Variation-Aware Routing for FPGAs

Satish Sivaswamy
Dept. of Electrical Engineering
University of Minnesota, MN 55455
satish@umn.edu

Kia Bazargan
Dept. of Electrical Engineering
University of Minnesota, MN 55455
kia@umn.edu

ABSTRACT

Chip design in the nanometer regime is becoming increasingly difficult due to process variations. ASIC designers have adopted statistical optimization techniques to mitigate the effects of variations. The FPGA community on the other hand, has only recently started focussing on the effects of variations. This paper presents a comparative study of the impact of variations on designs mapped to FPGAs and ASICs to get a measure of the severity of the problem in both the FPGA and ASIC domains. We also propose a variation aware router that reduces the yield loss by 7.61X, or the circuit delay by 3.95% for the same yield for the MCNC benchmarks.

1. INTRODUCTION

As feature sizes scale down to the sub 90 nanometer regime, the impact of process variations on the performance of digital circuits is fast becoming critical to account for in the design process. Failure to do so often results in large numbers of chips failing the timing/power requirements. In ASICs, this impact has been researched extensively and several techniques have been proposed to optimize the performance and power in the presence of process variations [15],[16],[17]. Until recently, the impact of process variations on FPGAs were largely ignored since designs mapped on FPGAs were typically much slower than their ASIC counterparts and at low clock frequencies, process variations do not have a significant impact. However, as we continue scaling down feature sizes, FPGAs are becoming faster and the impact of process variations can no longer be ignored. Recently there have been a few studies that consider the impact of process variations on FPGAs. In [1], the authors performed an in-depth architecture evaluation and studied the impact of parameter variations on the timing and leakage yield. [2] proposes a variation aware placement algorithm to improve the timing yield of FPGAs. The authors also study the impact of speed binning and guard-banding the circuit elements on the timing yield and the margin added to the critical path delay.

In the first part of the paper, we make a comparative study of designs mapped tpically on FPGAs and ASICs to gauge the difference between the impact of process variations on these two implementation platforms. Our motivation in doing this study is to stimulate extensive discussion about whether statistical optimization techniques are indeed as critical to FPGAs as they are for ASICs. If they have a lesser impact on FPGAs, the FPGA community would have some more time to study different statistical analysis and optimization techniques before fully adopting them in the design flow. It is our belief that statistical optimization techniques are more useful in the later stages of the design flow since we would have more complete and accurate information later in the design flow. As a result, the routing stage will have a significant impact on the performance of the final design since interconnect delays dominate the delay of logic blocks. In the second part of the paper, we propose a variation aware router to improve the timing yield in the presence of process variations. Experimental results show that our variation aware router improves the timing yield by 7.61X and, if we keep the yield loss constant, our router improves the circuit delay by 3.95%. The rest of the paper is organized as follows. Section 2 presents some background knowledge and gives the modeling details. Section 3 compares the impact of process variations on ASICs and FPGAs. Section 4 proposes a variation aware router and Section 4.2 presents some results and discussion. Section 5 concludes the paper.

2. MODELING VARIATIONS

In this section, we outline the model used for representing parameter variations. In this work, we modeled all variations as spatially correlated gaussian distributions. We considered variations in the transistor length (L_{eff}), transistor width (W_{eff}), metal thickness (T_{int}) and metal width (W_{int}). We used a single layer predictive technology model[7] for the interconnects. The framework that we present here can be used for an arbitrary number of metal layers and can handle more device parameters with only minor modifications. Due to lack of access to actual foundry data, we use the process parameters from [3]. They are shown in Table1.

In the table, μ refers to the mean and σ refers to the standard deviation of the process parameters. The inter and intra subscripts refer to the inter- and intra-die component of the variations. As in [3], we model the process variation as

$$X_{total} = X_{inter} + X_{intra} \tag{1}$$

Table 1: Process Parameters

Parameters	L_{eff} (nm)	W_{eff} (nm)	W_{int} (nm)	T_{int} (nm)
μ	60.0	150.0	150.0	500.0
$3\sigma_{inter}$	9.0	11.25	15.0	25.0
$3\sigma_{intra}$	4.5	5.625	7.5	12.5

where X_{inter} is the inter-die variation and X_{intra} is the intra-die variation. X_{inter} and X_{intra} are modeled as gaussian distributions. In our work, X_{total} refers to the total variation in any of the process parameters, L_{eff}, W_{eff}, T_{int} or W_{int}. The inter-die component of the variation affects all the components of a chip identically. The intra-die component however, depends on the location of the device(wire) on the chip and is subject to spatial correlation within the chip. Devices that are close to each other tend to have similar characteristics and to capture this effect, we adopt the approach used in [4].

