FPGA PERFORMANCE OPTIMIZATION VIA CHIPWISE PLACEMENT CONSIDERING PROCESS VARIATIONS

Lerong Cheng[1]*, Jinjun Xiong*[1]*, Lei He*[1]*, Mike Hutton*[2]

[1]EE Department, University of California, Los Angeles, CA 90095, USA
[2]Altera Corporation, San Jose, CA 95134, USA

ABSTRACT

Both custom IC and FPGA designs in the nanometer regime suffer from process variations. But different from custom ICs, FPGAs' programmability offers a unique design freedom to leverage process variation and improve circuit performance. We propose the following variation aware chipwise placement flow in this paper. First, we obtain the variation map for each chip by synthesizing the test circuits for each chip as a preprocessing step before detailed placement. Then we use the trace-based method to estimate the performance gain achievable by chipwise placement. Such estimation provides a lower bound of the performance gain without detailed placement. Finally, if the gain is significant, a variation aware chipwise placement is used to place the circuits according to the variation map for each chip. Our experimental results show that, compared to the existing FPGA placement, variation aware chipwise placement improves circuit performance by up to 19.3% for the tested variation maps.

1. INTRODUCTION

Design in the nanometer regime has witnessed tremendous challenges resulting from process variations. To combat process variations, statistical static timing analysis (SSTA) [1, 2] and statistical circuit optimization [3, 4] have been studied. FPGA architecture evaluation has been conducted with process variations [5], and the stochastic placement for FPGAs has been proposed in this conference [6]. However, all of these papers assume the same physical design applied to all chips, and do not consider chipwise physical design.

The programmability of FPGAs offers a unique opportunity to leverage process variation and improve circuit performance. For custom ICs, physical design for a targeted circuit must be the same for all chips. For FPGAs, however, we can potentially place (and route) each pre-fabricated FPGA chip differently for the same application. Compared to manufacturing level process control, this chipwise physical design technique only involves post-silicon design optimization and is more cost-effective. In this paper, we consider placement and propose the following variation aware chipwise design flow (see Fig. 1). For a given set of FPGA chips, we first generate the variation map for each chip, which may be obtained by synthesizing test circuits for each chip. Based on the variation map, we then estimate the potential delay improvement of chipwise placement. If the improvement is large, it is worthwhile to perform placement for each chip; otherwise, we just use the conventional design flow, which uses the same placement and route for all chips.

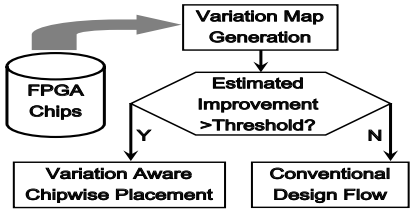

Fig. 1. Design flow of variation aware chipwise placement.

While synthesis of test circuits to generate the variation map is an ongoing research, we develop the following two key components for the above chipwise design flow. First, we propose an efficient high-level trace-based estimation method to evaluate the potential performance gain achievable through chipwise FPGA placement without detailed placement. Such estimation will provide a lower bound of the performance gain of detailed placement. Second, we develop a variation-aware detailed placement algorithm $vaPL$ within the VPR framework [7] to leverage process variation and optimize performance for each chip. Chipwise placement $vaPL$ is deterministic for each chip when the chip's variation map is known, and leads to different placements for different chips of the same application. Compared to the existing FPGA placement practice, our experimental results show that chipwise FPGA placement improves circuit performance by up to 12% for the tested variation maps.

The rest of the paper is organized as follows. Section 2 presents preliminaries and our problem formulation. Section 3 develops a trace-based estimation method to assess the potential performance gain of chipwise placement for FPGA.

This paper is supported in part by an NSF grant CCR-0306682. Address comments to lhe@ee.ucla.edu.

Section 4 discusses details of the chipwise placement technique with experimental results. We conclude this paper and discuss future work in section 5.

