical axis denotes the number of test pin used in each test time.

Step 5: Remaining Power/Pin Distribution at Time 0
There exists a case where P_0 (available power consumption at time 0) does not reach 0 after Step 4 as shown in Fig. 4 (a). This is because Step 4 terminates when one of the three conditions in Line 10 cannot be satisfied. In this step (Line 17-19), we find a core c_i with longest test time among the currently scheduled cores such that

(i) $m_{c_i} \geq 2$, (12)

(ii) $P_0 \geq power(c_i) \cdot \left(\dfrac{1}{m_{c_i}-1} - \dfrac{1}{m_{c_i}}\right)$, and (13)

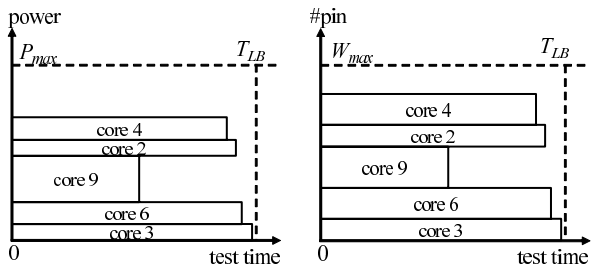
(iii) $\dfrac{P_{max}}{2} \geq \dfrac{power(c_i)}{m_{c_i}-1}$. (14)

If there exists such a core c_i, we update m_{c_i} to $m_{c_i} - 1$, and reduce the test time of c_i by increasing $f(c_i)$ according to Eq. (11) while satisfying power constraint by Eq. (13). Equation (14) can prevent one core from dominating power consumption, and help us to increase the test concurrency when the remaining cores are scheduled in next step (Step 6). This process is repeated while both of the following two conditions are satisfied.

1. $P_0 > 0$, and
2. there exists a core that satisfies all three Eqs. (12), (13) and (14).

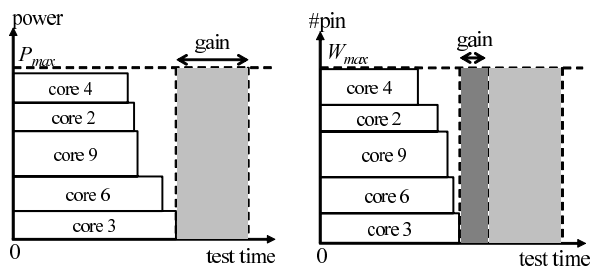
Figure 5 (a) shows a result where we apply this process to the current schedule generated after Step 4 shown in Fig. 4. In this example, frequencies for core 2, 3, 4 and 6 are increased. Consequently, the test time for these cores are reduced.

Similarly, there exists a case where W_0 (available test



(a) Power vs test time. (b) Pin vs test time.

Fig. 4 Test schedule after Step 4.



(a) Power vs test time after recalculating test frequencies. (b) Pin vs test time after redesign wrappers.

Fig. 5 Test schedule after Step 5.

pins at time 0) does not reach 0 after Step 4. In this case, we find a core c_i with longest test time, then assign 1 test pin to c_i and reduce the test time. This process is repeated while $W_0 > 0$ (Line 20-22). Figure 5 (b) shows a result where we apply this process to the current schedule corresponding to Fig. 5 (a).

Step 6: Test Scheduling for Remaining Cores

This step determines test schedules for the remaining unscheduled cores based on BFD heuristic. First, we pick a core c_i in the descending order based on $T_{LB}^{c_i}$ (Line 24). Then, we find a start time, wrapper design r_i^{test} and test frequency $f(c_i)$ for c_i such that the end test time of c_i is minimized as follows (Line 25).