2.1 Spatial Correlation

In [2], the authors do not consider the effect of spatial correlation among different circuit components and this may result in inaccuracies in delay estimation. This is illustrated in Figure 1. In the figure, the chip is divided into 25 grids.

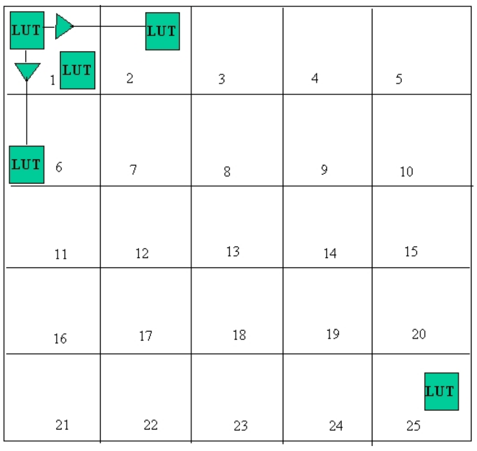


Figure 1: Spatial Correlation

When the effect of spatial correlation is ignored, the characteristics of the devices in each grid are assumed to be independent of the devices in other grids. Due to the nature of the fabrication processes involved, this is not the case. Devices that are located close to each other typically exhibit similar characteristics[1]. The two look-up tables (LUTs) in the grid 1 will exhibit almost identical characteristics and the LUTs in the grids 2 and 6 will have a strong correlation with those in grid 1. This correlation will decrease as we increase the distance of separation and the LUT in grid 25 will exhibit very little correlation with grid 1. This is true for routing resources as well. Hence, we cannot consider all devices to be independent of each other when performing timing analysis. The authors in [4] performed extensive critical dimension measurements on fabricated dies and proposed a simple piecewise linear model to capture the effect of spatial correlation. The model is illustrated in Figure 2. The X-axis shows the distance between two devices(wires), d, and the Y-axis shows the spatial correlation coefficient, ρ, between intra-die variation parameters at different locations on the chip. In the model, ρ_B denotes the baseline correlation coefficient and D_L denotes the characteristic correlation length. From Figure 2, we see that the spatial correlation between process parameters at different locations on the chip decreases linearly with the distance of separation and then remains constant (ρ_B) beyond a critical separation distance, which is typically half the size of the chip[4]. Formally, the model is expressed as

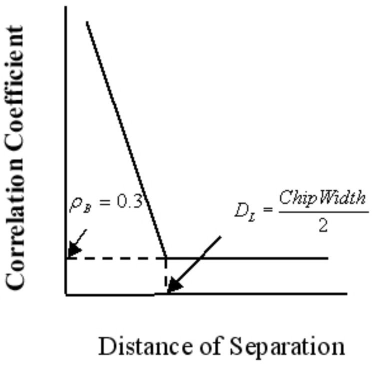


Figure 2: Spatial Correlation Model

$$\rho = \begin{cases} 1 - \frac{d}{D_L}(1-\rho_B) & d \le D_L \\ \rho_B, & d \ge D_L. \end{cases} \quad (2)$$

We divide the chip into several grids and assume that within a grid, all devices have identical process parameters, L_{eff}, W_{eff}, L_{int} and W_{int}. Further, as in [3], we make another assumption that the different types of parameters (L_{eff}, W_{eff}, L_{int} and W_{int}) are independent of each other. The inter-die component of variation is added to the entries of the covariance matrix of the spatially correlated variables corresponding to the intra-die component of variation. We then use the principal component analysis technique(PCA) [6] to convert these correlated random variables into a set of mutually independent variables such that each new random variable produced using PCA has a standard normal distribution,(N(0,1)).