2. PROBLEM FORMULATION

We choose the island-style SRAM-based FPGA architecture. A logic block is a cluster of fully connected basic logic elements (BLEs), and each BLE contains one lookup table (LUT) and one flip-flop. Logic blocks are surrounded by programmable routing channels. The routing wires in either horizontal or vertical routing channels are segmented by fully buffered routing switches. Without loss of generality, we call different types of resources in FPGAs *circuit elements*, including LUTs, flip-flops, buffers, multiplexers, and input and output pads. We assume all interconnect segments span four logic blocks with all tri-state buffer connection box, and use LUT size of 4 and cluster size of 10.

2.1. Sources of Process Variation

Three sources of process variation are considered. (1) *Inter-chip variation* models the variation caused by global variation that is shared for all device parameters within the chip, hence it is the same for all devices within a chip, but may be different for different chips. (2) *Intra-chip spatial variation* models location-dependent variation within the chip, hence it may be different for devices at different locations within the same chip. (3) *Uncorrelated random variation* models the residual variation that is not explainable by the above two sources of random variations.

The process parameter of interest X, which can be either a physical parameter such as channel length L_{eff}, or a parametric quantity such as threshold voltage V_{th}, is a function of the above three sources of random variation. The overall variation for X can be captured by the following first-order variation model

$$X = X_0 + \Delta X = X_0 + X_g + X_s + X_r, \quad (1)$$

where X_0 is the mean value of X; X_g, X_s and X_r model the inter-chip global variation, intra-chip spatial variation, and the uncorrelated random variation, respectively. Same as [2], we assume that X_g, X_s and X_r all follow a normal distribution with zero mean values. Moreover, X_g, X_s and X_r are mutually independent. The total variance of X is thus given by

$$\sigma_X^2 = \sigma_{\Delta X}^2 = \sigma_g^2 + \sigma_s^2 + \sigma_r^2, \quad (2)$$

where σ_g, σ_s, and σ_r are the variance of X_g, X_s and X_r, respectively.

2.2. Modeling of Spatial Correlation

It has been observed that devices that are physically close to each other are more likely to have similar characteristics than devices that are far apart. This phenomenon is called spatial correlation. We model the spatial correlation as a *homogeneous and isotropic random field* so that the spatial correlation between any two points depends only on the distance v between them. In another word, the spatial correlation can be described by a spatial correlation function $\rho(v)$. The *spatial correlation distance* is defined as the distance $\overline{v}$ so that beyond that distance, the spatial correlation $\rho(\overline{v})$ becomes sufficiently small (e.g., less than 1%).

For any number of chosen points on the chip, the joint spatial variation $\mathbf{X}=(X_1, X_2, ..., X_M)^T$ follows a standard multivariate Gaussian process with respect to their respective physical locations on the chip. Therefore, to fully characterize the M-dimensional Gaussian distribution, we only need to know its correlation matrix, which can be easily generated by knowing the spatial correlation function $\rho(v)$.

In [5], process parameters at different locations are assumed to be spatially independent. But this is not true in general. For example, for L_{eff} at two different locations n and m, we have

$$\begin{aligned} L_n &= L_{n,0} + \Delta L_n = L_{n,0} + L_g + L_{s,n} + L_{r,n}, \\ L_m &= L_{m,0} + \Delta L_m = L_{m,0} + L_g + L_{s,m} + L_{r,m}, \end{aligned} \quad (3)$$

It is clear that L_n and L_m not only share the same global variation modeled by L_g, but also share the spatially correlated variation modeled by $L_{s,n}$ and $L_{s,m}$, respectively. The covariance between L_n and L_m is given by

$$cov(L_n, L_m) = E(\Delta L_n \Delta L_m) = \sigma_g^2 + \sigma_s^2 \cdot \rho(v_{n,m}), \quad (4)$$

where $v_{n,m}$ is the distance between location n and m. Therefore, simply ignoring the spatially correlated variation in estimating the chip performance cannot be accurate.

2.3. Region-based Variation Map

The *variation map* of an FPGA chip describes the detailed device and interconnect performance on the chip after its fabrication. Ideally, the variation map is a smooth but complicated function of the location in the chip. For practical use, we adopt a region-based variation map as an approximation. That is, we divide the FPGA chip into a set of regions, and assume each region has the same performance characteristics. The performance characteristics for each region can be obtained by synthesizing test circuits for each chip as a pre-processing step before detailed placement. Obviously, the finer granularity of the region, the more accurate of the region-based variation map to the real variation map.