1. Let S be a set of start time candidates that consists of the end time of scheduled cores in the current schedule. For each candidate $s \in S$, we calculate available power consumption P_s and available test pin W_s from the current schedule.
2. For each candidate $s \in S$,
   a. Select a highest test frequency $f_s(c_i)$ such that
   $$\text{(i) } power(c_i) \cdot \frac{f_s(c_i)}{f_{max}(c_i)} \leq P_s.$$
   b. Select a wrapper design $r_{i,s}^{test}$ such that
   $$\text{(i) } pin(r_{i,s}^{test}) \cdot f_s(c_i) \leq W_s \cdot f_{ATE} \text{ and,}$$
   $$\text{(ii) } pin(r_{i,s}^{test}) \text{ is maximized.}$$
   c. Calculate the end time $t_{i,s}$ when c_i starts its test at time s with wrapper $r_{i,s}^{test}$ at frequency $f_s(c_i)$.
3. Schedule c_i at time s with wrapper $r_{i,s}^{test}$ at frequency $f_s(c_i)$ such that
   (i) $t_{i,s}$ is minimized,
   (ii) the test of c_i does not overlap the tests of cores already scheduled in the current schedule.

Figure 6 shows an example of the test scheduling for core 5. Here, a set of start time candidates S consists of five elements: s_1, s_2, s_3, s_4, s_5. For each candidate $s \in S$, we calculate a end time $t_{5,s}$ by determining a test frequency $f_s(c_5)$ and a wrapper design $r_{5,s}^{test}$ shown as a rectangle in Fig. 6. In this example, core 5 is scheduled to start its test at time s_4 with a wrapper $r_{5,4}^{test}$ at frequency $f_4(c_5)$ since the end time $t_{5,4}$ has a minimum value.

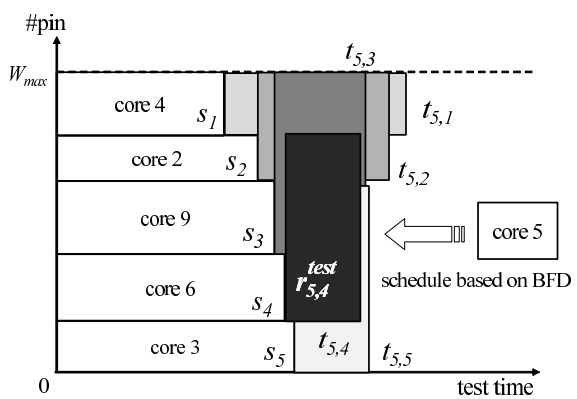


Fig. 6 An example of test scheduling for core 5.

This process is repeated until all the remaining cores are scheduled (Line 23-27). Finally, we can get a complete test schedule.

6. Experimental Results

In Sect. 6.1, we show experimental results for multi-clock domain SoCs with power constraints. Section 6.2 presents experimental results for single-clock domain SoCs with power constraints ("d695" and "h953" from ITC'02 SoC benchmarks [27] because only these two SoCs have power information in the benchmarks) in order to show the effectiveness of our approach compared to previous works. All the experimental results can be obtained within 0.1 sec. on a SunBlade 2000 workstation (1.05 GHz with 8 GB RAM).

6.1 Results for Multi-Clock Domain SoCs

Since there exists no approach that has tackled the test scheduling problem for multi-clock domain SoCs, it is difficult to compare with previous works. We have decided to analyze the trade-offs of the proposed method in terms of the number of available test pin, the clock frequency of ATE, maximum allowed power consumption and test time for two multi-clock domain SoCs. Table 2 shows the multi-clock domain SoC $MCDS_1$ used in this experiment. This SoC consists of 14 cores. First 10 cores are from "d695" in ITC'02 SoC benchmarks [27]. "flexible($\geq$ 2)" in column "wrapper list" denotes that we can design any wrapper (wrapper with any number of test pins) by the procedure proposed in [3], [4]. We use the same power consumption shown in [14], and assume that $f_{max}(c_i) = 50$ MHz and $atspeed(c_i) = no$ for these 10 cores. The wrappers for core 11 and core 12 are already designed (i.e., 64 wrapper pins and 472 test cycles for core 11, 32 wrapper pins and 782 test cycles for core 12, respectively). We assume that these two

Table 2 An multi-clock domain SoC $MCDS_1$.

