Short Papers

Variation-Aware Placement with Multi-Cycle Statistical Timing Analysis for FPGAs

Gregory Lucas, *Student Member, IEEE,* Chen Dong, *Student Member, IEEE,* and Deming Chen, *Member, IEEE*

***Abstract*—Deep submicron processes have allowed field-programmable gate arrays (FPGAs) to grow in complexity and speed. However, such technology scaling has caused FPGAs to become more susceptible to the effects of process variation. In order to obtain sufficient yield values, it is now necessary to consider process variation during physical design. It is common for FPGAs to contain designs with multi-cycle paths to help increase the performance, but current statistical static timing analysis (SSTA) techniques cannot support this type of timing constraint. In this paper, we propose an extension to block-based SSTA to consider multi-cycle paths. We then use this new SSTA to optimize FPGA placement with our tool VMC-Place for designs with multi-cycle paths. Experimental results show our multi-cycle SSTA is accurate to 0.59% for the mean and 0.0024% for the standard deviation. Our results also show that VMC-Place is able to reduce the 95% performance yield clock period by 15.36% as compared to VPR.**

***Index Terms*—Field-programmable gate array (FPGA) placement, multi-cycle paths, process variation, statistical static timing analysis.**

I. Introduction

The move to deep submicron processes has allowed field-programmable gate arrays (FPGAs) to significantly increase their size and performance. However, such a move has brought about a new challenge: process variation [2]. FPGAs face a unique challenge with minimizing the effects of process variation due to their reprogramability. The critical path of an application-specific integrated circuit (ASIC) can be tested after manufacturing, where the critical path of an FPGA cannot. Until recently, the effects of process variation were not as pronounced on FPGAs as compared to ASICs due to their regular architecture [3]. However, with larger FPGAs that run at faster speeds, process variation is now a factor. Therefore, it is necessary to consider process variation during the placement and routing stages of synthesis for FPGAs.

A common design technique in FPGAs is the use of complicated timing constraints, such as multi-cycle paths [4]. In order to be able to take advantage of these types of timing constraints, a statistical static timing analysis (SSTA) algorithm that supports them is required. The goal of SSTA is to find a pdf for the circuit delay. This pdf can then be used to find the performance yield of a circuit. The performance yield is defined to be the percentage of manufactured die that will function at a specific clock period. Mathematically, the performance yield, PY, can be defined as

$$PY = P(X_1 \leq y, X_2 \leq y, \cdots, X_n \leq y) \quad (1)$$

where X_n is the delay distribution at the outputs and y is the chosen clock period. Complicating this equation is the fact that the delay distributions exhibit both structural and spatial correlations that must be considered. In recent years, a number of SSTA algorithms have been proposed in the literature [5], however, they all ignore the issue of multi-cycle timing analysis.

In this paper, we propose a new variation-aware placement algorithm for FPGAs that scales well with circuit size. Our algorithm takes into account correlated variation, random variation, and multi-cycle timing constraints to obtain significantly better results than a single-cycle variation aware algorithm. We also propose a methodology for considering multi-cycle paths in a block-based SSTA framework. Our results show that our timing analysis algorithm is able to effectively handle multi-cycle timing constraints.

II. Problem Formulation and Motivation

Multi-cycle paths are a reality in modern industrial designs [4]. The ability to accurately identify and analyze them is key to attaining timing closure. As a motivational example we present the circuit shown in Fig. 1(a), circuit structure that is often found in high level and register transfer level synthesis. The circuit consists of two functional units (one multiplier and one adder), a multiplexer, and flip flops. Since multipliers take a longer time to complete their operation as compared to adders, a common technique to increase performance is to make the multiplier a multi-cycle functional unit. Fig. 1(c) shows a possible placement for this circuit, where CLB4 contains the MUX in Fig. 1(a), CLB2 and CLB1 are related to the multiplier, and CLB3 is related to the adder. This means that CLB1 contains a multi-cycle path CLB1–CLB2–CLB4, and CLB3 contains a single cycle path to CLB4. If the multi-cycle paths are not considered, then the longest path in the circuit will be the path from CLB1–CLB2–CLB4. A single cycle algorithm will try to optimize this circuit by placing CLB1 closer to CLB2, but when the design is actually run, the critical path will be the CLB3–CLB4 path. By considering multi-cycle paths, our algorithm is able to optimize the correct path, CLB3–CLB4, by moving CLB3 closer to CLB4 [Fig. 1(d)], thereby improving the performance of the circuit. As this example shows, the key to improving performance is being able to perform accurate multi-cycle SSTA during the placement optimization.