2.2 Delay Modeling

As in [3], we express the delay of a circuit element using a linear model expressed as:

$$d = d_0 + \sum_{\forall p_i} S_{i_0} \Delta p_i \quad (3)$$

where d_0 is the nominal delay of the circuit element and S_i's are the sensitivities of the delay to the various process parameters, p_i's. These sensitivities are computed around the nominal process condition from Hspice simulations. In Equation 3, the p_i's are correlated and they are in-turn expressed as a weighted linear sum of the corresponding prin-

[1]t_{ox} and Dopant concentration however, are truly random and exhibit no spatial correlations

cipal components. The expression for delay becomes:

$$d = d_0 + \sum_{\forall p_i} S_{i_0} \sum_{j=1}^{m} a_{ij} p'_j \qquad (4)$$

where m is the number of grids into which the chip is divided and p'_j's are mutually independent N(0,1) variables and a_{ij}'s are their coefficients from PCA. It is to be noted here that the delay variable, d, is a sum of weighted gaussians and is hence itself a gaussian. We adopt the same method as in [3] to compute the variances and covariances of the delay variables. In statistical static timing analysis (SSTA), finding the greatest of two delay variables is accomplished by doing a statistical max operation which approximates the maximum of two gaussian distributions as another gaussian [5]. The addition operation in static timing analysis (STA) is replaced in SSTA by a statistical sum operation which adds up the corresponding coefficients of two delay variables to produce a third delay variable in the canonical form(shown in Equation 4). With these two statistical operators (max and sum) at hand, the delay distribution at the output can be computed by traversing the timing graph as we would do in STA.

3. IMPACT OF PROCESS VARIATIONS: FPGAS VS ASICS

Designers in the ASIC community have been worrying about the effects of process variations for a long time and are making a transition from a corner-based design approach[2] to using statistical optimization techniques. Nevertheless, researchers in the FPGA community have only recently started considering the impact of variations. In this section, we compare the impact of process variation on FPGAs and ASICs to understand whether we have reached the stage, where, statistical analysis is as critical for FPGAs as it is for ASICs. The objective of this study is to find out if FPGAs and ASICs behave differently when subjected to variations. We do not perform any statistical optimization for FPGAs or ASICs in this section since we are interested in observing the impact of ignoring the effect of variations on the accuracy of delay prediction and hence its impact on the design flow.

In this study, we use a 65nm standard cell library from the Open Access tool kit [9] and a 65nm CMOS SRAM based FPGA. We had to rely on tools that were available to the academia since we did not have access to industry-grade CAD tools. We have tried our best to optimize both the FPGA and the ASIC flows and will present both flows used in this study. Another important aspect of our study is the selection of benchmarks. We chose to use the MCNC circuits for our FPGA flow and ISCAS89 circuits for our ASIC flow. One might argue that the benchmarks we use must be identical for both the FPGA and the ASIC flows. That would be true if we wanted to compare FPGAs and ASICs as implementation platforms to find out which platform is more suitable for a particular application. That however, is not of interest to us. We want to find out the impact of process variations on FPGAs and ASICs and we believe that in order to accomplish that, we need to look at the impact on typical designs implemented with FPGAs and ASICs. The MCNC benchmarks are widely used in academia for FPGAs but may not be representative of the issues that ASIC designers face. The same is true with the ISCAS benchmarks and FPGA designs.

[2]The design is optimized with multiple process-corners and is generally pessimistic and time consuming.

3.1 FPGA Design Flow

In this work we use an island style FPGA architecture with a cluster size of 1 and LUTs having 4 inputs. The routing architecture consists of 10% length-1, 25% length-2 and 65% length-6 wires with buffered switches. We used a subset switch-box topology. MCNC benchmarks are mapped to the architecture using T-Vpack[13] and are then placed and routed using VPR operated in the timing-driven mode[8]. Once we are done with the placement and routing of the design, we use our statistical static timing analyzer to analyze the delay distributions at various nodes in the circuit. The flow we use is shown in Figure 3(a).

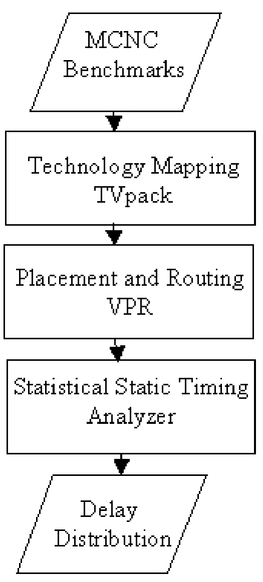
3.2 ASIC Design Flow

A standard-cell based design flow is more complex than the FPGA flow presented in section 3.1. This is primarily because of the need to use a large number of CAD tools to complete the physical design, which leads to a huge design space to explore for optimization. As we pointed out earlier, both the library and the CAD tools we used for our work are available freely to researchers. We have made our best effort to optimize the designs and we believe that the results that we obtained give us an insight into how a typical standard-cell based design behaves when subjected to process variations.

