IEEE 7th Workshop on RTL and High Level Testing (WRTLT'06), pp. 43-48, November, 2006.

Designing Power-aware Wrappers for Multi-clock Domain Cores Using Clock Domain Partitioning

Thomas Edison Yu[†], Tomokazu Yoneda[†], Danella Zhao[‡] and Hideo Fujiwara[†]
[†] Graduate School of Information Science, Nara Institute of Science and Technology
Kansai Science City, 630–0192, Japan
[‡] The Center For Advanced Computer Studies, University of Louisiana at Lafayette
Lafayette, LA, 70504
[†] E–mail:*{tomasu-y, yoneda, fujiwara}@is.naist.jp*
[‡] E–mail:*{dzhao}@cacs.louisiana.edu*

Abstract

This paper presents a method for designing power-aware test wrappers for embedded cores with multiple clock domains. We make use of partitioning of the clock domains into smaller sub-domains in combination with bandwidth conversion, multiple shift frequencies and gated-clocks to achieve greater flexibility when determining an optimal test schedule under tight power constraints.

Keywords:

wrapper design, multi-clock domain, embedded core testing, test scheduling

1 Introduction

The rapid advancement in the design and production of VLSI chips has made it possible to put entire systems onto a single chip which is commonly known as System-on-Chip (SoC). The increased complexity of SoC circuitry means an increase in the amount of test data which usually results in longer test application time. Furthermore, test access becomes a problem since the cores cannot be directly accessed from the I/O pins of the chip. The most common DFT (Design-for-Testability) used for SoCs with multiple cores is the design of a test data delivery method, more commonly known as TAM (Test Access Mechanism), and the use of *wrappers* which isolate cores under test. Recently, the IEEE 1500 standard for embedded core test has been approved to provide guidelines for core wrapper design and interfacing to TAMs. Several approaches to optimize wrapper designs for single frequency embedded core testing [1, 2,] as well as wrapper and TAM co-optimization algorithms [3, 4, 13, 14] have already been suggested. Still, most of these approaches don't directly address the testing of modern multi-clock domain IP cores which operate at various frequencies internally and have advantages such as reduced power and silicon area. Subsequently, multi-clock domain cores present clock skew and at-speed testing problems which must be considered during testing. Furthermore, power consumption during test has become a big issue because of high switching activity during scan-shift operations.

Most at-speed multi-clock domain core testing techniques that have been proposed are based on BIST [5, 6, 7] and utilizing techniques such as programmable capture windows [5] and directly controlling separate launch and capture clocks [6] to solve the clock-skew problems while still allowing at-speed testing.

The first non-BIST based multi-clock domain core wrapper design for IP-protected cores was proposed in [8], where the core was divided into its clock domains, referring to each domain as a *virtual core*. Single frequency wrapper design was performed on each virtual core to assign a *virtual wrapper* to each of them. *Virtual test bus lines* from each virtual core are connected to the external TAM via de-multiplexing and multiplexing interfaces to synchronize the flow of the test data. The method employs a single separate shift clock for all virtual cores, and it is multiplexed with the capture clock signals. In [9], the authors of [8] improved upon their design by allowing each virtual core to have a distinct shift frequency. In both [8] and [9] all virtual cores are concurrently active and lowering shift frequencies which lead to large increases in test time might result under a very tight power constraint.

In [10, 11], the use of gated-clocks to control the start and end times of the shift activity of each virtual core has been proposed. [11] used a 3-D bin packing algorithm which grouped virtual cores into *shelves* wherein all cores belonging to the same shelf become active at the same time and each shelf becoming active sequentially.

This paper proposes a 1500 compatible power-aware multi-clock domain core wrapper which partitions the IP core into smaller sub-groups and utilizes gated-clocks to control the start times of scan-shift operations and enable a more flexible and efficient use of the external bandwidth under a power constraint. A heuristic 3-D rectangular bin packing algorithm is also introduced which forms the basis of the proposed wrapper design method.

The rest of the paper is organized as follows. The overview of the proposed wrapper architecture and its scan-control block is given in Section 2. Section 3 gives the problem formulation

and Section 4 discusses the proposed 3-D bin packing algorithm. Section 5 discusses the experimental results and compares them with the results of previous works and Section 6 concludes this paper.