We formulate the multi-cycle path SSTA (MCSSTA) problem as follows.

a) *MCSSTA problem:* Given a statistical timing graph, $G(V, E)$, where each node, $v_i \in V$ (edge, $e_i \in E$), contains a random variable for the delay of the node(edge), find the maximum delay, $max(P_1, P_2, ..., P_n)$, over all paths in the circuit where an arbitrary set of the paths, defined as $MC(P)$, have a multi-cycle path constraint.

Manuscript received January 7, 2010; revised April 28, 2010. Date of current version October 20, 2010. This work was supported in part by the National Science Foundation, under Grants CCF 07-46608 and CCF 07-02501. A preliminary 4-page version of this paper was published at FPGA 2010 [1]. This paper was recommended by Associate Editor K. Bazargan.

The authors are with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: gmlucas2@illinois.edu; cdong3@illinois.edu; dchen@illinois.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2010.2056411

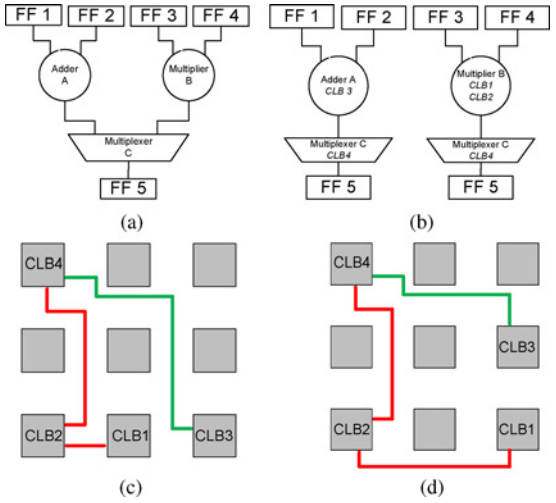

Fig. 1. Motivational circuit from high-level synthesis. (a) Original circuit. (b) Path decomposed circuit. (c) Single-cycle placement. (d) Multi-cycle placement.

The multi-cycle variation-aware FPGA placement problem is formulated as follows.

b) *MCSSTA-driven FPGA placement problem:* Given a circuit C, and a statistical timing graph, $G(V, E)$, place the CLBs to minimize the statistical critical path within the circuit, thereby increasing the PY.

III. Multi-Cycle SSTA

In this section, we first show how multi-cycle paths can be considered in SSTA using a block-based algorithm. Next, we extend our method so that it can be used for a principal components analysis (PCA) based timing analysis. We then describe the algorithm that is used to traverse the timing graph. To begin, we return to our motivational example as shown in Fig. 1(a) and decompose the circuit into two paths as shown in Fig. 1(b). The path A-C-FF5 is a single-cycle path, while the path B-C-FF5 is a two-cycle path. In order to find the pdf for the delay of this circuit, the max between two delay distributions that are of different cycle lengths must be found. In the following section, we describe how to find the max between two paths whose cycle constraints differ.

