Timing Verification for Adaptive Integrated Circuits

Rohit Kumar†, Bing Li‡, Yiren Shen†, Ulf Schlichtmann‡ and Jiang Hu†
†Department of Electrical and Computer Engineering, Texas A&M University
‡ Institute for Electronic Design Automation, Technische Universität München
rohit0513@gmail.com, b.li@tum.de, season_tamu@tamu.edu, ulf.schlichtmann@tum.de, jianghu@tamu.edu

***Abstract*—An adaptive circuit can perform built-in self-detection of timing variations and accordingly adjust itself to avoid timing violations. Compared with conventional over-design approach, adaptive circuit design is conceptually advantageous in terms of power-efficiency. Although the advantage has been witnessed in numerous previous works including test chips, adaptive design is far from being widely used in practice. A key reason is the lack of corresponding timing verification support. We develop new timing analysis techniques to fill this void. A main challenge is the large runtime complexity due to numerous adaptivity configurations. We propose several pruning and reduction techniques and apply them in conjunction with statistical static timing analysis (SSTA). The proposed method is validated on benchmark circuits including the recent ISPD'13 suite, which has circuit as large as $150K$ gates. The results show that our method can achieve orders of magnitude speedup over Monte Carlo simulation with about the same accuracy. It is also several times faster than an exhaustive application of SSTA.**

I. Introduction

An adaptive circuit uses on-chip sensors to detect timing variations and autonomously compensates the variations by body biasing [1], supply voltage tuning [2], circuit reconfiguration [3], etc. Unlike the conventional over-design approach, which applies extra power uniformly (or blindly) across entire circuit (or all fabricated chips) to cushion variations, adaptive circuits apply power differentially at each block according to individually observed performance, i.e., in a targeted manner. Evidently, adaptive circuit design provides variation resilience in a more power-efficient manner than over-design. This is especially true when the adaptivity is fine-grained (in blocks of hundreds/thousands of gates) and therefore has relatively precise compensation [2], [3].

Even with benefits demonstrated by test chips [1], adaptive circuit design is far from being widely adopted in realistic products. A main reason is the lack of corresponding timing verification support. The unconventional and sophisticated nature of adaptive design implies a relatively large risk of design errors. Obviously, correct circuit functioning is more essential than potential power savings. As such, an adaptivity-aware timing analysis tool is a fundamental premise for wide application of adaptive circuits designs. One may consider to use conventional timing analysis on adaptive circuit with some simple tweaks. Such an approach is either very restrictive or inefficient. For example, conventional timing analysis based on Monte Carlo simulation is directly applicable but very time consuming. Another naïve approach is to apply conventional timing analysis to each adaptivity mode. This approach is practical only if the adaptivity is coarse-grained and the number of adaptivity blocks is small.

In this work, we attempt to find a general and practical approach to timing verification for adaptive circuit designs. Since adaptivity is operated according to observed variations, variations need to be taken into consideration in the timing analysis. Fortunately, statistical static timing analysis (SSTA) has been an active research subject [4], [5] and these research works produce a rich body of techniques. Our study starts with an SSTA-based adaptivity enumeration approach. Then, we explore several pruning and reduction techniques in order to decrease computation cost. To our best knowledge, this is the first systematic effort on timing analysis for adaptive circuit design. The proposed techniques are tested on ISCAS'85 and ISPD'13 benchmark circuits, which include cases as large as $150K$ gates. Our technique is orders of magnitude faster than Monte Carlo simulation and provides very similar accuracy. Compared to SSTA-based exhaustive enumeration, our method yields almost identical result with several times less runtime.

II. Problem Formulation

Given a combinational logic circuit design $\mathcal{C}$ composed by a set of adaptivity blocks $\{b_1, b_2, ..., b_n\}$, timing constraints $\mathcal{T}$ and certain delay models, a fundamental objective of timing verification is to find whether or not $\mathcal{C}$ satisfies $\mathcal{T}$. When variations are considered, the objective is often changed to find the probability that $\mathcal{C}$ satisfies $\mathcal{T}$, i.e., timing yield [5].