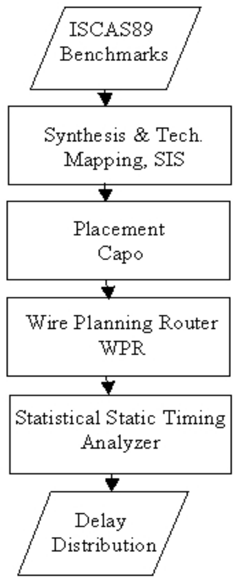
The ISCAS89 benchmarks are first synthesized and mapped to the standard-cell library using SIS[11]. We use the script *script.rugged* to perform technology independent optimization. After we have the mapped netlist, we use Capo[10] in the timing-driven mode to place the design. We then use the router in [12] to perform the routing. We use a 2 metal-layer interconnect model similar to [3]. We then use the statistical static timing analyzer in [3] to analyze the delay distributions. The ASIC flow is shown in Figure 3(b).

3.3 Comparative Studies

One of the main reasons statistical optimization techniques are being used is the inability to accurately predict paths in the circuit may lead to timing violations in the presence of variations. At the granularity of logic blocks, this translates to the inability to predict the latest arriving input that determines the arrival time at the output. Since the arrival times are random variables with a certain mean (μ) and standard deviation(σ), estimating the output arrival time in a purely deterministic manner (by just looking at the means) would lead to erroneous results if the means of the arrival times at the inputs are sufficiently close. On the other hand, if the $\mu + 3\sigma$ point of the delay of one input is lower than the $\mu - 3\sigma$ point of the delay of another input, there is no need to use the statistical max operation. We can estimate the maximum delay simply as the delay of the second input. We call this the *pruning* of the max operation. This is illustrated in Figure 4. The figure shows the arrival-time distributions at two inputs of a logic block. Input B completely dominates input A. The arrival-time distribution at the output is thus determined only by B and there is no contribution from A. We can determine the arrival time distribution at the output by performing a sum operation of


MCNC Benchmarks
Technology Mapping TVpack
Placement and Routing VPR
Statistical Static Timing Analyzer
Delay Distribution

(a) FPGA Design Flow



(b) ASIC Design Flow

Figure 3: Design Flows

the arrival time at input B and the delay of the logic block itself.

In the first part of this section we want to observe the number of times this happens on typical FPGA designs and compare it with standard-cell based designs. The motivation behind this study is to see whether the situation illustrated in Figure 4 happens more often in FPGAs than in ASICs. If this was the case, then, even with variations, we would be able to predict what input determines the signal arrival time at the output of logic blocks better than in ASICs. Extending this to paths, we can predict critical path(s) of FPGA circuits even in the presence of variations and can do a better job at optimizing the circuits. It would further mean that at the technology node considered in the study (65nm), process variations would be more detrimental to ASICs than FPGAs due to the uncertainty involved with different paths being critical for different chips. To answer these questions, we pass the benchmarks through the flows shown in Figures 3.1 and 3.2 and gather statistics on the number of statistical max operations that were performed (ignoring the pruned out cases) and the total number of max operations attempted. The results are shown in Table 2[3].

In Table 2, columns 2 and 6 give the number of statistial max operations performed in the FPGA and ASIC circuits listed. Columns 3 and 7 give the total number of max operations including the cases that were pruned out. We can see that on an average, the statistical max operation is performed only about 40% of the time for FPGA designs. This number is however, about 57% for the ASIC circuits and it is about 60% for the larger circuits. This indicates that in FPGA circuits, for the most part (60% of the time), we can deterministically predict inputs that are responsible for setting the arrival times at outputs of logic blocks.