core	at-speed requirement	wrapper list (pins)	test freq. (MHz)	power (unit)
1	no	flexible($\geq$ 2)	50	660
2	no	flexible($\geq$ 2)	50	602
3	no	flexible($\geq$ 2)	50	823
4	no	flexible($\geq$ 2)	50	275
5	no	flexible($\geq$ 2)	50	690
6	no	flexible($\geq$ 2)	50	354
7	no	flexible($\geq$ 2)	50	530
8	no	flexible($\geq$ 2)	50	753
9	no	flexible($\geq$ 2)	50	641
10	no	flexible($\geq$ 2)	50	1144
11	yes	fixed (64)	100	480
12	yes	fixed (32)	200	940
13	no	flexible($\geq$ 2)	20	212
14	no	flexible($\geq$ 2)	25	345

Table 3 Test time results [μs] for multi-clock domain SoC $MCDS_1$ ($1 \le m_{c_i} \le 8$).

f_{ATE}	P_{max}	W_{max}								
		32 pin			64 pin			128 pin		
		Case1 m_{c_i} = integer	Case3 m_{c_i} = 1 (diff1)	T_{LB} (diff1)	Case1 m_{c_i} = integer	Case3 m_{c_i} = 1 (diff1)	T_{LB} (diff1)	Case1 m_{c_i} = integer	Case3 m_{c_i} = 1 (diff1)	T_{LB} (diff1)
200 MHz	1000	637.13	UT		632.41	UT		632.41	UT	
	2000	325.29	375.65 (15.5 %)		301.08	321.52 (6.8 %)		301.08	321.52 (6.8 %)	
	3000	312.09	318.73 (2.1 %)	295.27 (-5.4 %)	204.20	209.42 (2.6%)	204.20 (0.0 %)	204.20	209.42 (2.6 %)	204.20 (0.0 %)
100 MHz	1000	UT	UT		637.13	UT		632.41	UT	
	2000	UT	UT		325.29	375.65 (15.5 %)		301.08	321.52 (6.8 %)	
	3000	UT	UT		312.09	318.73 (2.1 %)	295.27 (-5.4 %)	204.20	209.42 (2.6 %)	204.20 (0.0 %)
50 MHz	1000	UT	UT		UT	UT		637.13	UT	
	2000	UT	UT		UT	UT		325.29	375.65 (15.5 %)	
	3000	UT	UT		UT	UT		312.09	318.73 (2.1 %)	295.27 (-5.4 %)

Table 4 Test time results [μs] for $p93791$ when $f_{ATE} = 50\,MHz$ ($1 \le m_{c_i} \le 8$).

P_{max}	W_{max}											
	32 pin				64 pin				128 pin			
	Case1 m_{c_i} = integer	Case2 $m_{c_i} = 2^x$ (diff1)	Case3 m_{c_i} = 1 (diff1)	T_{LB} (diff1)	Case1 m_{c_i} = integer	Case2 $m_{c_i} = 2^x$ (diff1)	Case3 m_{c_i} = 1 (diff1)	T_{LB} (diff1)	Case1 m_{c_i} = integer	Case2 $m_{c_i} = 2^x$ (diff1)	Case3 m_{c_i} = 1 (diff1)	T_{LB} (diff1)
15000	36385	36430 (0.1 %)	UT		20381	20506 (0.6 %)	UT		9712	10562 (8.8 %)	UT	
30000	36227	36227 (0.0 %)	36511 (0.8 %)		19086	19992 (4.7 %)	20430 (7.0 %)		9604	10033 (4.5 %)	10401 (8.3 %)	
50000	36227	36227 (0.0 %)	36480 (0.7 %)	34988 (-3.4 %)	18420	19992 (8.5 %)	20411 (10.8 %)	17494 (-5.0 %)	9444	9903 (4.9 %)	10401 (10.1 %)	8747 (-7.4 %)

cores are tested at higher frequencies than other cores, and $atspeed(c_i) = yes$. Core 13 and core 14 are copies of core 7 and core 5, respectively. However, we assume that these two cores are tested at lower frequencies than other cores.