If $\mathcal{C}$ is an adaptive circuit, it includes variation sensors [6], [7], circuit tuning mechanisms [1], [2], and an adaptivity policy. Without loss of generality, we assume the adaptive tuning is offline, i.e., it is performed at power-on or circuit idle time between normal operations. The input to an adaptivity policy is an integer vector $\vec{x} = (x_1, x_2, ..., x_m)^T$ resulting from m variation sensors. For example, $x_i = 1$ means that sensor i detects a high risk of timing violation. The set of all possible sensor observation vectors is denoted by X. If there are n adaptivity blocks in $\mathcal{C}$, the output of adaptivity control is a vector $\vec{f} = (f_1, f_2, ..., f_n)^T$ where f_i specifies adaptivity configuration for block b_i. The value of f_i is an element of a set of adaptivity configurations $\{\phi_0, \phi_1, ..., \phi_q\}$, where ϕ_0 means no adaptivity action. For example, $f_j = \phi_0$ ($f_j = \phi_1$) chooses low (high) VDD for block b_j. All elements in the same adaptivity block follow the same adaptivity configuration. The set of all possible adaptivity configurations is represented by F. Then the adaptivity policy can be described by a function $\Pi : X \rightarrow F$. Timing verification for adaptive circuit is to find the probability that $\mathcal{C}$ satisfies $\mathcal{T}$ for a given adaptivity policy Π. In practice, an adaptivity policy is typically simple rule-based [1]–[3] in order to avoid large implementation overhead. Our techniques can handle different adaptivity policies by querying them as black-box routines.

III. Basic Approaches

When variations are considered, each component delay becomes a random variable. In this work, we assume the random variables follow Gaussian distribution, which is a

978-3-9815370-4-8/DATE15/©2015 EDAA

reasonable approximation [5]. We consider spatial correlations among variations using Principal Component Analysis (PCA) as in [4]. A naïve approach is Monte Carlo simulation, where each run emulates one chip instance including its adaptivity actions and timing analysis is performed for this instance. To obtain high statistical confidence level, the number of runs is typically very large. Monte Carlo simulation will be used as a baseline for accuracy and runtime comparison in this work.

Now we discuss an adaptivity scenario enumeration approach. In this approach, we perform SSTA on a circuit for each adaptivity scenario individually, and then assemble the results into the overall timing yield. If there are m variation sensors and each sensor has up to p levels of output, there could be at most $|X| = p^m$ variation observation scenarios. If there are n adaptivity blocks and each block has up to q configuration options, there are at most $|F| = q^n$ configuration scenarios. Then, there are $\min(|X|, |F|)$ adaptivity scenarios.

SSTA is first conducted for the circuit without any adaptivity actions. Then, a probability density function (PDF) of timing slack can be obtained for each node of the circuit, including all sensor nodes. From these PDFs, one can estimate the probability of each sensor observation $P(\vec{x}_k)$, which is also the probability of an adaptivity scenario.

By running SSTA on each scenario, we can obtain the yield $Y(\vec{f}_k)$ for each configuration corresponding to observation $\vec{x}_k$. Then, the overall circuit timing yield can be estimated by

$$Y(\mathcal{C}) = \sum_{k=1}^{\min(|X|,|F|)} P(\vec{x}_k) \cdot Y(\vec{f}_k) \quad (1)$$

IV. Pruning and Reduction Techniques

A. Adaptivity Configuration Pruning

Suppose circuit $\mathcal{C}$ has n adaptivity blocks such that all gates in the same block follow the same adaptivity actions. We use $f(b_i)_j = \phi_j$ to represent adaptivity configuration j for block b_i. We define that $f(b_i)_j$ dominates $f(b_i)_k$ when $f(b_i)_j$ is a more powerful adaptive tuning than $f(b_i)_k$. For example, $f(b_i)_j$ uses higher supply voltage than $f(b_i)_k$, or $f(b_i)_j$ applies forward body bias (FBB) while $f(b_i)_k$ employs reverse body bias (RBB). We denote the dominance by $f(b_i)_k \prec f(b_i)_j$. If the case where $f(b_i)_j$ and $f(b_i)_k$ are identical is included, the notation becomes $f(b_i)_k \preceq f(b_i)_j$. Similarly, an adaptivity vector $\vec{f}_j$ dominates another one $\vec{f}_k$ if $f(b_i)_k \preceq f(b_i)_j, i = 1, 2, ..., n$. We use similar notation $\vec{f}_k \preceq \vec{f}_j$ for this definition.