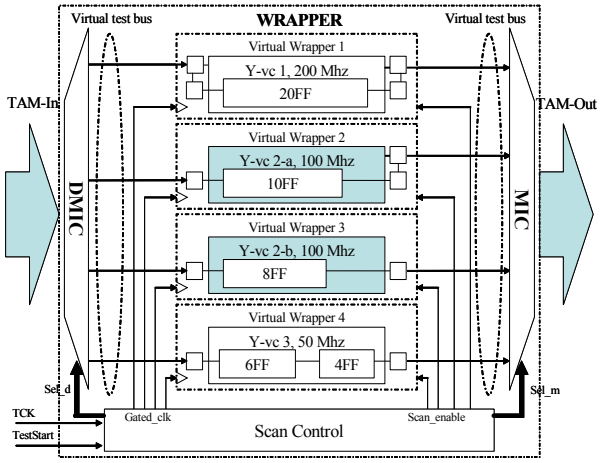

Figure 1. Proposed multi-clock domain core wrapper

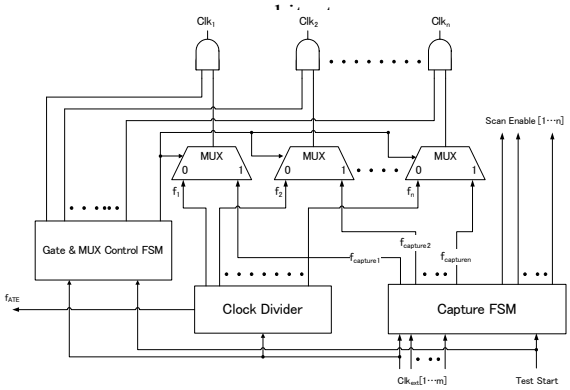

Figure 2. Proposed scan control circuit

2 Multi-clock Domain Core Wrapper (MDCW)

In this paper the partitioning of clock domains into smaller groups, which would be referred to as sub-domains, is allowed. Furthermore, it is supposed that the IP-core designer provides the following information:

P_{max}: Maximum allowed power dissipation (can be peak or average value)

N_C: No. of clock domains

N_{si}: No. of sub-domains for each clock domain D_i ($1 \leqq i \leqq N_c$)

For each sub-domain S_{ij} ($1 \leqq j \leqq N_{si}$) of clock domain D_i

- pi_{ij}: No. of primary input pins
- po_{ij}: No. of primary output pins
- bi_{ij}: No. of bidirectional I/O
- sc_{ij}: No. of internal scan chains and their lengths l_{ijk} ($1 \leqq k \leqq sc_{ij}$)
- p_{ij}: Power dissipation at ATE frequency f_{ATE}

We now extend the definition of the virtual core from [8]. For this paper, a group of one or more sub-domains from D_i is a *virtual core* G_{ip} ($1 \leqq p \leqq N_{gi}$). N_{gi} is the total number of possible combinations of the sub-domains of D_i. Furthermore, to differentiate from the virtual core defined in [8, 9] which we would refer to as X-vc, we would refer to virtual core in this paper as Y-vc.

The basic architecture of the proposed multi-clock domain core wrapper is shown in Figure 1. The core is divided into smaller Y-vc's, each having its own virtual wrapper. These Y-vc's are connected to the external TAM via a pair of de-multiplexing and multiplexing interface circuits (DMIC-MIC) which performs bandwidth matching and test data flow-control between the external TAM and the internal virtual test bus lines.

The scan control circuitry for the proposed wrapper is shown in Figure 2. A capture windows similar to that proposed in [8, 9] would be used during the capture phase. For this to work, we added a *gate and MUX control circuit* to control the clock signals during test as well as switch to the functional clocks during normal operation. The use of gated clocks enables a more flexible and efficient test schedule compared to [9]. Instead of the shelf method[11], our scheme allows partitioned testing and partitioning some of the domains can further reduce test time. A comparison of schedules obtained in [9] and by our method is shown in Figure 3.

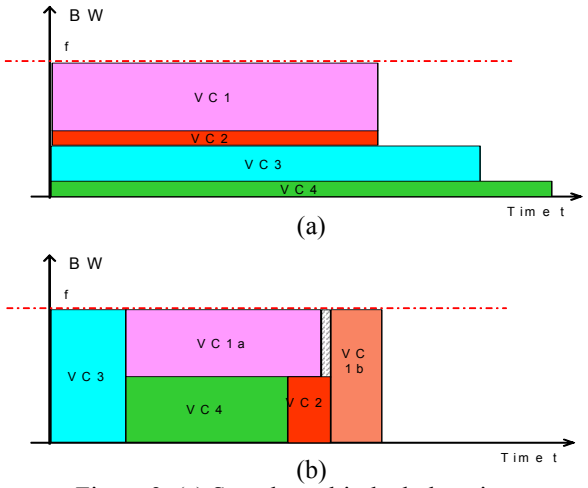

Figure 3. (a) Sample multi-clock domain core using [9] (b) using proposed method

3 Problem Formulation

In this section, we would formally present the wrapper design problem P_{MDCW}.