2.4. Problem Formulation

Without considering process variation, the existing deterministic placement approach, denoted as $dtPL$, finds one "best" possible placement solution under the worst-case timing models in order to guard-band designs for all possible manufacturing conditions. Because the same placement solution is then applied to all chips, as a result of process variation, chips with the same placement solution may in fact exhibit different performance. In the absence of speed-binning, all these chips have to be labeled and sold with the worst performance, thus decreasing profit.

An alternative approach, as proposed in this work, is to leverage the programmability offered by FPGA to perform chipwise placement for each FPGA chip according to its variation map. This approach is denoted as $vaPL$. It requires first to characterize individual FPGA chip to obtain its *variation map* before detailed placement of the design; and then finds the best placement solution for each FPGA chip according to its own variation map. Under the approach $vaPL$, each FPGA chip may have different placement solutions, because each FPGA chip can have different variation maps resulting from process variation. Through this chipwise placement, we can potentially achieve the best possible performance for each FPGA chip.

To realize this potential performance optimization, the $vaPL$ approach requires a change of the existing design practice. For example, to use $vaPL$, designers have to characterize each FPGA chip to obtain the variation map first. To justify such an effort, we need to understand quantitatively that (1) how much the potential performance gain we can achieve; and (2) if the potential gain is large, how we could exploit it for FPGA performance optimization. We will solve the above two questions in the following sections.

3. TRACE-BASED ESTIMATION

In the following, we develop a trace-based estimation of the potential performance gain for chipwise placement for FPGA designs, but without detailed placement. For a benchmark to be implemented in a given FPGA architecture, we use *Ptrace* [5] to obtain the statistical profile, or trace, of this benchmark. Trace includes switching activity, circuit element utilization rate, and a set of near critical paths with their path structures. For the purpose of estimation, we assume chipwise FPGA placement only changes the location of paths, but not their layout structures, i.e., critical paths for all chips of the same application have the same layout structures. Results following this assumption would, theoretically, give a lower bound on the potential performance gain achievable by detailed chipwise placement. This is because the layout of the critical path may be changed by the detailed placement and the fixed layout could have a longer delay than the that optimized by the detailed placement.

3.1. Spatially Correlated Critical Path Delay

We assume that the variation in circuit element delays is mainly due to the parametric variations in effective channel length L_{eff} and threshold voltage V_{th}. We employ the first order canonical delay model [2] to model circuit element delays in FPGAs:

$$d_{i,n} = d_{i,0} + t_{i,1} \cdot \Delta L_{i,n} + t_{i,2} \cdot \Delta V_{i,n}, \tag{5}$$

where $d_{i,n}$ is the delay of the i^{th} type circuit element at location n; $d_{i,0}$ is the nominal delay, $t_{i,1}$ and $t_{i,2}$ are its delay sensitivities with respect to L_{eff} and V_{th}, respectively. The value of $t_{i,1}$ and $t_{i,2}$ can be obtained via SPICE simulation.

For a chosen path k, the path delay D_k is the sum of all circuit elements' delays on this path, i.e.,

$$\begin{aligned} D_k &= \sum_{\forall(i,n)\in p_k} d_{i,n} = D_{k,0} + \sum_{\forall(i,n)\in p_k} t_{i,1} \cdot \Delta L_{i,n} \\ &\quad + \sum_{\forall(i,n)\in p_k} t_{i,2} \cdot \Delta V_{i,n} \end{aligned} \tag{6}$$

where p_k is the set of circuit elements on the path.

In deterministic case, the path delay is a constant value and the critical path is unique. In the presence of process variations, however, each path delay is a random variable and the critical path is not unique as different paths may be frequency-limiting at different process space with certain probability. Therefore, the chip-level statistical critical path delay should be computed as the statistical maximum of all path delays. For a chosen set of near critical paths from the trace, we compute the statistical critical path delay as

$$D_{chip} = \max_k(D_k). \tag{7}$$

Because the max function in (7) is a non-differentiable function, no closed form formula is known to compute the distribution of D_{chip} exactly. To overcome the difficulty in evaluating the distribution of D_{chip}, we resort to the technique used in [8], which approximates the statistical maximum of a set of normal distributions as another normal distribution. Details of the derivation is presented in [9].