Definition 1 – Robust adaptivity configuration: *An adaptivity configuration $\vec{f}_j$ for circuit $\mathcal{C}$ is robust if timing yield satisfies $\sup Y(\mathcal{C}, \vec{f}_j) = 1$ under this configuration.*

Pruning Rule 1: *If an adaptivity configuration $\vec{f}_j$ is robust, then any other configuration $\vec{f}_k$ satisfying $\vec{f}_j \preceq \vec{f}_k$ is also robust and does not need to be evaluated.*

Definition 2 – Failing adaptivity configuration: *An adaptivity configuration $\vec{f}_j$ for circuit $\mathcal{C}$ is failing if timing yield satisfies $\inf Y(\mathcal{C}, \vec{f}_j) = 0$ under this configuration.*

Pruning Rule 2: *If an adaptivity configuration $\vec{f}_j$ is failing, then any other configuration $\vec{f}_k$ satisfying $\vec{f}_k \preceq \vec{f}_j$ is also failing and does not need to be evaluated.*

B. Circuit Reduction

Circuits can be reduced from the timing point of view. For example some serial/parallel timing arcs can be merged [8]. Also, the timing arcs that are never critical even under variations can be neglected.

C. Circuit Partitioning

The timing analysis complexity can be reduced if adaptivity blocks are independent of each other. If a circuit is composed by n parallel and almost independent adaptivity blocks, the adaptivity configuration of a block does not need to be considered in conjunction with other blocks. If a block has q configurations, we only need to examine $q \cdot n$ scenarios. By contrast, a full enumeration for a general case needs to check q^n scenarios.

D. Block Merging

Some blocks can be merged into a virtual block in the adaptivity enumeration without affecting accuracy. This is based on the following concepts and observations.

Definition 3 – Critical fanin cone: *For a primary output O, its critical fanin cone $\Gamma(O)$ is the set of blocks whose timing may affect the probability of satisfying timing constraint at O.*

Definition 4 – Mutually don't care: *For two primary outputs O_i and O_j, if block $b_k \in \Gamma(O_i), \notin \Gamma(O_j)$ and block $b_l \in \Gamma(O_j), \notin \Gamma(O_i)$, then b_k and b_l are mutually don't care blocks.*

Merging Rule: *If two blocks are mutually don't care, they can be merged to a single virtual block during the adaptivity enumeration without affecting analysis results.*

V. Overall Timing Analysis Algorithm

Our timing analysis algorithm starts from the exhaustive adaptivity enumeration described in Section III and is enhanced by the pruning and reduction techniques introduced in Section IV. The pseudo code of our timing analysis is shown in Algorithm 1. At the very beginning, we set the circuit to no adaptivity ϕ_0 and perform SSTA. Then, we can estimate the probability of each adaptivity configuration. In addition, the SSTA result can help the circuit reduction (step 3). Next, we partition the circuit according to Section IV-C. Steps 5-29 describe how to compute timing yield for each partition.

At the beginning for each partition, we first merge blocks according to Section IV-D. A set R_i is to keep track of all robust adaptivity configurations for this partition (step 7). Starting from all blocks with ϕ_0, we enumerate different adaptivity configurations with increasing number of changing blocks (step 8, 9 and 10). For example, if there are 4 blocks, we first examine configurations with only 1 block having adaptivity beyond ϕ_0 while all the other blocks remain at ϕ_0. There are $C_1^4 = 4$ such configurations. Next, we examine combination C_2^4 blocks with adaptivity beyond ϕ_0, and so on. For each combination of changing blocks, we enumerate all adaptivity configurations in a non-descending order of dominance. In steps 13 and 14, we apply a change while keeping the other blocks at ϕ_0. If the new configuration dominates anyone in R_i or its probability is less than a threshold δ, we simply skip it (step 16 and 17). Otherwise, SSTA is performed for

this configuration. If this configuration is robust, it is added to R_i (step 21 and 22). At step 28, timing yield of a partition is obtained according to the yield of each configuration and probability of each configuration. Step 30 returns the final timing yield of the entire circuit.