Problem P_{MDCW}: Given the test parameters for a multi-clock domain core C as described in section 2 and the information below:

W_{ext}: TAM width allotted to the core

f_{ATE}: ATE frequency

$F = \{f_1, f_2, ..., f_m \mid f_j = 2 \times f_{j+1}, j \in 1, ..., m-1\}$: The set of allowed shift frequencies

determine the multi-clock domain wrapper design for C including:

N_{vi}: No. of virtual cores G_{ip} ($1 \leqq p \leqq N_{vi}$) under domain D_i

For each Y-vc G_{ip} of clock domain D_i

- $f_{ip} \in F$: Shift frequency
- w_{ip}: No. of virtual bus lines
- Its wrapper scan chain design and the length of its longest scan chain l_{ipmax}
- t_{sip}: Scan-in start time

under the following constraints:

1) The bandwidth used by all the active Y-vc's at any time t cannot exceed the total bandwidth coming from the ATE:

$$W_{ext} \times f_{ATE} \geq \max_{0 \leq t \leq T} \left(\sum_{i=1}^{N_c} \sum_{p=1}^{N_{vi}} a_{ipt} \, f_{ip} \, w_{ip} \right) \quad (1)$$

$$where \quad : a_{ipt} = \begin{cases} 0, t < t_{sip} \,||\, t > (t_{sip} + l_{ip\,max} \times f_{ATE} / f_{ip}) \\ 1, t_{sip} \leq t \leq (t_{sip} + l_{ip\,max} \times f_{ATE} / f_{ip}) \end{cases}$$

2) The total power dissipation of all active Y-vc's at any time t cannot exceed the maximum allowed power dissipation P_{max}:

$$P_{max} \geq \max_{0 \leq t \leq T} \left(\sum_{i=1}^{N_c} \sum_{p=1}^{N_{vi}} a_{ipt} \, p_{ip} \times f_{ip} / f_{ATE} \right) \quad (2)$$

such that the test application time is minimized. Since all the virtual cores become active and inactive independently, the total scan-shift time for one test pattern can be computed from the Y-vc with the latest end time as shown below:

$$T = \max_{1 \leq i \leq N_c} \left(\max_{1 \leq p \leq N_{vi}} \left(t_{sip} + l_{ip\,max} \times f_{ATE} / f_{ip} \right) \right) \quad (3)$$

Each Y-vc will have a distinct shift frequency and to simplify the clock generation circuitry, the ratio of usable frequencies is a two's exponent. The 3-D bin packing algorithm that would be discussed in the next section would be used to minimize the scan-shift time T while also optimizing the number of virtual cores.

4 3-D Bin Packing Algorithm

Rectangular 2-D bin packing have been extensively used to solve the test scheduling problem for embedded cores [4, 12]. For scheduling under a power constraint, it has been extended into a restricted 3-D bin packing problem where the length, width and height represent pin, peak power and total test time, respectively, for an SoC. Specifically, when a cube overlaps in the time domain, they cannot overlap at any of the other two domains. For this paper, instead of pin count, the length would represent the external bandwidth $BW_{ex}=W_{ext} \times f_{ATE}$ and each virtual core G_{ip} of an IP-core C can be represented by one cube among a set of permissible cubes. The cubes are packed into the 3-D bin until all the sub-domains S_{ij} of all the domains D_i have been included in the packed virtual cores while minimizing the total height. Since it has been shown that the restricted 3-D bin packing problem is *NP-Hard* in [14], this paper proposes a heuristic algorithm to solve the problem.