Table 3 shows test time results when $f_{ATE} = 200$ MHz, 100 MHz and 50 MHz for $MCDS_1$. We did experiments for the following three cases with respect to the integer frequency divisor m_{c_i} used in the proposed algorithm: (1) m_{c_i} is integer, (2) m_{c_i} is 2's exponent and (3) $m_{c_i} = 1$ (i.e., it is the same as the case we set $atspeed(c_i) = yes$ for all cores). However, the results for Case2 are identical for Case1 and we did not include in the table. Column "T_{LB}" denotes the power-independent theoretical lower bound on test time defined by Eq. (10). In this table, the test time results are shown as "μsec." and the number in parentheses denotes the test time increase relative to Case1. "UT" denotes that there exists no solution for the given parameters. In this SoC, since core 11 should be tested at 100 MHz with 64 pins, we observe that there exists no solution for three cases: 1) $f_{ATE} = 100$ MHz and $W_{max} = 32$, 2)$f_{ATE} = 50$ MHz and $W_{max} = 32$, and 3) $f_{ATE} = 50$ MHz and $W_{max} = 64$. We also observe that test time depends on the product of f_{ATE} and W_{max}. Therefore, when we use a high speed ATE, we can test SoCs with small number of test pins. On the other hand, even when we use a low speed ATE, we can achieve the same test time by using more test pins. From this results, the designer can decide the number of test pins and the speed of the test pin considering the total cost for them. Moreover, we can observe the effectiveness of lowering test frequencies by comparing Case1 with Case3. We can obtain savings in test time up 15% by lowering test frequencies. The difference from power-independent lower bound T_{LB} is at most only 5.4% when $P_{max} = 3000$. Therefore, we can say that the proposed heuristic algorithm is effective and efficient.

Table 4 shows the test time results for another multi-clock domain SoC $p93791$ from ITC'02 SoC benchmarks [27] when $f_{ATE} = 50\,MHz$. As the original benchmark SoC do not have any data related to power consumption, we used the following settings for each core c_i : (1)

Table 5 Test time results (# cycles) for single-clock domain SoCs.

SoC	P_{max}	W_{max}								
		32 pin			64 pin			128 pin		
		3D [14]	EA [15]	proposed	3D [14]	EA [15]	proposed	3D [14]	EA [15]	proposed
d695	1000	NA	NA	**44528**	NA	NA	**27482**	NA	NA	**24707**
	1500	45560	-	**42981**	27573	-	**22690**	16841	-	**16239**
	2000	43221	-	**42632**	24171	-	**21838**	14128	-	**12753**
	2500	43221	-	**42564**	23721	-	**21616**	12993	-	**11180**
h953	5×10^9	NA	NA	**119357**	NA	NA	**119357**	NA	NA	**119357**
	6×10^9	122636	122636	**119357**	122636	122636	**119357**	122636	122636	**119357**
	7×10^9	**119357**	**119357**	**119357**	**119357**	**119357**	**119357**	**119357**	**119357**	**119357**

$f_{max}(c_i) = 50\,MHz$, (2) $atspeed(c_i) = no$ and (3) $power(c_i)$ is the total number of scan FFs in c_i. If we limit m_{c_i} to 2's exponent, test time is increased up to 8.8%. Test time is further increased up to 10.8% by limiting m_{c_i} to 1. Finally, the difference from T_{LB} is at most only 7.4% when $P_{max} = 50000$, and it shows the effectiveness and efficiency of the proposed algorithm.

6.2 Comparison with Other Approaches

In order to show the effectiveness of our approach compared to previous works, we present experimental results for the single-clock domain SoCs with power constraint. We use "d695" and "h953" from ITC'02 SoC benchmarks [27] as the single-clock domain SoCs by assuming that $f_{ATE} = 50$ MHz, and $f_{max}(c_i) = 50$ MHz and $atspeed(c_i) = no$ for all core $c_i \in C$. This is because only these two SoCs have power information in the benchmarks (for "d695", we use the same power consumption shown in [14]). Table 5 shows the test time results of the proposed method and the previous power-constrained approaches [14], [15] which are applicable only to the single-clock domain SoCs. In this table, test time results are shown as the number of clock cycles at 50 MHz. "NA" denotes that the approach is not applicable for the constraint. "-" denotes that no result is shown for the constraint in the approach. For d695, we observe that the proposed approach can achieve a 6.9% reduction in average test time compared to [14]. For h953, we observe that the proposed approach can achieve the lower bound (119357) on the SoC test time [8] under all power constraints. Moreover, in both SoCs under tight power constraints ($P_{max} = 1000$ for d695, $P_{max} = 5 \times 10^9$ for h953), only the proposed method can provide a solution. This is because only the proposed method can lower the test frequencies and reduce the power consumption during test. From these results, we conclude that the proposed power-conscious virtual TAM technique and test scheduling algorithm are effective.