Input : Circuit $\mathcal{C}$ in n blocks $B = \{b_1, b_2, ..., b_n\}$
Adaptivity configuration options $\{\phi_0, \phi_1, ..., \phi_q\}$
Adaptivity policy Π.
Output: Timing yield $Y(\mathcal{C})$

1 $f_i \leftarrow \phi_0, \forall b_i \in B$;
2 $SSTA(\mathcal{C})$; // Statistical static timing analysis
3 $\mathcal{C}^* \leftarrow Reduction(\mathcal{C})$; // Section IV-B
4 $\Psi = \{\psi_1, \psi_2, ...\} \leftarrow Partition(\mathcal{C}^*)$; //Section IV-C
5 **for** *each* $\psi_i \in \Psi$ **do**
6 $\quad \psi_i^* \leftarrow Merge\_blocks(\psi_i)$; //Section IV-D
7 $\quad R_i \leftarrow \emptyset$; // Set of robust adaptive configurations
8 $\quad$ **for** $j = 1$ *to* $|\psi_i^*|$ **do**
9 $\quad\quad S \leftarrow$ set of $C_j^{|\psi_i^*|}$ combinations of blocks;
10 $\quad\quad$ **for** *each combination* $c \in S$ **do**
11 $\quad\quad\quad F(c) \leftarrow$ all configurations of c in non-descending order of dominance;
12 $\quad\quad\quad$ **for** *for each* $\vec{f} \in F(c)$ **do**
13 $\quad\quad\quad\quad$ Apply $\vec{f}$ to c;
14 $\quad\quad\quad\quad$ Apply ϕ_0 to ψ_i^*/c;
15 $\quad\quad\quad\quad f(\psi_i^*) \leftarrow$ current configuration of ψ_i^*;
16 $\quad\quad\quad\quad$ **if** $(f_r \preceq f(\psi_i^*), f_r \in R_i)$ **or** $(P(f(\psi_i)) < \delta)$ **then**
17 $\quad\quad\quad\quad\quad$ Continue;
18 $\quad\quad\quad\quad$ **end**
19 $\quad\quad\quad\quad$ **else**
20 $\quad\quad\quad\quad\quad$ SSTA(ψ_i^*);
21 $\quad\quad\quad\quad\quad$ **if** $\sup Y(\psi_i^*, f(\psi_i^*)) == 1$ **then**
22 $\quad\quad\quad\quad\quad\quad R_i \leftarrow R_i \cup \{f(\psi_i^*)\}$;
23 $\quad\quad\quad\quad\quad$ **end**
24 $\quad\quad\quad\quad$ **end**
25 $\quad\quad\quad$ **end**
26 $\quad\quad$ **end**
27 $\quad$ **end**
28 $\quad Y(\psi_i) \leftarrow \sum_{\forall f(\psi_i^*)} P(f(\psi_i^*)) \cdot Y(\psi_i^*, f(\psi_i^*))$;
29 **end**
30 Return $Y(\mathcal{C}) \leftarrow \prod_{\forall \psi_i \in \Psi} Y(\psi_i)$;

Algorithm 1: Timing analysis for adaptive circuit design.

VI. Experiment

We evaluate the effectiveness of our approach by experimental comparisons on public benchmark circuits. Since there is no previous work on timing analysis for adaptive circuits, to the best of our knowledge, we compare the following approaches.

- Monte Carlo simulation. Spatial correlation among variations is considered. We use Monte Carlo simulation as a baseline for evaluating the accuracy and runtime of our approach.
- Exhaustive SSTA. SSTA is performed for every adaptivity configuration (Section III).
- Ours. Adaptivity configurations are enumerated with pruning/reduction, and evaluated by SSTA (Algorithm 1).

The experiments are performed on ISCAS'85 and ISPD'13 [9] benchmark circuits, with the largest case having about $150K$ gates. We use the Elmore delay model and gate RC values are obtained from the ISPD'13 benchmarks. These circuits are placed by Feng Shui placer [10], where spatial correlation model is extracted. We consider process variations including gate length variation, whose standard deviation σ is 5% of nominal value, and gate width variation with σ being 2.7% of nominal value. Monte Carlo simulations are performed 10K and 20K iterations for small and large circuits, respectively. The probability threshold δ for neglecting an adaptivity configuration is 0.0001. All of the methods are implemented in C++ and the program runs on a Linux server with 4 AMD Opteron processors of $2.2GHz$ operating frequency and $256GB$ memory.