Initialization: Cube Creation and Ordering

Virtual cores can be created to represent any combination of sub-domains belonging to the same clock domain. Thus, if a domain D_i has N_{si} sub-domains, then the total number of possible virtual cores for D_i is just the sum of all the possible combinations of S_{ij}.

$$N_{gi} = \sum_{j=1}^{N_{si}} {}_{N_{si}}C_j \quad (4)$$

Single frequency wrapper design such as in [3] is performed on all virtual cores. We denote the maximum number of virtual test bust lines that can be assigned to a Y-vc as Vtb_{max}. Each virtual core G_{ip} ($1 \leqq p \leqq N_{gi}$) can have a maximum of Vtb_{max} possible wrapper designs and the same number of cubes would be created. From the list of cubes, the cube with the shortest test time regardless of power or bandwidth constraints would be selected as its *ideal cube*. It was proven in [11] that halving the shift frequency of the virtual core while doubling the virtual test bus width can result in either an increase in scan-shift time or no increase at all. We use this property to maintain the test time while still minimizing the power dissipation of the core. Each Y-vc G_{ip} is expressed as a triplet $C_{ip} = \{BW_{ip}, p_{ip}, t_{ip}\}$, where $BW_{ip} = w_{ip} \times f_{ip}$ is the bandwidth of G_{ip}, w_{ip} is the virtual test bus width assigned to it, and $f_{ip} \in F$ is its shift frequency. The power dissipation p_{ip} at f_{ip} can be expressed as:

$$p_{ip} = \frac{p_{ip\,max} \times f_{ip}}{f_1} \quad (5)$$

where f_1 is the maximum allowed shift frequency. For this paper, we set $f_1 = f_{ATE}$ and p_{ipmax} is the power dissipation at f_1 as expressed below:

$$p_{ip\,max} = \sum_{j=1}^{N_{vi}} b_{ij} \, p_{ij} \quad (6)$$

where $b_{ij} = 1$ when sub-domain S_{ij} belongs to G_{ip} and $b_{ij} = 0$ if not. The minimum test time t_{ip} can be computed as follows:

$$t_{ip} = \frac{l_{ip\,max}}{f_{ip}} \quad (7)$$

The ideal cubes of virtual cores representing whole domains that satisfy the bandwidth and power restrictions is put into a list L_D. The remaining ideal cubes not in the list but satisfy the bandwidth and power constraints are added to L_G. The ideal cubes of virtual cores representing only one sub-domain is then added to L_G to ensure that all sub-domains would be scheduled. Finally, a master list L_M is created and all elements of L_D and L_G are added into it while deleting duplicates.

An area attribute $A_{ip} = BW_{ip} \times t_{ip}$ is also computed per virtual core of single sub-domains. The bigger the area attribute, the harder it is to pack. It follows that sub-domain groups which have big sub-domains must be prioritized by independently sorting all three lists from the virtual core which has the member sub-domain with the biggest area attribute to

the smallest one. If two virtual cores have the same sized sub-domains, then the overall area attribute of the whole virtual cores are compared during sorting. After the list preparations, bin packing can be started from Step 1.

Step 1: Packing of Domain Cubes

Before packing, the algorithm takes note of the current time in the schedule, denoted by a variable *curr_time*. Since having more virtual cores mean a bigger and more complex scan-control circuitry, the algorithm only divides domain virtual cores when necessary. In this step, the algorithm only looks at list L_D until it finds a cube that has $p_{ip} \leqq p_{avail}$ and $BW_{ip} \leqq BW_{avail}$. p_{avail} is the available power and BW_{avail} is the available bandwidth, respectively. If a cube is found and packed into the bin, the algorithm checks what sub-domains belong to it. All three lists are then updated by removing all cubes that has at least one member sub-domain equal to any of the sub-domains of the packed core. It continues the above process until (a) BW_{avail} and/or p_{avail} becomes zero or (b) if a proper cube cannot be found in L_D. Under condition (a), the algorithm looks among currently scheduled virtual cores for the Y-vc with the earliest test end time and sets *curr_time* equal to it. Then Step 1 is repeated. Under condition (b), the algorithm proceeds to Step 2. For the benchmark core hTCAD01[9] shown in Table 1 with BW_{ext} = 1600 and P_{max} = 3000, Step 1 packs the Y-vc of domain no. 5 as shown in Figure 4.

Step 2: Packing of Sub-domain Group Cubes

Not finding a cube in Step 1 means that partitioning a domain is necessary. In this step the algorithm only looks at list L_G until it finds a cube that has $p_{ip} \leqq p_{avail}$ and $BW_{ip} \leqq BW_{avail}$. If a cube is found and packed into the bin, the algorithm checks what sub-domains belong to it and updates all three lists by removing all cubes that has at least one member sub-domain equal to any of the sub-domains of the packed core. Step 2 is repeated while there still available power and/or bandwidth or if a proper cube cannot be found. If BW_{avail} and/or p_{avail} become zero, *curr_time* is again updated and the algorithm goes back to Step 1. But if a proper cube cannot be found in L_G, the algorithm proceeds to step 3. In Figure 4, Step 2 is illustrated when the Y-vc of sub-domain group S_{61} and S_{63} of domain no. 6 is packed into the bin.