A. Max Between Different Cycle Paths

In order to calculate the max operation between two paths with different multi-cycle constraints, a normalizing operation is applied to the multi-cycle paths so that the delay distribution is expressed as a function of a single cycle. Equation (2) shows the normalizing operation

$$\mu_{\text{norm}} = \frac{\mu_o}{n} \qquad \sigma_{\text{norm}} = \frac{\sigma_o}{n} \tag{2}$$

where μ_o and σ_o are the original mean and standard deviation of the delay distribution and n is a normalizing constant that has been applied to the path. We offer the following property of normal distributions [6] to justify the normalizing operation where $\Phi(Z)$ is the cdf calculation for a normal distribution:

$$\Phi\left(\frac{x-\mu}{\sigma}\right) = \Phi\left(\frac{\frac{1}{n}x - \frac{1}{n}\mu}{\frac{1}{n}\sigma}\right). \tag{3}$$

Thus, by dividing the mean and standard deviation by the normalizing constant, we are able to find the performance yield as a function of the single-cycle clock period. However, by normalizing the delay distribution, the covariance between the delay distributions of the single and multi-cycle paths change. Equation (4) shows necessary change to the covariance

$$\text{Cov}\left(\frac{1}{n}X, Y\right) = \frac{1}{n}\text{Cov}(X, Y). \tag{4}$$

Therefore, the adjusted covariance can be found by dividing all the multi-cycle path covariances by the normalizing constant. Using (2) and (4) it is possible to find the max between the two paths A–C–FF5 and B–C–FF5 as follows:

$$\text{pdf}_{\text{FF5}} = \max\left(\frac{1}{n}(B+C), A+C\right) \tag{5}$$

with a correlation coefficient, ρ, of

$$\rho = \frac{\text{Cov}\left(\frac{1}{n}(B+C), A+C\right)}{\frac{1}{n}\sigma_{(B+C)}\sigma_{(A+C)}}. \tag{6}$$

B. Application to PCA

In this section, the normalization operation and the correlation change equations are extended for use in a PCA-based timing analysis. Principal component analysis simplifies the traversal of the timing graph to a PERT-like traversal by expressing each delay distribution as a function of its principal components as shown in

$$d = d_o + k_1 p_1' + \cdots + k_m p_m'. \tag{7}$$

The principal components are all independent, which significantly simplifies the tracking of correlation throughout the circuit. Three properties for expressing the delay distribution as a function of the principal components are given in [7].

Property 1: $\sigma_d^2 = \sum_{i=1}^{m} k_i^2$.

Property 2: $\text{Cov}(d, p_i') = k_i$.

Property 3: Let d_i and d_j be two random variables

$$d_i = d_i^o + k_{i1} p_1' + \cdots + k_{im} p_m'$$

then

$$d_j = d_j^o + k_{j1} p_1' + \cdots + k_{jm} p_m'$$

$$\text{Cov}(d_i, d_j) = \sum_{r=1}^{m} k_{ir} k_{jr}. \tag{8}$$

Through the use of property 1, the normalization operation for a multi-cycle path can be defined as

$$d_{norm} = \frac{d_0}{n} + \frac{k_1}{n} p_1' + \cdots + \frac{k_m}{n} p_m'. \tag{9}$$

The major advantage of PCA comes in the form of the covariance calculation. Property 3 can be directly used to calculate the new correlation between the two delay distributions since it uses the principal components. This means that no new equations are necessary for the correlation calculation.

C. Normalizing Constant Calculation

In this section, we show how the above normalizing constant is calculated. We present the calculation for all types of multi-cycle paths which include start multi-cycle (SMC), end multi-cycle (EMC), and phase shifted versions of both types of multi-cycle paths [8]. We look at two general cases when

$$Freq_{DST} = \frac{Freq_{SRC}}{C} + Offset \qquad \text{(SMC)}$$

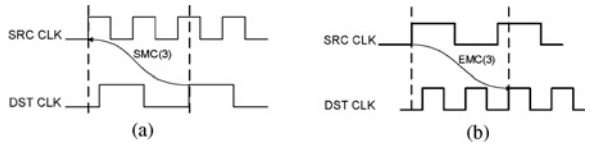

Fig. 2. Multi-cycle path types. (a) Start multi-cycle. (b) End multi-cycle.

and

$$Freq_{DST} = C * Freq_{SRC} + Offset \qquad \text{(EMC)}$$

where $Freq_{DST}$ is the destination register clock frequency, $Freq_{SRC}$ is the source register clock frequency, C is the multi-cycle constraint value, and $Offset$ is an offset value. Fig. 2 shows the two types of constraints. In Fig. 2(a), a phase shifted version of a start multi-cycle path with a multi-cycle constraint of three cycles is shown and in Fig. 2, a phase shifted version of an end multi-cycle path with a multi-cycle constraint of three cycles is shown.