The experimental results from ISCAS'85 circuits are shown in Table I and II, with tight and loose timing constraints, respectively. In both cases, the error from our method is about 0.9% compared with Monte Carlo results. Speedups of $32\times$ and $64\times$ are achieved for tight and loose timing constraints, respectively. Greater speedup is obtained for cases with loose constraints as they have less timing critical paths and allow more pruning and reduction. The speedup is not uniform among different circuits as the pruning/reduction highly depends on circuit structure. Comparing with the exhaustive SSTA, our approach is several times faster and the result difference is one order of magnitude less than the error with respect to Monte Carlo method. This is because our method is derived from the exhaustive approach.

Results from ISPD'13 circuits are in Table III and IV. Again, the two tables are for different timing constraints. The speedup compared to Monte Carlo is hundreds and sometimes thousands. The error is greater, but still less than 4% most of time. Although the exhaustive SSTA approach is usually faster than Monte Carlo, its runtime scales poorly with circuit size and it cannot finish in three hours for a large case.

VII. Conclusions

Adaptive circuit design is a promising approach to handling variations with high power-efficiency. This work proposes the first systematic timing verification method to assist adaptive design, to the best of our knowledge. Since an adaptive circuit may have many configurations, analyzing all cases can be very time consuming. We propose a set of pruning and reduction techniques. These techniques are validated on benchmark circuits of up to $150K$ gates. The results show that our method provides order of magnitudes speedup compared to Monte Carlo with very small errors.

Acknowledgement

The authors thank Professor Sachin Sapatnekar at University of Minnesota for helpful discussions. This work is partially supported by NSF (CCF-1255193), SRC (2013-TJ-2421) and Alexander von Humboldt Foundation.

References

[1] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. A. Antoniadis, A. P. Chandrakasan, and V. De. Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. *IEEE Journal of Solid-State Circuits*, 37(11):1396–1402, November 2002.

TABLE I. EXPERIMENTAL RESULTS FROM ISCAS'85 CIRCUITS WITH TIGHT TIMING CONSTRAINTS.

Circuit	#gates	Monte Carlo (MC)		Exhaustive SSTA		Ours		With respect to MC		W.r.t. Exhaustive	
		Yield	CPU (s)	Yield	CPU (s)	Yield	CPU (s)	% Error	Speedup	% Error	Speedup
c432	171	0.566	1.45	0.560	0.02	0.560	0.02	1.09	72.5	-0.01	1
c499	218	0.586	1.86	0.590	0.18	0.590	0.14	0.68	13.3	-0.01	1.46
c880	383	0.573	3.54	0.567	0.20	0.576	0.04	0.523	88.5	-0.09	5
c1355	562	0.516	4.60	0.538	4.26	0.538	3.80	4.26	1.21	-0.04	1.12
c1908	972	0.580	6.98	0.587	10.54	0.581	3.33	0.172	2.1	-0.03	3.17
c2670	1287	0.563	10.56	0.568	1.16	0.568	0.51	0.88	20.71	-0.17	2.27
c3540	1705	0.644	13.73	0.646	1.63	0.646	0.50	0.31	27.46	0.04	3.26
c5315	2351	0.597	21.30	0.598	2.87	0.598	0.32	0.16	66.56	0	8.97
c6288	2416	0.512	41.79	0.516	15.01	0.516	14.28	0.781	2.93	-0.01	1.05
							Average	0.91	32×	0.08	3×

TABLE II. EXPERIMENTAL RESULTS FROM ISCAS'85 CIRCUITS WITH LOOSE TIMING CONSTRAINTS.

Circuit	#gates	Monte Carlo (MC)		Exhaustive SSTA		Ours		With respect to MC		W.r.t. Exhaustive	
		Yield	CPU (s)	Yield	CPU (s)	Yield	CPU (s)	% Error	Speedup	% Error	Speedup
c432	171	0.946	1.40	0.943	0.02	0.943	0.01	-0.31	140	-0.09	2
c499	218	0.939	1.83	0.942	0.19	0.941	0.13	0.21	14	0.12	1.5
c880	383	0.960	3.55	0.953	0.20	0.953	0.03	-0.73	118	-0.02	6.7
c1355	562	0.958	4.56	0.952	4.34	0.951	2.97	-0.73	1.5	0.01	1.5
c1908	972	0.929	6.87	0.924	10.68	0.924	2.51	-0.53	2.7	0.02	4.3
c2670	1287	0.918	10.67	0.918	1.17	0.917	0.27	-0.10	40	0.03	4.3
c3540	1705	0.961	13.78	0.958	1.65	0.958	0.09	-0.31	153	0.05	18
c5315	2351	0.917	20.75	0.912	2.89	0.912	0.21	-0.54	99	-0.01	14
c6288	2416	0.942	41.27	0.938	7.40	0.937	5.81	-0.53	7.1	0.03	1.3
							Average	-0.396	64×	0.01	6×

TABLE III. EXPERIMENTAL RESULTS FROM ISPD'13 CIRCUITS WITH TIGHT TIMING CONSTRAINTS.

