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

Gregory Lucas Chen Dong Deming Chen
Department of Electrical and Computer Engineering, University of Illinois Urbana-Champaign
gmlucas2, cdong3, dchen@illinois.edu

ABSTRACT

Deep submicron processes have allowed 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 SSTA techniques cannot support this type of timing constraint. 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. Our 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 improve the clock period by 9.42% or the performance yield by 68.51% compared to a single-cycle variation-aware placer.

Categories and Subject Descriptors

B.7 [**Integrated Circuits**]: Design Aids—*placement and routing*

General Terms

Algorithms

1. INTRODUCTION

The move to deep submicron processes has allowed FPGAs to significantly increase their size and performance. However, the increase has not been without trade offs. Moving to submicron processes has brought about a new challenge: process variation. Until recently, the effects of process variation were not as pronounced on FPGAs as compared to ASICs due to their regular architecture [10]. However, with larger FPGAs that run at faster speeds, process variation is now a factor.

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 [8] [3], however, they all ignore the issue of multi-cycle timing analysis.

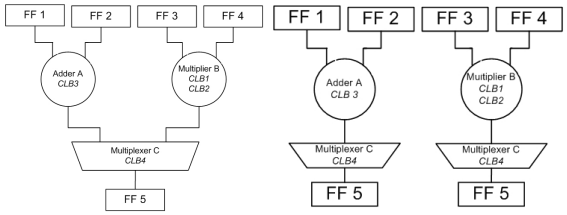
In this paper, we first propose an SSTA algorithm that can consider multi-cycle paths. We then propose a new variation-aware placement algorithm for FPGAs. Our algorithm takes into account correlated variation, random variation, and multi-cycle timing constraints to obtain significantly better results.

2. FORMULATION AND MOTIVATION

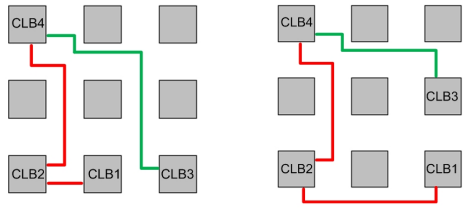
Multi-cycle paths are a reality in modern industrial designs [4]. The ability to accurately identify and analyze them is key to attaining timing closure. As a motivational example we present the circuit shown in Figure 1(a), a circuit structure that is often found in high level and RTL synthesis. The 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. Figure 1(c) shows a possible placement for this circuit, where CLB 4 contains the MUX in Figure 1(a), CLB2 and CLB1 are related to the multiplier and CLB3 is related to the adder. This means that CLB 1 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–CLB 2–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 (Figure 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 placement.

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

FPGA'10, February 21–23, 2010, Monterey, California, USA.



(a) Original circuit (b) Path decomposed circuit



(c) Single-cycle placement (d) Multi-cycle placement

Figure 1: Motivational circuit from high-level synthesis

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

MCSSTA-driven 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.

3. 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 component based (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 Figure 1(a) and decompose the circuit into two paths as shown in Figure 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, based on Clark's Approximation [5], 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.

3.1 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

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

where μ_o and σ_o are the original mean and standard deviation of the delay distribution and n is a multi-cycle constraint that has been applied to the path. We offer the following property of normal distributions [9] 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 multi-cycle path cycle constraints, 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.

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

Therefore, the adjusted covariance can be found by dividing all the multi-cycle path covariances by the number of cycles they are given to propagate.

3.2 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 Equation 5 [3].

$$d = d_o + k_1 p_1^{'} + \cdots + k_m p_m^{'} \tag{5}$$

Two properties for expressing the delay distribution as a function of the principal components are given in [3]:

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

Property 2: Let d_i and d_j be two random variables:

$$d_i = d_i^o + k_{i1} p_1^{'} + \cdots + k_{im} p_m^{'}$$
$$d_j = d_j^o + k_{j1} p_1^{'} + \cdots + k_{jm} p_m^{'}$$

then

$$\mathrm{Cov}(d_i, d_j) = \sum_{r=1}^{m} k_{ir} k_{jr}$$

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

$$d_{norm} = \frac{d_0}{n} + \frac{k_1}{n} p_1^{'} + \cdots + \frac{k_m}{n} p_m^{'} \tag{6}$$

The major advantage of PCA comes in the form of the covariance calculation. Property 2 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.

3.3 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.

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 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 [7] and add a list to each vertex (edge) that specifies all the multi-cycle constraints in which the vertex (edge) participates.

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; 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 Figure 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

Algorithm 1 Modified PERT-like traversal

for each cycle constraint, n, at node v **do**
  $\text{pdf}_{max,n} = 0$
  for each input edge e_{in} **do**
    if $n \in e_{in}$ **then**
      $\text{pdf}_{input} = \text{pdf}_{source(e_{in})} + \text{pdf}_{e_{in}}(n) * \frac{1}{n}$
      $\text{pdf}_{max,n,v} = \max(\text{pdf}_{max,n,v}, \text{pdf}_{input})$
    end if
  end for
end for

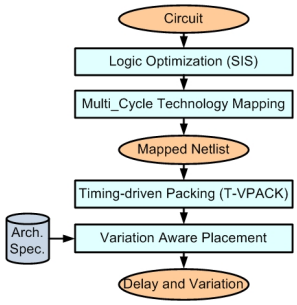


Figure 2: CAD evaluation flow

analysis, the max between the different cycle constraints can be directly calculated as shown in Algorithm 2, where $\text{pdf}_{max,n,sink}$ is the pdf of the delay for cycle constraint n.