It is to be noted that a closed-form solution exists only

[3]We did the same studies with the wilton and Imran switchboxes and the results were similar

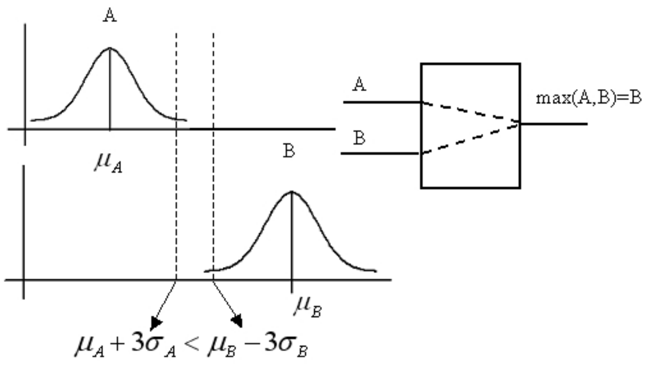


Figure 4: Finding Max delay of 2 Paths

for the maximum of two random variables and as a result, we would have to make pair-wise comparisons if the number of inputs to a logic block is greater than 2. As a result, the order in which inputs are considered is important as it determines the number of maxes performed as well as the accuracy of the estimation of the output distribution. For instance, consider the situation illustrated in Figure 5. The LUT that we are interested in has 4 inputs (A,B,C,D),

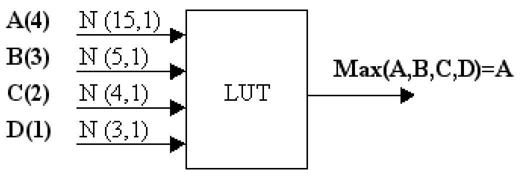


Figure 5: Finding Max delay of 2 Paths

and the order in which they are considered are indicated in

FPGA				**ASIC**			
Circuit	**# Maxes Perf.**	**Tot. # Maxes**	**% Maxes**	**Circuit**	**#Maxes Perf.**	**Tot. # Maxes**	**% Maxes**
ex5p	1646	4073	40.41	s5378	969	1651	58.70
alu4	2176	5430	40.07	s9234	826	1569	52.65
ex1010	6545	16098	40.65	s13207	3100	5153	60.16
seq	2361	6269	37.66	s15850	3376	6693	50.44
des	2355	6611	35.62	s38417	8693	15007	57.93
spla	6696	13870	48.27	s38584	13902	22659	61.35
pdc	6418	17249	37.20	s1196	465	909	51.11
apex2	2525	6733	37.50	s35932	9036	14251	63.40
misex3	1943	4996	38.89				
apex4	2066	4508	45.82				
Average			**40.21**				**56.97**

Table 2: Comparison of Maxes performed

parentheses. They are all normal distributions with their means and standard-deviations marked on the inputs to the LUT. With this particular order in which inputs are considered, we would perform 2 max operations {max(D,C), max(B,max(D,C))}. We would not perform a max operation with A since it would dominate the other variable as shown in Figure 4. If on the other hand, we considered input A first, then we would not have performed any max operations at all for this logic block since $\mu_A - 3\sigma_A$ is greater than the $\mu + 3\sigma$ values of the other inputs. As a direct consequence of this limitation, the average number of maxes performed as reported in Table 2 is a pessimistic estimate. This problem is more pronounced for the FPGA architecture than the standard-cell designs we considered, since the standard-cell based designs had a large number of two-input gates and the order in which the max operation was performed did not matter for those gates. We were more interested in knowing how FPGAs are affected by process variations, so we performed more studies to gather information about the tightness probabilities of the inputs of logic blocks in FPGAs. Tightness probability was defined in [14]. If we have two random variables A (N(μ_A,σ_A)) and B(N(μ_B,σ_B)), the tigtness probablity of A (T_A) is the probability that A>B. T_A can be computed from the following analytic expressions[14]:

$$\phi(x) \equiv \frac{1}{\sqrt{2\pi}} exp(-\frac{x^2}{2}) \tag{5}$$

$$\Phi(y) \equiv \int_{-\infty}^{y} \phi(x)dx \tag{6}$$

$$\theta \equiv ({\sigma_A}^2 + {\sigma_B}^2 - 2\rho\sigma_A\sigma_B)^{1/2} \tag{7}$$

$$\begin{aligned} T_A &= \int_{-\infty}^{\infty} \frac{1}{\sigma_A}\phi\Big(\frac{x-\mu_A}{\sigma_A}\Big)\Phi\left(\frac{\big(\frac{x-\mu_B}{\sigma_B}\big) - \rho\big(\frac{x-\mu_B}{\sigma_B}\big)}{\sqrt{1-\rho^2}}\right)dx \\ &= \Phi\big(\frac{\mu_A - \mu_B}{\theta}\big) \end{aligned} \tag{8}$$

Given a large number of random variables, each has a tightness probability corresponding to the probability that it is larger than the rest of the variables. We can get a better insight at the contribution of different LUT inputs to the output arrival times when we look at the tightness probabilities of the inputs. The statistics we gathered on the tightness probabilities are presented in Table 3.