Circuit	#gates	Monte Carlo (MC)		Exhaustive SSTA		Ours		With respect to MC		W.r.t. Exhaustive	
		Yield	CPU (s)	Yield	CPU (s)	Yield	CPU (s)	% Error	Speedup	% Error	Speedup
usb_phy	609	0.515	1.66	0.518	0.08	0.519	0.01	0.77	166	0.20	8.0
fft	32281	0.527	469	0.536	19.05	0.538	1.39	2.08	338	0.37	13.79
cordic	41601	0.582	476	0.571	54.05	0.573	1.47	-1.54	324	0.35	36.76
des_perf	112644	0.519	3146	_	> 3Hrs	0.540	41.38	4.04	76.02	_	_
matrix_mult	155325	0.576	4758	0.589	1938	0.591	280	2.60	16.98	0.34	6.92
edit_dist	130661	0.515	3480	0.531	1243	0.531	2.51	3.1	1386	0	495
							Average	1.84	385×	0.25	110×

TABLE IV. EXPERIMENTAL RESULTS FROM ISPD'13 CIRCUITS WITH LOOSE TIMING CONSTRAINTS.

Circuit	#gates	Monte Carlo (MC)		Exhaustive SSTA		Ours		With respect to MC		W.r.t. Exhaustive	
		Yield	CPU (s)	Yield	CPU (s)	Yield	CPU (s)	% Error	Speedup	% Error	Speedup
usb_phy	609	0.926	1.62	0.927	1.0	0.927	0.01	0.10	162	0	100
fft	32281	0.905	418.4	0.917	9.3	0.919	0.12	1.54	3486	0.22	77.5
cordic	41601	0.939	465.6	0.924	52.26	0.924	0.809	-1.59	575	0	64.6
des_perf	112644	0.900	2807	_	> 3Hrs	0.924	24.78	2.66	113	_	_
matrix_mult	155325	0.937	4053	0.961	981.2	0.963	8.39	2.77	483	0.20	120
edit_dist	130661	0.902	3572	0.937	1220	0.938	0.62	3.99	5761	0.10	1968
							Average	1.89	1763×	0.104	466×

[2] X. Liang, G.-Y. Wei, and D. Brooks. Revival: a variation-tolerant architecture using voltage interpolation and variable latency. *IEEE Micro*, 29(1):127–138, January 2009.

[3] K.-N. Shim and J. Hu. Boostable repeater design for variation resilience in VLSI interconnects. *IEEE Transactions on VLSI Systems*, 21(9):1619–1631, September 2013.

[4] H. Chang and S. S. Sapatnekar. Statistical timing analysis under spatial correlations. *IEEE Transactions on Computer-Aided Design*, 24(9):1467–1482, September 2005.

[5] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer. Statistical timing analysis: from basic principles to state of the art. *IEEE Transactions on Computer-Aided Design*, 27(4):589–607, April 2008.

[6] T. Sato and Y. Kunitake. A simple flip-flop circuit for typical-case designs for DFM. In *Proceedings of the IEEE International Symposium on Quality Electronic Design*, pages 539–544, 2007.

[7] A. Drake, R. Senger, H. Deogun, G. Carpenter, S. Ghiasi, T. Nguyen, N. James, M. Floyd, and V. Pokala. A distributed critical-path timing monitor for a 65nm high-performance microprocessor. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 398–399, 2007.

[8] C. W. Moon, H. Kriplani, and K. P. Belkhale. Timing model extraction of hierarchical blocks by graph reduction. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 152–157, 2002.

[9] M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke, and C. Zhuo. An improved benchmark suite for the ISPD-2013 discrete cell sizing contest. In *Proceedings of the ACM International Symposium on Physical Design*, pages 168–170, 2013.

[10] A. R. Agnihotri, S. Ono, and P. H. Madden. Recursive bisection placement: Feng Shui 5.0 implementation details. In *Proceedings of the ACM International Symposium on Physical Design*, pages 230–232, 2005.