7. Conclusions

This paper has proposed a power-constrained test scheduling method for multi-clock domain SoCs. To the best of our knowledge, a test scheduling problem for multi-clock domain SoCs has been addressed and formulated for the first time in this paper. Moreover, we have proposed a technique to reduce power consumption of cores during test while the test time of the cores remain the same or increase a little by utilizing virtual TAMs. The experimental results showed that the proposed test scheduling method can achieve short test time compared to the previous power-constrained test scheduling methods.

Acknowledgments

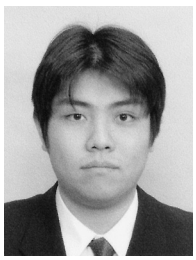
This work was supported in part by Japan Society for the Promotion of Science (JSPS) under Grants-in-Aid for Scientific Research B(No. 15300018) and for Young Scientists B(No.18700046). The authors would like to thank Prof. Kewal K. Saluja, Prof. Michiko Inoue, Dr. Satoshi Ohtake and members of Computer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

References

[1] Y. Zorian, E.J. Marinissen, and S. Dey, "Testing embedded-core based system chips," Proc. International Test Conference, pp.130–143, Oct. 1998.
[2] "IEEE standard testability method for embedded core-based integrated circuits." IEEE Std 1500-2005, 2005.
[3] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Test wrapper and test access mechanism co-optimization for system-on-chip," J. Electron. Testing Theory and Appl. (JETTA), vol.18, no.2, pp.213–230, April 2002.
[4] W. Zou, S.R. Reddy, I. Pomeranz, and Y. Huang, "SOC test scheduling using simulated annealing," Proc. VLSI Test Symposium, pp.325–329, May 2003.
[5] E.J. Marinissen, S.K. Goel, and M. Lousber, "Wrapper design for embedded core test," Proc. International Test Conference, pp.911–920, Oct. 2000.
[6] T. Ono, K. Wakui, H. Hikima, Y. Nakamura, and M. Yoshida, "Integrated and automated design-for-testability implementation for cell-based ICs," Proc. Asian Test Symposium, pp.122–125, Nov. 1997.
[7] P. Varma and S. Bhatia, "A structured test re-use methodology for core-based system chips," Proc. International Test Conference, pp.294–302, Oct. 1998.
[8] S.K. Goel and E.J. Marinissen, "Effective and efficient test architecture design for SOC," Proc. International Test Conference, pp.529–538, Oct. 2002.
[9] M. Nourani and C.A. Papachristou, "Structural fault testing of embedded cores using pipelining," J. Electron. Testing Theory and Appl. (JETTA), vol.15, no.1/2, pp.129–144, Aug.–Oct. 1999.
[10] S. Ravi, G. Lakshminarayana, and N.K. Jha, "Testing of core-based systems-on-a-chip," IEEE Trans. Comput.-Aided Des. Integr. Cir-