We verify our timing model for critical path delays by comparing the critical path timing distributions obtained by our technique, Monte Carlo simulation, and [5] where neither spatial correlation nor path-convergence induced correlation is considered. We define the *distribution error* as the integration of the absolute error between two distributions. Among all benchmarks we have tested, we find that the distribution error between [5] and Monte Carlo simulation is about 29%, while the error between our model and Monte Carlo simulation is about 15%. In another word, we improve the critical path delay estimation by 14%. This convincingly shows the importance of considering spatial correlation and path convergence for accurate timing evaluation in the presence of process variations.

3.2. Estimation of Performance Gain

Assuming that the statistically critical paths obtained from the trace can be placed anywhere on the chip, we estimate the achievable performance gain through chipwise placement by computing the statistical difference between the minimum achievable circuit delay and the maximum possible circuit delay, i.e.,

$$D_{gain} = D_{max} - D_{min}, \quad (8)$$

$$D_{max} = \max_k(D_{chip,k}), \quad (9)$$

$$D_{min} = \min_k(D_{chip,k}), \quad (10)$$

where $D_{chip,k}$ is the statistical critical path delay for the k^{th} instance of the same application. The delay difference D_{gain} gives us an indication of the potential delay reduction we can obtain by chipwise placement of each FPGA chip in the presence of process variations. Because all path delays are modeled as random variables, the delay reduction is also represented as a random variable. We can evaluate the distribution of D_{max}, D_{min}, and their correlation by using similar techniques as discussed in [9]. Therefore, we can obtain the distribution of D_{gain} by computing the statistical difference between D_{max} and D_{min}. The average potential delay reduction μ_{gain} is the mean of D_{gain}.

We study the average potential delay reduction under different design settings. We examine FPGA designs with different sizes in terms of CLB numbers, i.e., Small (30×30), Medium (45×45) and Large (60×60). We categorize the benchmarks into two application types: i.e., long critical path L_p and short critical path S_p. Three variation amounts are studied, i.e., the 3-sigma variation is $3\sigma_X$=10% (L_v), $3\sigma_X$=15% (M_v), and $3\sigma_X$=20% (H_v) of the nominal values, respectively. The correlation distance $\overline{v}$ is set as short range ($1mm$), medium range ($2mm$), and long range ($3mm$), respectively. The variation ratio between inter-chip global variation, intra-chip spatial variation, and uncorrelated random variation is set as $\sigma_g : \sigma_s : \sigma_r$ = 1:1:1.

We report the relative delay reduction in percentage with respect to the nominal chip delay (μ_{gain}/nominal value) in Table 1. From Table 1, we can see the the average potential delay reduction via chipwise placement ranges from 2.4% to 14%. When process variations increase, the potential delay reduction also increases, and the reduction is always higher for short critical path applications than for long critical path applications. We also observe from Table 1 that different spatial correlation distances result in different delay reductions, and it is always better to perform chipwise placement under the long range spatial correlation distance than that of the short range correlation distance, and the relative gain for short critical path chips under high variation is up to 14%.

CorrDist		Short Range			Medium Range			Long Range		
Chip size		S	M	L	S	M	L	S	M	L
L_v	L_p	2.4	2.4	2.4	2.8	2.9	2.9	3.0	3.3	3.4
	S_p	5.7	5.7	5.7	6.9	7.0	7.0	7.3	7.6	7.6
M_v	L_p	3.5	3.6	3.6	4.1	4.3	4.4	4.5	4.9	5.0
	S_p	8.2	8.2	8.2	10	10	10	11	11	11
H_v	L_p	4.7	4.8	4.8	5.5	5.7	5.8	6.0	6.6	6.6
	S_p	11	11	11	13	13	13	14	14	14

Table 1. Delay reduction in percentage

4. FPGA CHIPWISE PLACEMENT

4.1. Algorithm

Encouraged by the high potential gain as shown in previous section, we proceed to develop a detailed placement algorithm to optimize FPGA designs for performance by leveraging the presence of correlated process variation. The algorithm is also denoted as $vaPL$.