We computed tightness probabilities (TPs) of all inputs to a LUT and defined the dominant input to be the one with the maximum TP. Table 3 shows the histogram of the dominant input TPs. For example, in ex5p, 79.2% of all the LUTs have an input with a TP of 1,i.e., no statistical max is needed to find its output delay distribution. For the same circuit, in 9.2% of the LUTs, the dominant inputs has a TP between 0.95 and 1. As we expected, the results in Table 3 indicate that some of the max operations performed in the circuits were redundant and were done simply because of the sub-optimal order in which the maxes were performed. We observe that on an average, about 76% of all LUTs had a single input which determined the output arrival time and about 90% of the LUTs had a single input determining the output arrival time 90% of the time i.e., the TP of the dominant input is in the range [1,0.9].

We believe that the reason for a single input dominating all other inputs to a logic block in FPGAs for the most part can be attributed to the constraints in the routing architecture. Though the depth of logic on different input to output paths in designs implemented on both FPGAs and ASICs are balanced, the routing paths may not be balanced. The restricted routing architecture in FPGAs can cause the router to take more detours than in ASICs. As a result of the limited routing flexibility in FPGAs due to the pre-fabricated routing structures, we are more likely to have balanced routing paths in addition to a balanced circuit topology for ASICs. Another reason this might be happening can be due to the fact that the connections in FPGAs pass through a lot of programmable elements and hence is much slower than the routes in ASICs. Variations play a more critical part when the fraction of the delay change due to variations is significant when compared to the delay at nominal process conditions. In ASICs, this ratio is likely to be more since they are inherently much faster than the FPGAs and consequently, the change would be more significant in ASICs than in FPGAs.

This means that the critical paths through a majority of the blocks are unaffected by process variations. Nevertheless, the table provides information about only individual blocks and not paths in the circuit. Analyzing different paths in the circuit would provide more information about the extent to which variations affect critical and near-critical paths of circuits. This lead us to perform more studies on the criticalities of different paths in the presence of variations.

We first enumerated the top 10 critical paths of the circuits by performing a deterministic timing analysis with all

Circuit	**Perc. of Blocks with Tightness Prob. of slowest Input**						
	1	**[0.95,1)**	**[0.9,0.95)**	**[0.8,0.9)**	**[0.75,0.8)**	**[0.5,0.75)**	**<0.5**
Ex5p	79.24	9.20	2.91	2.53	1.03	4.88	0.18
alu4	71.63	11.16	3.41	3.61	2.23	7.61	0.32
misex3	61.08	17.02	4.64	5.50	2.07	8.94	0.71
apex4	78.52	9.66	2.85	2.77	0.63	5.15	0.39
apex2	72.43	12.82	2.87	4.20	1.64	5.69	0.31
ex1010	84.34	6.93	1.67	2.73	0.86	3.30	0.13
pdc	88.57	5.22	1.48	1.52	0.69	2.36	0.13
spla	72.22	13.27	3.16	3.73	1.59	5.79	0.18
seq	70.70	12.79	3.19	4.79	1.59	6.73	0.17
des	79.27	9.35	2.32	2.63	1.31	4.89	0.18
Average	**75.80**	**10.74**	**2.85**	**3.40**	**1.37**	**5.53**	**0.27**

Table 3: Tightness Probabilities

parameters set to nominal values. We then perform Monte Carlo simulation with 10,000 samples to enumerate top 10 statistically most critical paths. We compute the statistical criticality of a net as the ratio of the number of times the net was on the critical path while performing Monte Carlo simulation to the total number of trials. The statistical criticality of a node is computed similarly. Arrival tightness probabilities of all nets are then computed using the same procedure as in [14]. The criticality probability of a path is defined as the product of the tightness probabilities of all nets on a path. We then enumerate the top 10 statistically critical paths. If all paths in a circuit are considered, the critical path coverage would be 1. However, the number of critical paths could potentially be exponential in terms of the circuit size. Hence we settled for the 10 most critical paths which corresponds to a critical path coverage of 0.999 for our benchmarks. The motivation behind this study was to check whether the paths that were deemed critical by STA also had a high criticality probability under the effect of variations. If the same paths are reported as critical by both STA and Monte Carlo, then we can conclude that even in the presence of variations, STA predicts critical paths with high accuracy.