Algorithm 2 Sink max algorithm

final_max = 0;
for each cycle constraint, n at the sink **do**
  final_max = max(final_max, $\text{pdf}_{max,n,sink}$)
end for

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 is present between different clock domains.

4. SSTA DRIVEN MULTI-CYCLE PLACEMENT

In this work, we use the MCSSTA-driven CAD flow shown in Figure 2. Each benchmark circuit goes through technology independent logic optimization using SIS [6] and is technology-mapped to 6-LUTs using the multi-cycle mapper [4]. The mapped netlist is then fed into T-VPACK and VPR [2], 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 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 Equation 7.

$$Crit_{stat}(i,j) = 1 - \frac{\mu_{slack}(i,j) - 3\sigma_{slack}(i,j)}{\mu_{d\_max} + 3\sigma_{d\_max}} \quad (7)$$

where $\mu_{slack}(i,j)$ and $\sigma_{slack}(i,j)$ are the mean and standard deviation of the statistical slack at j^{th} 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.

Algorithm 3 adapted from [2] shows the overall process of our variation-aware placer.

Table 1: MCSSTA Accuracy

Benchmark	Monte Carlo		MCSSTA		% Diff	
Name	μ(ns)	σ(ns)	μ(ns)	σ(ns)	μ	σ
alu4_clma	5.11	0.19	5.09	0.18	0.48	0.06
alu4_diffeq	6.36	0.22	6.26	0.22	1.53	0.06
alu4_tseng	5.80	0.22	5.80	0.20	0.002	0.25
apex2_s298	8.77	0.30	8.75	0.33	0.14	-0.36
apex2_tseng	mem	mem	5.96	0.19	–	–
apex4_elliptic	mem	mem	5.24	0.19	–	–
apex4_frisc	5.55	0.20	5.49	0.19	0.95	0.27
des_clma	mem	mem	5.32	0.18	–	–
ex1010_tseng	7.99	0.27	7.89	0.29	1.24	-0.28
ex5p_diffeq	5.70	0.19	5.64	0.19	0.97	-0.02
ex5p_elliptic	mem	mem	5.64	0.17	–	–
misex3_diffeq	mem	mem	6.31	0.23	–	–
misex3_tseng	5.60	0.22	5.60	0.21	-0.007	0.18
pdc_tseng	mem	mem	8.68	0.29	–	–
seq_diffeq	mem	mem	6.82	0.26	–	–
seq_tseng	5.24	0.19	5.24	0.17	0.08	0.40
spla_clma	8.36	0.29	8.32	0.32	0.51	-0.39
spla_diffeq	8.32	0.30	8.32	0.30	-0.003	-0.03
spla_s298	7.96	0.28	7.86	0.29	1.24	-0.11
spla_tseng	mem	mem	8.23	0.29	–	–
Average					**0.59**	**0.0024**

Algorithm 3 Statistical multi-cycle FPGA placement

Compute Correlation Matrices
Create Timing Graph and Load Principal Components
Random Placement;
Initialize Temperature;
while ExitCriterion() == False **do**
  Statistical Multi-Cycle Timing Analysis();
  Compute Statistical Criticality Across Cycles
  while InnerLoopCriterion () == False **do**
    New Swap;
    Compute Statistical $\Delta Cost$
    Assess Swap($\Delta Cost$, temperature)
  end while
  Update Temperature;
end while

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 work, 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.

5. EXPERIMENTAL RESULTS

In this section we perform two sets of experiments. First, we test the accuracy of our multi-cycle SSTA (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. We test our algorithm on the multi-cycle benchmarks from [4]. For the experiments, we use a cluster size of 10.