As a proof of concept, our $vaPL$ algorithm is implemented inside the VPR framework [7], and we modify the T-Vplace provided by VPR, which is a deterministic placement engine without considering process variation. The original T-Vplace is based on simulated annealing (a general iterative optimization framework), and so is our current $vaPL$ implementation. But it is understood that the same concept can be easily extended to other deterministic placement algorithm as well. Note that $vaPL$ is deterministic to each FPGA chip once the chip's variation map is known. But $vaPL$ may still lead to different placements for different chips of the same application, as each chip's variation map may be different.

In the original T-Vplace, the timing cost function is an estimation of the critical path delay, which is computed as the sum of deterministic delays of circuit elements along the critical path. In contrast, the critical path delay in our $vaPL$ is computed according to the chip's variation map, i.e.,

$$D_{var} = \sum_{(i,n)\in p_{crit}} (d_{i,0} + t_{i,1} \cdot \Delta l_{i,n} + t_{i,2} \cdot \Delta v_{i,n}), \quad (11)$$

where p_{crit} is the set of circuit elements on the critical path, and $\Delta l_{i,n}$ and $\Delta v_{i,n}$ are the actual change of effective channel length L_{eff} and threshold voltage V_{th} to the i^{th} circuit element at location n, respectively, according to the given variation map. In another word, given the variation map, $\Delta l_{i,n}$ and $\Delta v_{i,n}$ are no longer two random variables, but two deterministic sample instances of their respective random distributions ($\Delta L_{i,n}$ and $\Delta V_{i,n}$) as given in (5).

After placement, we finish the design by using the routing algorithm provided by VPR. Finally, a detailed static timing analysis (STA) is performed to obtain the exact critical path delay of the chip. Similarly, the STA need to compute the critical path delay according to the chip's variation map, therefore, we denote such an STA run as $vaSTA$.

4.2. Experimental Result

Twelve MCNC benchmarks are used for our experiments. We set the total variation ($3\sigma_X$) for each circuit element as 10% of their respective nominal value. The ratio between inter-chip global variation, intra-chip spatial variation, and uncorrelated random variation (σ_g:σ_s:σ_r) is set as 1:1:1, and the spatial correlation distance $\overline{v}$ is set as $2mm$.

For each benchmark to be implemented on N number of FPGA chips, we conduct two types of experiments. The first experiment is based on the existing deterministic placement practice, and we denoted it as $dtPL$. Specifically, we perform the deterministic $dtPL$ algorithm to obtain the best possible design for one FPGA chip according to the nominal delay value, and that placement is then applied to all chips for the same design. The second experiment is based on our proposed variation-aware chipwise placement $vaPL$. Specifically, we perform chipwise $vaPL$ algorithm to obtain the best possible design for each individual FPGA chip according to each chip's variation map.

We compare the circuit performance between $vaPL$ and $dtPL$ for all benchmarks in Table 2. The chip size for each benchmark is decided so that the utilization rate [1] is about 90%. For the chips obtained from $dtPL$, the 3-sigma timing, according to the existing practice for FPGA designs without considering process variation, is obtained by taking the worst-case delay for all circuit elements on the critical path. Results from this approach is reported under column 3 in Table 2. This approach is apparently too pessimistic, as it is very unlikely for all circuit elements on the critical paths to have the worst-case delay at the same time. To reduce pessimism and consider correlated process variation, we can use the true 3-sigma timing as a measure of chip performance. To do this, we first run $vaSTA$ for each chip to obtain its exact timing according to the variation map. Such exact timing is not known to designers. Therefore, we take the maximum of all exact timings obtained from all tested variation maps [2] for the same application as an approximation of the true 3-sigma timing. The approximated true 3-sigma timing for chips from both $dtPL$ and $vaPL$ are reported under columns 4 and 5 in Table 2.