Step 3: Filling Idle Space by Decreasing Virtual Test Bus Lines

In Step 3, the algorithm searches for the packed cube using the biggest bandwidth and denotes its test end time as T_{endmax}. The master list L_M is then traversed for a cube that satisfies the available power p_{avail}. The algorithm then computes for the new scan-shift time t_{ipnew} given a new bandwidth $BW_{ipnew} = BW_{avail}$. Because of the limited selection of usable shift frequencies, choosing the next lowest f_k would automatically lead to a doubling of the scan-shift time. So for step 3, the assigned virtual test bus width is decreased and t_{ipnew} is computed. The shift frequency is halved and the virtual test bus is doubled to minimize power as long as t_{ipnew} remains constant. If $t_{ipnew} \leqq T_{endmax} \times (1+dmax/100)$ then the cube is packed into the bin and all three lists are updated as before. *dmax* is a heuristic value which expresses how much t_{ipnew} can go over T_{endmax} The algorithm repeats Step 3 until there is no available bandwidth and/or power or no suitable cubes can be found in L_M. If BW_{avail} and/or p_{avail} becomes zero, *curr_time* is updated and the algorithm goes back to Step 1. If no suitable cubes were found, the algorithm proceeds to Step 4. For example, the ideal cube of domain no. 7 has f_{ip} = 100MHz, w_{ip} = 15, p_{ip} = 40 and t_{ip} = 0.10 μ sec. In Figure 4, the idle bandwidth was 300MHz and for Y-vc_7, w_{ip} was first reduced to 3 at f_{ip} = 100MHz. Consequently t_{ip} increased to 0.48 μ sec. While keeping t_{ip} constant, we are able to increase the w_{ip} to 12, while decreasing the shift frequency f_{ip} to 25MHz and the power p_{ip} was lowered to 10 before packing the cube into the bin.

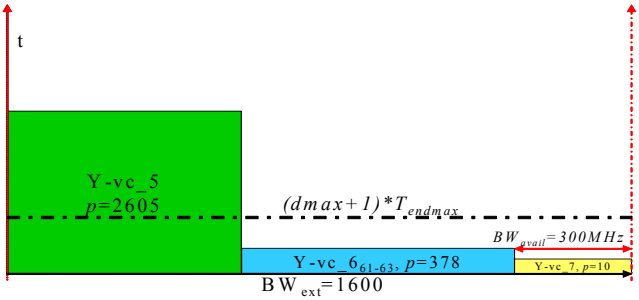

Figure 4. Packing results from steps 1 - 3

Step 4: Filling Idle Space by Decreasing Shift Frequency

Reaching Step 4 means that no cube satisfies the available power p_{avail}. There is no choice but to lower the shift frequency of a virtual core to fit the available idle space in the bin. The algorithm determines T_{endmax} and makes a copy of the list L_M called L_{tmp}. Then the shift frequencies of all cubes remaining in L_{tmp} are lowered until their power is less than or equal to p_{avail}.

For each cube, the new scan-shift time t_{ipnew} is computed given a new bandwidth $BW_{ipnew} = BW_{avail}$. The shift frequency is halved and the virtual test bus is doubled to minimize power as long as t_{ipnew} remains constant. Unlike Step 3, the algorithm looks for a cube that satisfies $t_{ipnew} \leqq T_{endmax} \times (1+emax/100)$ and closest to T_{endmax} as we have found during experimentation that this gives better results than simply packing the first cube that satisfies the first condition. The cube is packed into the bin and all three main lists are updated as before. Step 4 is repeated until no cubes were found or until BW_{avail} and/or p_{avail} becomes zero. The algorithm then updates *curr_time* and goes back to Step 1. Note that *emax* is a heuristic value independent of *dmax* which expresses how much t_{ipnew} can go over T_{endmax}. Steps 1 through 4 are repeated until all the lists are empty. For example, the ideal cube of sub-domain S_{33} has f_{ip} = 100MHz, w_{ip} = 3, p_{ip} = 186 and t_{ip} = 0.93 μ sec. In Figure 5, although there is a large BW_{avail}, p_{avail} is only 116 so the f_{ip} of Y-vc_3_{33} was decreased to 50MHz, and this led to a decrease in power

p_{ip} = 93 while w_{ip} remained the same. Consequently t_{ip} doubled to 1.86 μ sec. Figure 6 shows the finished schedule.