cuits Syst., vol.20, no.3, pp.426–439, March 2001.
[11] T. Yoneda and H. Fujiwara, "Design for consecutive testability of system-on-a-chip with built-in self testable cores," J. Electron. Testing Theory and Appl. (JETTA), Special Issue on Plug-and-Play Test Automation for System-on-a-Chip, vol.18, no.4/5, pp.487–501, Aug. 2002.
[12] Y. Huang, W.T. Cheng, C.C. Tsai, N. Mukherjee, O. Samman, Y. Zaidan, and S.M. Reddy, "Resource allocation and test scheduling for concurrent test of core-based SOC design," Proc. Asian Test Symposium, pp.265–270, Nov. 2001.
[13] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "On using rectangle packing for SOC wrapper/TAM co-optimization," Proc. VLSI Test Symposium, pp.253–258,, April 2002.
[14] Y. Huang, N. Mukherjee, S. Reddy, C. Tsai, W.T. Cheng, O. Samman, P. Reuter, and Y. Zaidan, "Optimal core wrapper width selection and SOC test scheduling based on 3-dimensional bin packing algorithm," Proc. International Test Conference, pp.74–82, Oct. 2002.
[15] Y. Xia, M.C. Jeske, B. Wang, and M. Jeske, "Using distributed rectangle bin-packing approach for core-based SoC test scheduling with power constraints," Proc. International Conference on Computer-Aided Design, pp.100–105, Nov. 2003.
[16] E. Larsson, K. Arvidsson, H. Fujiwara, and Z. Peng, "Efficient test solutions for core-based designs," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.23, no.5, pp.758–775, May 2004.
[17] Q. Xu, Y. Zhang, and K. Chakrabarty, "SOC test architecture optimization for signal integrity faults on core-external interconnects," Proc. Design Automation Conference, pp.676–681, June 2007.
[18] A. Khoche, "Test resource partitioning for scan architectures using bandwidth matching," Digest of International Workshop on Test Resource Partitioning, pp.1.4–1–1.4–8, 2001.
[19] 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.
[20] Q. Xu and N. Nicolici, "Multi-frequency test access mechanism design for modular SOC testing," Proc. Asian Test Symposium, pp.2–7, Nov. 2004.
[21] Q. Xu and N. Nicolici, "Wrapper design for multi-frequency IP cores," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.13, no.6, pp.678–685, June 2005.
[22] Q. Xu, N. Nicolici, and K. Chakrabarty, "Multi-frequency wrapper design and optimization for embedded cores under average power constraints," Proc. Design Automation Conference, pp.123–128, June 2005.
[23] D. Zhao, U. Chandran, and H. Fujiwara, "Shelf packing to the design and optimization of a power-aware multi-frequency wrapper architecture for modular IP cores," Proc. Asia and South Pacific Design Automation Conference, pp.714–719, Jan. 2007.
[24] T.E. Yu, T. Yoneda, D. Zhao, and H. Fujiwara, "Using domain partitioning in wrapper design for IP cores under power constraints," Proc. VLSI Test Symposium, pp.369–374, May 2007.
[25] P. Girard, "Survey of low-power testing of VLSI circuits," IEEE Des. Test Comput., vol.19, no.3, pp.82–92, May/June 2002.
[26] K. Hatayama, M. Nakao, and Y. Sato, "At-speed built-in test for logic circuits with multiple clocks," Proc. Asian Test Symposium, pp.292–297, Nov. 2002.
[27] E.J. Marinissen, V. Iyengar, and K. Chakrabarty, "A set of benchmarks for modular testing of SOCs," Proc. International Test Conference, pp.519–528, Oct. 2002.

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 testing. He is a member of the IEEE Computer Society.



Kimihiko Masuda received the B.E. degree in engineering from Kyoto Institute of Technology, Kyoto, Japan, in 2003, and M.E. degree in information science from Nara Institute of Science and Technology, Nara, Japan, in 2005. Presently he works for SHARP Corporation.

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. In 1981 he was a Visiting Research Assistant Professor at the University of Waterloo, and in 1984 he was a Visiting Associate Professor at McGill University, Canada. 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 the IEICE Young Engineer Award in 1977, IEEE Computer Society Certificate of Appreciation Award in 1991, 2000 and 2001, Okawa Prize for Publication in 1994, IEEE Computer Society Meritorious Service Award in 1996, and IEEE Computer Society Outstanding Contribution Award in 2001. He is an advisory member of IEICE Trans. on Information and Systems and an editor of IEEE Trans. on Computers, J. Electronic Testing, J. Circuits, Systems and Computers, J. VLSI Design and others. Dr. Fujiwara is a fellow of the IEEE, a Golden Core member of the IEEE Computer Society and a member the Information Processing Society of Japan.