We performed these studies for both the MCNC benchmarks and the ISCAS89 standard-cell benchmarks. The results of a benchmark from MCNC and ISCAS89 benchmark suites are shown in Figures 6(a) and 6(b). The results of the other MCNC and ISCAS89 benchmarks were similar to Figure 6. In Figure 6, we show the deterministic path delays on the y-axis and the statistical path criticalities of the paths that were indentified by STA obtained from Monte Carlo simulation on the x-axis. The graphs show a marked difference in the way FPGA and Standard-cell designs are affected by variations. For the MCNC benchmark, misex3, the top 6 paths reported by STA were also among the top 6 paths reported by Monte Carlo although the third and fourth most critical paths in STA were interchanged in Monte Carlo. The top 6 paths had a total path criticality of 0.9996. This indicates that STA does a good job of predicting critical paths and we can do a good job at optimization if we consider the critical and the top few near-critical paths given by STA rather than concentrating only on the most critical path. Nevertheless, there still exists a non-zero probability that another path could be critical under the impact of variations [4]. On the other hand, when we look at the ISCAS89 benchmark, s38417, only two of the paths determined by STA to be critical had a non-zero probability of being critical with Monte Carlo. The path criticalities of those paths summed up to 0.3554. This number is much lower than the number for the FPGA design (0.9996). This indicates that for standard-cell designs, STA does a poor job of finding critical paths and that the impact of variations on standard-cell designs are much higher and hence statistical optimization techniques are more critical to standard-cell designs than FPGAs.

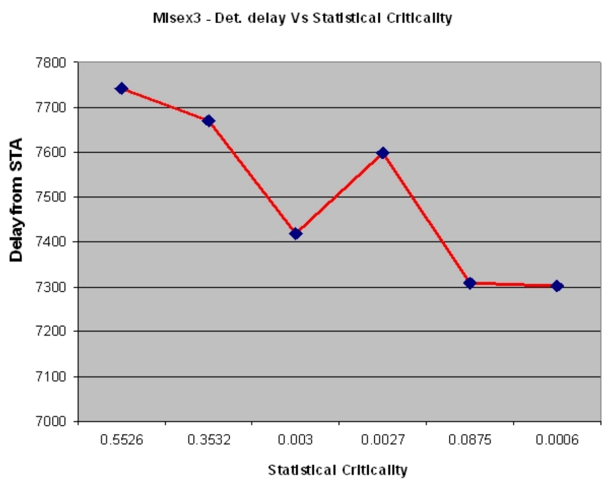
Though the discussion in this section makes a case that statistical optimization is more critical for ASICs, we do not discount the importance of using it for FPGAs. From our studies, it is clear however that we have not yet reached the stage when statistical optimization techniques are as critical for FPGA designs as they are for standard-cell designs. In [2], the authors report a yield loss of only about 7 chips out of 10,000 chips for the MCNC benchmarks with a deterministic timing analysis inside the optimization loop. This further supports our belief that we have not yet reached the stage where statistical optimization is as critical for FPGAs as they are for ASICs. Nevertheless, process variations will have a more substantial impact in future technologies and we need to explore various optimization techniques. [2] proposes a variation aware placement for FPGAs and in Section 4 we propose a variation aware router for FPGAs to maximize the timing yield.