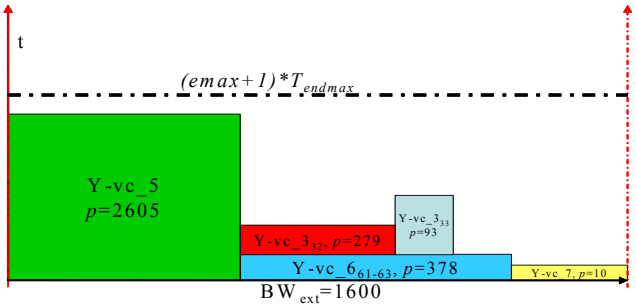

Figure 5. Cube of sub-domain 3_{33} packed in step 4

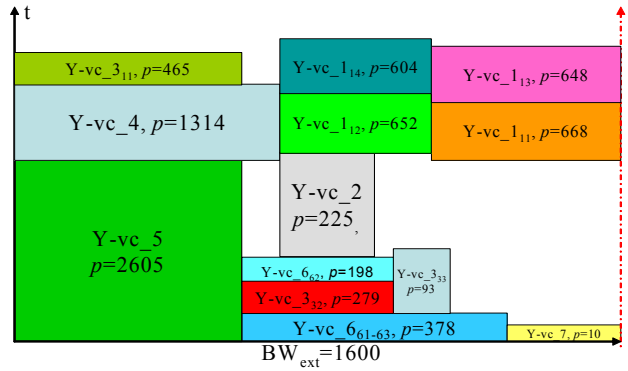

Figure 6. Finished test schedule for hTCAD01 at with BW_{ext} = 1600 and P_{max} = 3000

5 Experimental Results

The experiments were done using the benchmark multi-clock domain core hTCADT01 used in [8, 9, 10, 11]. This core has seven clock domains, and we assumed that each domain can be further partitioned into sub-domains as shown in Table 1. In the table, *sd#* denotes the sub-domain number, f_{func} is the functional frequency, N_{in}, N_{out} , N_{bi} and N_{sc} are the number of inputs, outputs, bidirectional I/O and scan chains in the specific sub-domain respectively, Lsc_{ij} is the length of each scan chain and P is the power dissipation of the sub-domains when shifting at 100MHz and is the sum of all the scan chain lengths Ls_{cij} belonging to that sub-domain.

The experiment was conducted under four different power constraints P_{max}: 1500, 3000, 4500 and infinity. The maximum allowed frequency f_1 is 100MHz which is equal to $f_{ATE.}$ to synchronize the internal shift frequencies with the ATE. To see the effectiveness of the proposed method, the resulting shift times denoted by T_{new} are compared to the results from [9], marked *T*[9], and from [11], marked *T*[11], in Tables 2 and 3. All times are in microseconds. %diff[9] and %diff[11] are the differences in percentage to [9] and [11] respectively. During the experiments, *dmax* and *emax* were independently varied from 0 to 200 to find the optimal combination. The experiments were done using a Sun Fire V490 1.35GHz UltraSPARC IV workstation with 32GB memory and the 40,000 looped reruns of the program didn't take more than 1 sec. of CPU time.

At the tightest power constraint P_{max} = 1500, our algorithm was able to decrease the shift time with a maximum of 24.42% compared to [9]. The average gain in test time as P_{max} is decreased, increases dramatically from 8.69% at P_{max} = *infinity* to 18.36% at P_{max} = 1500. This can be attributed to the fact that wider W_{ext} and lower P_{max} makes splitting the domains more effective because of the extra freedom it gives during scheduling. Compared to [11], the trend is different and there is an almost constant average gain of around 4% across all power constraints and a maximum gain of 14.25% at P_{max} = 3000. Small differences in time (0-1%) are attributed to discrepancies in rounding-off among the programs used and makes them negligible, so our algorithm matches or exceeds [11] in 90% of the cases. Special note must be given to the non-motonic trend of our results. For example, at P_{max} = 1500, W_{ext} = 9 yields a time of 15.91 μ s and W_{ext} = 10 yields a time of 16.68 μ s. This can be attributed to how the algorithm was designed to minimize the number of virtual cores while still optimizing the test time. Thus, there will be occasions when the algorithm would choose to pack a bigger cube instead of dividing it as long as bandwidth and power are available.

6 Conclusion

We have presented a novel method of designing a test wrapper for multi-clock domain cores by effectively utilizing clock domain partitioning and gated-clocks. With minimal hardware overhead for gated-clock control, we have dramatically improved upon earlier methods which concurrently activate all the domains during test, especially under tight power constraints. Furthermore, the division of clock domains enabled us to give better results than the previous methods which utilize gated-clocks.