Table 2: VMC-Place Results - Variation Aware

Benchmark		Single-Place		VMC-Place		Clock Period for 95% Yield			
Name	LUTs	μ(ns)	σ(ns)	μ(ns)	σ(ns)	D (ns)	VMC-Place (ns)	% Improve	PY Improve %
alu4_clma	4499	5.96	0.21	4.79	0.16	6.31	5.06	19.86	94.99
alu4_diffeq	1695	5.03	0.17	4.51	0.15	5.31	4.76	10.27	88.73
alu4_tseng	1658	5.09	0.18	4.58	0.16	5.40	4.84	10.30	85.93
apex2_s298	2258	6.37	0.22	5.38	0.19	6.74	5.71	15.37	94.85
apex2_tseng	2681	6.13	0.18	5.32	0.18	6.44	5.63	12.56	94.60
apex4_elliptic	2566	5.28	0.18	5.03	0.17	5.59	5.31	4.89	38.20
apex4_frisc	1812	5.75	0.19	5.24	0.17	6.07	5.52	9.02	83.24
des_clma	6380	5.49	0.37	5.01	0.13	6.10	5.23	14.30	70.70
ex1010_tseng	4126	7.50	0.27	7.03	0.25	7.95	7.46	6.19	51.44
ex5p_diffeq	1447	5.80	0.20	4.87	0.14	6.13	5.12	16.50	94.95
ex5p_elliptic	2404	4.56	0.43	5.28	0.18	5.27	5.58	-5.89	-4.08
misex3_diffeq	2523	4.58	0.16	4.37	0.15	4.85	4.63	4.64	33.11
misex3_tseng	1603	5.54	0.19	4.51	0.14	5.86	4.76	18.86	94.99
pdc_tseng	4560	7.91	0.26	7.39	0.24	8.33	7.79	6.50	61.99
seq_diffeq	2716	4.38	0.40	4.62	0.18	5.04	4.92	2.38	3.90
seq_tseng	1796	5.05	0.17	4.53	0.15	5.33	4.78	10.22	88.63
spla_clma	6162	7.76	0.27	6.97	0.24	8.21	7.37	10.19	87.07
spla_diffeq	3330	7.18	0.23	6.67	0.23	7.57	7.05	6.88	66.31
spla_s298	3954	7.39	0.24	6.80	0.24	7.80	7.20	7.70	73.38
spla_tseng	4077	6.99	0.24	6.48	0.22	7.40	6.84	7.53	67.17
Average						**6.39**	**5.78**	**9.42**	**68.51**

We model the effect of variation on each component in the CLB, as well as wire delay for a 32nm process (based on the Predictive Technology Model (PTM) [1]). 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. Interconnects are considered to have 10% random variation.

5.1 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 following a normal distribution, and a comparison is made between the results from MCSSTA and the results from our Monte Carlo simulation. Table 1 shows the results from the comparison.

It can be seen that MCSSTA closely matches the Monte Carlo results. On average, 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 [3] for a single-cycle SSTA. The benchmarks marked "mem" could not be completed due to memory limitations of the Monte Carlo simulation.

5.2 Multi-cycle variation-aware placement comparison

For the second experiment we compare our new version of placement, named VMC-Place, that is variation-aware with MCSSTA against a single-cycle variation-aware version of VPR (Single-Place). Table 2 shows the results. The size of each benchmark is shown in the column labeled LUTs. For the "Single-Place" run we perform a single-cycle placement that is driven by a single-cycle SSTA engine. For the "VMC-Place" column we run VMC-Place by passing it a multi-cycle timing constraints file. A MCSSTA is then performed on both placements. The column "Clock Period for 95% Yield"' represents the clock period that is needed for each placement to achieve a 95% performance yield (PY) rate. The last column, titled "'PY Improvement"' shows the PY improvement of VMC-Place when both are clocked at the 95% yield clock period. The results show that VMC-Place improves the placement quality by reducing the clock period for 95% PY by 9.42% or by increasing the PY by 68.51%. We also compare against the original deterministic VPR augmented with a multi-cycle aware deterministic timing analysis engine. We find that on average, the clock period for 95% PY can reduced by 5.27% or the PY can be increased by 43.91%. Detailed results are omitted due to the page limitation.

6. CONCLUSIONS AND FUTURE WORK

We have presented MCSSTA, a new accurate multi-cycle capable SSTA algorithm. Using MCSSTA we created VMC-Place, a new version of the VPR placement tool which supports multi-cycle paths and is variation-aware. Through experiments, we have shown that it is possible to achieve significant improvements in FPGA placement by considering multi-cycle paths.

7. ACKNOWLEDGMENTS

This work is partially supported by NSF grant CCF 07-46608 and NSF grant CCF 07-02501.

8. REFERENCES

[1] Predictive Technology Model, September 2009. [Online]. Available: http://www.eas.asu.edu/~ptm/.

[2] V. Betz, J. Rose, and A. Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, 1999.

[3] H. Chang and S. S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. In *ICCAD*, 2003.

[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 *ICCAD*, 2007.

[5] C. E. Clark. The greatest of a finite set of random variables. *Operations Research*, 1961.

[6] E. M. Sentovich et. al. SIS: A system for sequential circuit synthesis. Technical report, 1992.

[7] M. Hutton, D. Karchmer, B. Archell, and J. Govig. Efficient static timing analysis and applications using edge masks. In *FPGA*, 2005.

[8] J. Le, X. Li, and L. Pileggi. STAC: Statistical timing analysis with correlation. In *DAC*, 2004.

[9] S. Ross. *A First Course in Probability*. Pearson Prentice Hall, Upper Saddle River, NJ, 2006.

[10] S. Sivaswamy and K. Bazargan. Statistical analysis and process variation-aware routing and skew assignment for FPGAs. *Trans. Reconfigurable Technology Systems*, 2008.