The overall idea for calculating the normalizing constant is to calculate the fraction of time that the multi-cycle operation has to execute compared to the base clock frequency. Generally, the base clock is chosen to be the fastest clock in the design, so that the normalizing constant is greater than 1 for all paths; however, any clock can be chosen to be the base clock without any loss of accuracy. Mathematically, this is expressed as

$$n = \frac{\text{multi-cycle path execution time}}{\text{base clock period}}. \qquad (10)$$

More specifically, for the SMC constraint, the normalizing constant for the path is found using (11) when no offset is present and is found using (12) when an offset is present, assuming that the base clock period is T_{BASE_CLK}, T_{SRC_CLK} is the clock period for the source clock, and C is the multi-cycle constraint that is applied to the path. Due to the page constraint, the detailed equations for the EMC constraint are not presented, but are easily derived given the presented equations

$$n = \frac{C * T_{SRC\_CLK}}{T_{BASE\_CLK}} \qquad (11)$$

$$n = \frac{(C-1) * T_{SRC\_CLK} + Offset}{T_{BASE\_CLK}}. \qquad (12)$$

D. Multi-Cycle Graph Traversal

In this section, we show how the concepts shown above can be applied to a general timing graph using a block-based SSTA traversal.

1) *Timing Graph Setup:* A general timing graph consists of two or more vertices connected by edges where the vertex stores an arrival time value and an edge stores a delay value for an associated delay element such as a lookup table (LUT) or wire. Also added to the graph is a source node, which connects to all inputs, and a sink node, to which all outputs connect. For our multi-cycle SSTA-based timing graph, at each vertex, multiple arrival times composed of a mean and a standard deviation value for each timing constraint are stored. In the case of PCA, the standard deviation is represented by principal components which are stored at the corresponding vertex (edge). To be able to distinguish between single-cycle and multi-cycle paths, we follow an approach similar to [9] and add a list to each vertex (edge) that specifies all the multi-cycle constraints in which the vertex (edge) participates.

Algorithm 1 Modified PERT-like traversal

```
for each cycle constraint, n, at node v do
    pdf_{max,n,v} = 0
    for each input edge e_in do
        if n ∈ e_in then
            pdf_input = pdf_{source(e_in)} + pdf_{e_in}(n) * 1/n
            pdf_{max,n,v} = max(pdf_{max,n,v}, pdf_input)
        end if
    end for
end for
```

Algorithm 2 Sink max algorithm

```
final_max = 0;
for each cycle constraint, n at the sink do
    final_max = max(final_max, pdf_{max,n,sink})
end for
```

2) *Traversal:* We propose a modified PERT-like traversal in order to consider multi-cycle constraints during block-based SSTA. The algorithm maintains the breadth-first traversal of the original PERT-like traversal [7]; however, at each vertex, Algorithm 1 is executed.

When the modified PERT-like traversal has been completed, the sink node contains a delay distribution for each timing constraint in the circuit. In the example of Fig. 1(c), the sink node would contain two delay distributions, one for the 1 cycle constraint and one for the 2 cycle constraint. Due to the fact that PCA is being used and that the distributions were normalized during the timing analysis, the max between the different cycle constraints can be directly calculated as shown in Algorithm 2, where $\text{pdf}_{\text{max},n,\text{sink}}$ is the pdf of the delay for cycle constraint n.

This results in a pdf that can be used to find the PY for a given clock frequency. The merging across the delay distribution for each timing constraint in the timing graph is necessary to ensure an accurate answer due to the significant amount of spatial and structural correlation that can be present between different clock domains.