Comparing columns 3 and 4 in Table 2, we can see that the worst-case timing is indeed too pessimistic compared to the true 3-sigma one, and the relative pessimism reduction by using the true 3-sigma timing is about 49.5% on average. Comparing columns 4 and 5, we find that placement results from $vaPL$ are always better than those from $dtPL$. The performance improvement for $vaPL$ is up to 11.6%, or 5.3% on average.

We also show the delay improvement of $vaPL$ for all tested chips. The actual timing for chips from both $dtPL$ and $vaPL$ is determined by performing $vaSTA$ according to the same variation map. The performance differences between $dtPL$ and $vaPL$ are then collected. For each benchmark, the performance improvement for all chips forms a histogram. Four such histogram are plotted in Fig. 2. The x-axis is the relative performance improvement for $vaPL$ over $dtPL$; and the y-axis is the percentage of chips that can achieve such improvement among all N chips implemented. For example, for the benchmark *diffeq*, there are about 30% of test chips that can achieve 7% performance improvement by performing chipwise placement through $vaPL$ when compared to $dtPL$.

The achievable performance improvement for $vaPL$ is also reported in Table 3, including the maximum and average improvement obtained from detailed placement, and estimated performance improvement using trace-based model in Section 3. For the same four benchmarks as shown in Fig. 2, *diffeq* and *spla* have shorter critical paths, while the other two have longer critical paths. *clma* uses medium chip size (45×45) and the other three use small chip size (30×30). We observe that the average performance improvement are always higher for the short critical path applications when compared to the long critical path applications. This observation agrees with what we have seen in section 3.2 via the estimation method. Column 5 of Table 3 shows the estimated average delay improve using the method in

1	2	3	4	5
Bench-mark	Chip size	$dtPL$ (ns)		$vaPL$ (ns)
		WC	3-sigma	3-sigma
alu4	13 × 13	35.8	19.0 (-46.9%)	18.4 (-3.2%)
apex2	15 × 15	43.8	24.3 (-44.5%)	21.9 (-9.9%)
apex4	12 × 12	35.3	19.7 (-44.3%)	17.4 (-11.6%)
clma	37 × 37	79.0	41.4 (-47.6%)	39.7 (-4.1%)
diffeq	14 × 14	54.3	24.5 (-55.0%)	23.6 (-3.7%)
elliptic	21 × 21	67.5	34.5 (-48.8%)	32.9 (-4.6%)
ex5p	12 × 12	37.4	20.8 (-44.4%)	19.9 (-4.3%)
misex3	13 × 13	35.9	19.8 (-44.9%)	19.4 (-2.0%)
s298	16 × 16	80.5	41.3 (-48.7%)	39.5 (-4.4%)
s38584.1	27 × 27	41.3	21.6 (-47.8%)	20.6 (-4.6%)
seq	15 × 15	33.1	18.5 (-44.3%)	18.0 (-2.7%)
spla	20 × 20	50.0	28.4 (-43.3%)	26.0 (-8.5%)
average	-	49.5	26.2 (-46.7%)	24.8 (-5.3%)

Table 2. Comparison between $vaPL$ and $dtPL$.

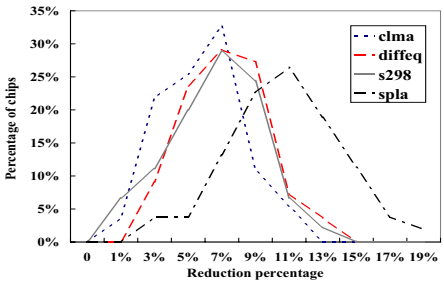

Fig. 2. Performance improvement histogram.

[1] In this paper, utilization rate is defined as the utilization rate of logic blocks, i.e., the number of used logic blocks over the total number of available logic blocks in the FPGA chip.

[2] We test 60 different variation maps for each benchmark in this paper

Appl type	Benchmark	Max	Average	Estimated
Long Critical Path	clma	11.50%	6.91%	2.9%
	s298	14.80%	7.32%	2.8%
Short Critical Path	diffeq	13.20%	7.89%	6.9%
	spla	19.30%	12.10%	6.9%

Table 3. Performance improvement.