Acknowledgements

This work was supported in part by Japan Society for the Promotion of Science (JSPS) under Grants-in-Aid for Scientific Research B(2)(No.15300018), JSPS under Grants-in-Aid for Young Scientists (B) (No. 18700046) and the grant of JSPS Research Fellowship (No. S06089).

References

[1] E. J. Marinissen, S. K. Goel and M. Lousberg, "Wrapper Design for Embedded Core Test," *Proc. of IEEE International Test Conference (ITC)*, pp. 911–920, 2000.

[2] S. K. Goel and E. J. Marinissen, "Effective and Efficient Test Architecture Design for SoCs," *Proc. of IEEE International Test Conference (ITC)*, pp. 529–538, 2002.

[3] V. Iyengar, K. Chakrabarty and E. J. Marinissen, "Test Wrapper and Test Access Mechanism Co- Optimization for System-on-Chip," *Journal of Electronic Testing: Theory and Application*, vol. 18, pp. 213–230, April 2002.

[4] V. Iyengar, K. Chakrabarty and E. J. Marinissen, "Test Access Mechanism Optimization, Test Scheduling, and Tester Data Volume Reduction for System-on-Chip," *IEEE Trans. On Computers*, vol. 52, no. 12, pp. 1619–1632, December 2003.

[5] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan and J. Rajski, "Logic BIST for Large Industrial Designs: Real Issues and Case Studies," *Proc. of IEEE International Test Conference (ITC)*, pp. 358–367, 1999.

[6] K. Hatayama, M. Nakao and Y. Sato, "At- Speed Built-in Test for Logic Circuits with Multiple Clocks," *Proc. of the 11th Asian Test Symposium (ATS)*, pp. 292–297, 2002.

[7] V. Jain and J. Waicukauski, "Scan Test Volume Reduction in Multi-Clocked Designs with Safe Capture Technique," *Proc. of IEEE International Test Conference (ITC)*, pp. 148–153, 2002.

[8] Q. Xu and N. Nicolici, "Wrapper Design for Testing IP Cores with Multiple Clock Domains," *Proc. of Design, Automation, and Test in Europe (DATE)*, pp. 416–421, 2004.

[9] Q. Xu, N. Nicolici and K. Chakrabarty, "Multi-Frequency Wrapper Design and Optimization for Embedded Cores Under Average Power Constraints," *Proc. of ACM/IEEE Design Automation Conference (DAC)*, pp. 123–128, 2005.

[10] D. Zhao, U. Chandran and H. Fujiwara, "Design and Optimization of A Time-Gated Multi-Frequency Wrapper Architecture for Embedded IP Cores," *University of Louisiana at Lafayette Technical Report*, TR-2006-8-001, February 2006.

[11] D. Zhao, and U. Chandran, "Design of A Time-Gated Multi-Frequency Wrapper Architecture for Modular SoC Testing," *Proc. of IEEE 15th North Atlantic Test Workshop (NATW)*, May 2006.

[12] T. Yoneda, K. Masuda and H. Fujiwara, "Power- Constrained Test Scheduling for Multi-Clock Domain SoCs," *Proc. of Design, Automation, and Test in Europe (DATE)*, pp. 297–302, 2006.

[13] 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. Of the International Conference on Computer-Aided Design (ICCAD)*, pp. 100–105, 2003.

[14] Y. Huang et al., "Optimal Core Wrapper Width Selection and SOC Test Scheduling Based on 3-D Bin Packing Algorithm," *Proc. of IEEE International Test Conference (ITC)*, pp. 74–82, 2002.

Table 1. hCADT01 clock and sub-domain information

sd#	f_{func} *(MHz)*	N_{in}	N_{out}	N_{bi}	P	N_{sc}	Lsc_{ij}				
1.1	200	50	8	18	668	4	168	168	166	166	
1.2	200	25	8	18	652	4	163	163	163	163	
1.3	200	25	8	18	648	4	162	162	162	162	
1.4	200	9	8	18	604	4	151	151	151	151	
2.1	133	144	67	72	450	3	150	150	150		
3.1	120	39	4	24	465	5	93	93	93	93	93
3.2	120	30	3	24	279	3	93	93	93		
3.3	120	20	1	24	186	2	93	93			
4.1	75	61	10	36	657	3	219	219	219		
4.2	75	50	21	36	657	3	219	219	219		
5.1	50	87	200	36	1563	3	521	521	521		
5.2	50	30	24	36	1042	2	521	521			
6.1	33	96	10	24	278	5	82	81	81	17	17
6.2	33	30	18	24	198	4	82	81	18	17	
6.3	33	20	40	24	100	2	82	18			
7.1	25	15	30	72	40	4	10	10	10	10	