IV. SSTA Driven Multi-Cycle Placement

In this paper, we use the MCSSTA-driven CAD flow shown in Fig. 3. Each benchmark circuit goes through technology independent logic optimization using SIS [10] and is technology-mapped to 6-LUTs using the multi-cycle mapper [4]. The mapped netlist is then fed into T-VPACK and VPR [11], which perform timing-driven packing (i.e., clustering LUTs into the CLBs) and placement.

We target the cost function in VPR and enhance it to consider process variation and multi-cycle paths. Given that the PY depends on both the mean and variance of the path delay, the deterministic critical path delay can no longer serve as the absolute measure of performance since it is possible for near-critical paths to be statistically critical. Therefore, it is important to compute the slack and criticality statistically during placement.

After performing a MCSSTA traversal, the statistical arrival time and statistical required time of each timing node, as well

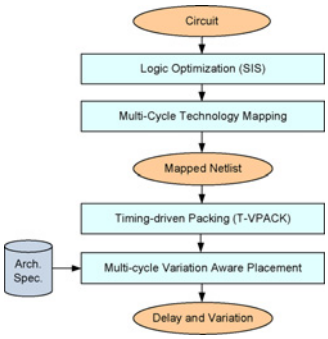

Fig. 3. CAD evaluation flow.

Algorithm 3 Statistical multi-cycle FPGA placement

```
1: Compute Correlation Matrices
2: Create Timing Graph and Load Principal Components
3: Random Placement;
4: Initialize Temperature;
5: while ExitCriterion() == False do
6:    Statistical Multi-Cycle Timing Analysis();
7:    Compute Statistical Criticality Across Cycles
8:    while InnerLoopCriterion () == False do
9:       New Swap;
10:      Compute Deterministic ΔCost
11:      Assess Swap(ΔCost, temperature)
12:   end while
13:   Update Temperature;
14: end while
```

as the statistical critical path delay can be computed. The statistical criticality for each pin j in net i is then computed, considering variation in the slack and critical path according to

$$Crit_{stat}(i,\, j) = 1 - \frac{\mu_{slack}(i,\, j) - 3\sigma_{slack}(i,\, j)}{\mu_{d\_max} + 3\sigma_{d\_max}} \tag{13}$$

where $\mu_{slack}(i,\, j)$ and $\sigma_{slack}(i,\, j)$ are the mean and standard deviation of the statistical slack at jth pin of net i. μ_{d_max} and σ_{d_max} are mean and standard deviation of the statistical critical path. We derive the statistical criticality function in this way so that when two slacks have a similar mean but different variations, the term assigns larger criticality to the path with the greatest variation, weighting it more heavily in the future iterations.

Through experimentation, we have found that straightforward modification of the cost function leads to excessive runtime. This is due to the computational cost of the MCSSTA traversal. In order to reduce the runtime, we propose a modified flow where an MCSSTA is performed once per temperature to calculate the statistical criticality for each timing node. Within temperatures, we then perform a multi-cycle deterministic timing analysis to determine the accept/reject decision for a move. We find that this approach is able to significantly reduce the runtime of our placement flow with minimal degradation of the placement result.

Algorithm 3 adapted from [11] shows the overall process of our variation-aware placer. The first step in this algorithm is to capture spatial correlations in the form of correlation matrices. Each variation parameter has its own correlation matrix. Principal components (PC) analysis is then used to generate the uncorrelated PCs for each location. Since principal components are computed based on physical location, the timing graph is updated for CLBs that are moved during a simulated annealing move. In this paper, we assume that devices within the same CLB are perfectly correlated. Sensitivities of all circuit components, which capture device performance changes due to process parameter variation, are pre-characterized using HSPICE and stored in the architecture file as input.

V. Experimental Results

In this section we perform two sets of experiments. First, we test the accuracy of our MCSSTA through Monte Carlo simulations in MATLAB; and second, we implement our multi-cycle variation-aware placement algorithm in the VPR framework to show the performance improvements that can be obtained.