Utilization rate	Max improve	Average improve
60%	23.22%	15.62%
70%	21.65%	14.80%
80%	20.38%	13.02%
90%	20.01%	12.11%
99%	19.30%	12.10%

Table 4. Comparison between different utilization rates.

section 3.2. We notice that the actual delay improve achieved by chipwise placement is higher than the estimated value, which is the lower bound of performance gain due to the fact that the estimation fixes the layout of the critical path. This is expected, as the assumption of fixed critical path layout is lifted in our detailed placement implementation for $vaPL$.

We further studied the impact of utilization rate on performance improvement. We have tried different utilization rates, ranging from 60% to 99%. Table 4 shows the experimental results for the benchmark *spla*. According to Table 4, we can see when the utilization rate decreases, the performance improvement by chipwise placement becomes larger. This observation is not surprising, because when the utilization rate is low, it leaves more room for improvement for chipwise placement. This also convincingly shows that our chipwise placement technique is especially valuable for FPGA because the typical utilization rate is 62.5% [10].

5. DISCUSSION AND CONCLUSION

In this paper, we have proposed a design flow for FPGA to leverage process variations by utilizing the programmability of FPGA for performance optimization. For a given set of FPGA chips, we first generate the variation map for each chip. Test circuits may be synthesized for each chip to obtain the variation map. Based on the variation map, we estimate the potential delay improvement of chipwise placement. If the improvement is large, it is worthwhile to perform placement for each chip; otherwise, we just use the conventional design flow, which uses the same placement and route for all chips.

There are two key components developed for the implementation of the chipwise placement flow. First, we have developed an accurate and efficient estimation method to quantitatively assess the potential performance gain for FPGAs by chipwise placement without detailed placement. Such estimation provides a lower bound of the performance gain achievable by variation aware chipwise placement. Second, we have studied the FPGA performance optimization problem by chipwise placement for each FPGA chip. Experimental results have shown that our proposed chipwise placement technique improves FPGA performance by up to 19.3%.

Experimental results from this work warrant further studies on design optimization for FPGAs in the presence of process variations. In the future, we plan to explore the combined performance and power optimization by chipwise physical synthesis (technology mapping, clustering, placement and routing) of FPGAs.

Moreover, our chipwise placement can also be combined with speed binning. We speculate that the delay improvement achievable by chipwise placement for the chips in the same bin would be similar. If so, the same placement could be applied for all chips in one speed bin to reduce the design time of chipwise physical synthesis.

6. ACKNOWLEDGMENT

The authors thank Dr. Arif Rahman from Xilinx for discussion on variation map generation.

7. REFERENCES

[1] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," in *Proc. Int. Conf. on Computer Aided Design*, Nov. 2003, pp. 621 – 625.

[2] C. Visweswariah, K. Ravindran, K. Kalafala, S. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," in *Proc. Design Automation Conf*, June 2004.

[3] M. Mani, A. Devgan, and M. Orshansky, "An efficient algorithm for statistical minimization of total power under timing yield constraints," in *Proc. Design Automation Conf*, June 2005, pp. 309–314.

[4] M. R. Guthaus, N. Venkateswaran, C. Visweswariah, and V. Zolotov, "Gate sizing using incremental parameterized statistical timing analysis," in *Proc. Int. Conf. on Computer Aided Design*, Nov 2005.

[5] H.-Y. Wong, L. Cheng, Y. Lin, and L. He, "FPGA device and architecture evaluation considering process variations," in *Proc. Int. Conf. on Computer Aided Design*, Nov 2005.

[6] Y. Lin, M. Hutton, and L. He, "Placement and timing for fpgas considering variations," August 2006.

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

[8] C. Clark, "The greatest of a finite set of random variables," in *Operations Research*, 1961, pp. 85 – 91.

[9] "Technical report," in *//http:eda.ee.ucla.edu*.

[10] B. L. T. Tuan, "Leakage power analysis of a 90nm fpga," in *Proc. of Custom Integrated Circuits Conference*, Sept. 2003, pp. 57–60.