Table 2. Comparison of Scan-shift Time for hCADT01 under P_{max} = 1500

	P_{max}=1500				
W_{ext}	T[9]	T[11]	T_{new}	*%diff[9]*	*%diff[11]*
16	20.84	16.9	15.75	24.42	6.80
15	20.84	16.9	15.75	24.42	6.80
14	20.84	16.9	15.75	24.42	6.80
13	20.84	16.9	15.75	24.42	6.80
12	20.84	16.9	15.85	23.94	6.21
11	20.84	16.97	15.85	23.94	6.60
10	20.84	17.07	16.68	19.96	2.28
9	20.84	17.47	15.91	23.66	8.93
8	20.84	17.57	16.9	18.91	3.81
7	20.84	19.29	18.92	9.21	1.92
6	20.84	19.6	19.63	5.81	-0.15
5	25.04	23.38	23.19	7.39	0.81
4	29.76	26.16	25.34	14.85	3.13
3	41.68	34.06	34.08	18.23	-0.06
2	59.88	51.61	50.81	15.15	1.55
1	116.04	100.7	98.69	14.95	2.00

Table 3. Comparison of Scan-shift Time for hCADT01 under P_{max} = 3000, 4500 and infinity

	P_{max}=3000					P_{max}=4500					P_{max}=*infinity*				
W_{ext}	T[9]	T[11]	T_{new}	*%diff[9]*	*%diff[11]*	T[9]	T[11]	T_{new}	*%diff[9]*	*%diff[11]*	T[9]	T[11]	T_{new}	*%diff[9]*	*%diff[11]*
16	10.42	9.13	8.89	14.68	2.63	7.44	6.94	6.88	7.53	0.86	7.44	7.53	6.88	7.53	8.63
15	10.42	10.42	9.02	13.44	13.44	8.76	8.33	7.53	14.04	9.60	7.49	8.33	7.53	-0.53	9.60
14	10.42	10.42	9.02	13.44	13.44	8.88	8.79	7.81	12.05	11.15	8.88	8.79	7.81	12.05	11.15
13	10.42	10.53	9.03	13.34	14.25	10.42	8.93	8.53	18.14	4.48	9.59	8.79	8.53	11.05	2.96
12	10.42	10.53	10.32	0.96	1.99	10.42	9.75	9.44	9.40	3.18	10.42	9.15	9.44	9.40	-3.17
11	11.62	11.12	10.52	9.47	5.40	10.42	9.75	10.18	2.30	-4.41	10.42	9.75	10.18	2.30	-4.41
10	12.08	11.24	10.88	9.93	3.20	11.62	10.58	11.2	3.61	-5.86	11.62	10.58	11.2	3.61	-5.86
9	13.00	12.56	12.14	6.62	3.34	12.78	11.85	11.85	7.28	0.00	12.78	11.83	11.85	7.28	-0.17
8	14.48	13.50	13.18	8.98	2.37	14.88	13.50	13.36	10.22	1.04	14.88	13.5	13.36	10.22	1.04
7	17.76	16.57	15.14	14.75	8.63	15.63	15.43	15.09	3.45	2.20	15.63	15.43	15.09	3.45	2.20
6	20.84	17.19	17.19	17.51	0.00	19.20	17.19	17.19	10.47	0.00	19.18	17.19	17.19	10.38	0.00
5	23.24	21.81	20.64	11.19	5.36	23.24	21.81	20.64	11.19	5.36	23.24	21.81	20.64	11.19	5.36
4	29.76	26.15	25.52	14.25	2.41	29.01	26.15	25.52	12.03	2.41	29.01	24.84	25.52	12.03	-2.74
3	38.36	34.06	34.08	11.16	-0.06	38.36	34.06	34.08	11.16	-0.06	38.36	34.06	34.08	11.16	-0.06
2	58.02	50.36	50.48	13.00	-0.24	58.02	50.36	50.48	13.00	-0.24	58.02	50.36	50.48	13.00	-0.24
1	116.04	98.44	98.63	15.00	-0.19	116.04	98.44	98.63	15.00	-0.19	116.04	98.44	98.63	15.00	-0.19