We test our algorithm on the multi-cycle benchmarks from [4] as well as several larger benchmarks from VPR5 that were created using the same methodology. Each benchmark goes through the design flow as shown in Fig. 3. In the benchmarks, multi-cycle paths make up 97.47% of the paths and single-cycle paths make up 2.53% of the paths on average. For the experiments, we use a cluster size of 10. All the experiments were run on a computer with an Intel Core 2 2.0 GHz processor and 2 GB of RAM running Debian Linux.

We model the effect of variation on each component in the CLB, as well as wire delay for a 32 nm process (based on the predictive technology model [12]). As was stated earlier, we run HSPICE simulations to determine each component's propagation delay. On top of the simulated delay, we assume devices have 10% random variation and 10% correlated variation with three variation sources, namely gate length, doping concentration, and oxide thickness. For doping concentration we assume all variation is random. Interconnects are considered to have 10% random variation.

A. Multi-Cycle SSTA Comparison

To test the accuracy of MCSSTA, the correlation matrix, timing graph (including multi-cycle constraints), and delay sensitivities for each process parameter are written to a MATLAB file. From this information, one thousand correlated samples are generated for every timing edge in the timing graph, and a comparison is made between the resulting mean and standard deviation from MCSSTA and the mean and standard deviation from our Monte Carlo simulation. On average, we found that the mean value is off by 0.59% and the standard deviation value is off by 0.0024%. These are very comparable to the values reported in [7] for a single-cycle SSTA.

B. Multi-Cycle Variation-Aware Placement Comparison

For the second set of experiments we run two different comparisons on the benchmarks through our new version of placement, named VMC-Place, that is variation-aware with MCSSTA. In the first set, we compare VMC-Place against the original VPR placement (S-VPR). For this comparison, we compare VMC-Place with the full MCSSTA engine and also the hybrid VMC-Place that uses both an MCSSTA and a deterministic multi-cycle timing analysis in order to decrease runtime. Second, we compare VMC-Place against the original VPR augmented with a multi-cycle aware deterministic timing

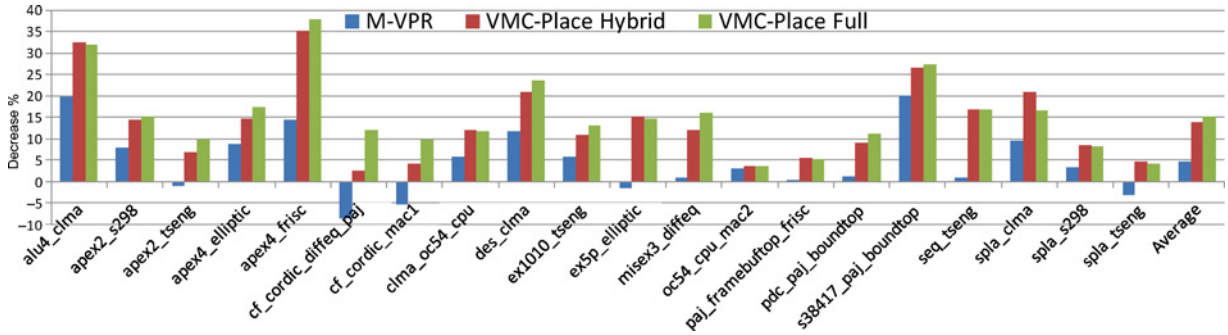

Fig. 4. Clock period reduction of three different programs compared to S-VPR for each benchmark.

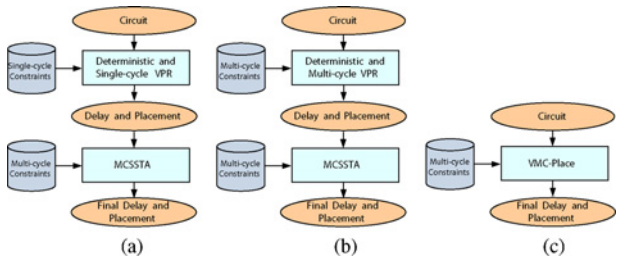

Fig. 5. Comparison methodology. (a) S-VPR. (b) M-VPR. (c) VMC-Place.

analysis engine (M-VPR). The setup for each comparison is shown graphically in Fig. 5. Through these three comparisons the need for considering variation, and multi-cycle paths together to obtain performance increases is shown. Also shown is the ability of our hybrid timing analysis flow to reduce runtime without sacrificing performance.

Our results are shown in Fig. 4. The figure shows the clock period reduction compared to S-VPR for each benchmark for the 95% performance yield clock period. On average, M-VPR reduces the clock period by 4.70%, VMC-Place with the full MCSSTA engine reduces the clock period by 15.36%, and VMC-Place with the hybrid timing engine reduces the clock period by 13.88%. Interestingly, the figure shows that for some benchmarks, M-VPR actually provides a worse solution than the original VPR. This result is directly due to the fact that M-VPR and S-VPR cannot account for variations during its optimization. Our results show that M-VPR is able to reduce the mean delay for the benchmarks by 5.8% but that the total variation increases by 18.9% which offsets any gain obtained by considering the multi-cycle paths for some benchmarks. The results show that we are able to attain significant improvements in the clock period when multi-cycle paths and variation are considered. They also show that by using our hybrid timing analysis approach that significant performance is not lost.

We found that VMC-Place with the full MCSSTA timing engine incurs a significant runtime overhead of 22.49× as compared to S-VPR and does not scale well. However, by switching to the hybrid timing analysis engine we were able to reduce the runtime overhead to 7.61× with only a 1.48% loss in clock period reduction.

VI. Conclusion and Future Work

In this paper, we have made a number of contributions. First, we have proposed a method for considering multi-cycle paths during SSTA. Second, we have used MCSSTA to create VMC-Place, a new version of the VPR placement tool which supports multi-cycle paths and is variation-aware. We have also shown that it is possible to make the tool scale well with a hybrid timing analysis approach. Through our experiments, we have shown that our MCSSTA algorithm is accurate, and that it is possible to achieve significant improvements in FPGA placement by considering multi-cycle paths. To the best of our knowledge this is the first published SSTA algorithm that supports multi-cycle paths. Our VMC-Place is also the first variation-aware placement algorithm to support multi-cycle paths. In the future, we plan to extend VMC-Place to include routing and non-Gaussian parameter variations.

Acknowledgment

The authors would like to thank L. Wan at the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, for his help in collecting data.

References

[1] G. Lucas, C. Dong, and D. Chen, "Variation-aware placement for FPGAs with multi-cycle statistical timing analysis," in *Proc. FPGA*, 2010, pp. 177–180.
[2] S. Nassif, "Design for variability in DSM technologies [deep submicron technologies]," in *Proc. ISQED*, 2000, p. 451.
[3] S. Sivaswamy and K. Bazargan, "Statistical analysis and process variation-aware routing and skew assignment for FPGAs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 1, no. 1, p. 4, Mar. 2008.
[4] L. Cheng, D. Chen, M. D. F. Wong, M. Hutton, and J. Govig, "Timing constraint-driven technology mapping for FPGAs considering false paths and multi-clock domains," in *Proc. ICCAD*, 2007, pp. 370–375.
[5] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer, "Statistical timing analysis: From basic principles to state of the art," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 4, no. 4, pp. 589–607, Apr. 2008.
[6] S. Ross, *A First Course in Probability*. Saddle River, NJ: Pearson Prentice-Hall, 2006.
[7] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," in *Proc. ICCAD*, 2003, p. 621.
[8] Altera Corporation. (2008, Jul.). *AN481: Applying Multicycle Exceptions in the Timing Quest Timing Analyzer* [Online]. Available: http://www.altera.com/literature/an/an481.pdf
[9] M. Hutton, D. Karchmer, B. Archell, and J. Govig, "Efficient static timing analysis and applications using edge masks," in *Proc. FPGA*, 2005, pp. 174–183.
[10] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis," Univ. California, Berkeley, Tech. Memo. UCB/ERL M92/41, 1992.
[11] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Boston, MA: Kluwer, 1999.
[12] *Predictive Technology Model*. (2009, Sep.) [Online]. Available: http://www.eas.asu.edu/~ptm