4. VARIATION-AWARE ROUTER

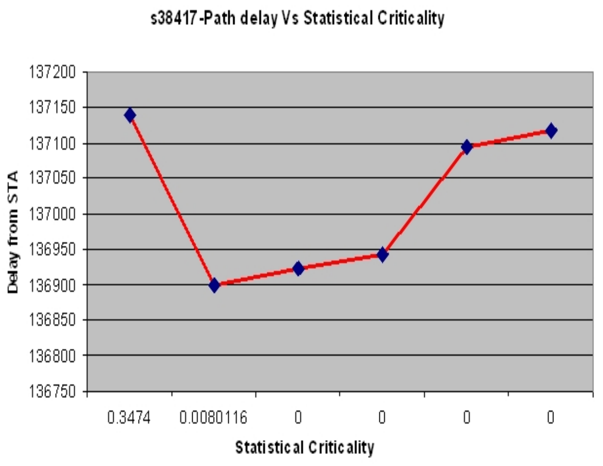
At the routing stage, we have a better estimate of the congestion of the routing resources unlike the placement stage when an optimistic delay estimate is used to estimate the criticalities of nets. Consequently, we believe that introducing statistical optimization at the routing stage would be more beneficial. In this section, we present a variation aware router to optimize timing and to maximize the timing yield. We modify the router in VPR[8] to optimize statistical criticalities of nets.

VPR uses a negotiated congestion-delay algorithm where timing critical nets are given a higher priority to use the

[4]For instance, if the circuit in Figure 6(a) has 10,000 paths, statistically there are 4 paths not identified by STA that may turn out to be critical and cause a timing violation due to variations



(a) Path Criticalities for Misex3, $\sum Path\_Criticality = 0.9996$



(b) Path Criticalities for S38417, $\sum Path\_Criticality = 0.3554$

Figure 6: Path Criticalities of FPGA Vs. Standard-Cell Designs

fastest routing resources available and less critical nets are forced to take a detour if needed, to alleviate congestion. In every routing iteration, all nets are ripped-up and re-routed and this process is repeated until there are no over-congested regions. To get the minimum delay, all the nets are marked as critical for the very first routing iteration. Once a routing iteration is complete, the net delays are known and a timing analysis is used to compute the criticalities of nets. The cost function that the router tries to optimize consists of a criticality component and a congestion component. The congestion cost of routing resources are also updated after every routing iteration. The cost of using a routing resource, n, as a part of routing a net (i,j) is given by[8]:

$$Cost(n) = Crit(i,j)delay(n) + [1-(Crit(i,j)][b(n)+h(n)]p(n) \quad (9)$$

where, *Crit(i,j)* is the criticality of the net, *delay(n)* is the delay of the routing resource *n*. The second term in the cost function captures the effect of congestion. *b(n)* is the base cost of using a resource and is set to *delay(n)* and *h(n)* is the historic congestion factor which keeps track of the congestion levels of the resource in previous iterations and *p(n)* is monitors congestion in the present iteration.

In our router, to perform timing optimization in the presence of variations, we replace the criticality computation in [8] with statistical criticality. We make no changes to the first routing iteration and set the criticality of all nets to be 1.0. After the first routing iteration, we traverse the routing trees of all the nets and compute the statistical means and standard deviations of the net delays in the form shown in Equation 3. We then perform SSTA and compute the arrival and required tightness probabilities. We use the criticality computation technique presented in [14] to compute the statistical criticalities of the nets. For subsequent routing iterations of VPR, we use statistical criticalities inside the optimization engine. The pseudo-code for our algorithm is shown in Figure 7 and is adapted from [13].

4.1 Experimental Setup

We compare the efficiency of our variation-aware router with VPR's timing-driven router for the MCNC benchmark circuits. When using the deterministic router, we adopt the standard practice of guard-banding the delay of individual circuit components with their 3σ values. The timing yield with guard-banding depends on the value of the guard-band factor and there is a trade-off between the guard-band factor chosen and the extra margin this places on the critical path delay. This trade-off has been explored in [2] and we use the 3σ value as the guard-band factor since it gives the best yield in [2]. As in [2], we define yield loss as the number of chips that fail timing specifications out of 10,000 chips. We do not modify the placement algorithm and run VPR's timing-driven placement engine to generate placements for the benchmarks. We use the same placement and routing channel-width for both VPR's router and our router. The architecture of the FPGA we used for our experiments is described in Section 3.1. We present our results in Table 4.

4.2 Results and Discussion

Table 4 presents the results of running VPR and our variation aware router on 20 MCNC benchmarks. First, we place and route circuits with all the elements guard-banded with a factor of 3σ. We then perform STA to obtain the guard-banded delay and this is shown in the column, T_{guard_band}. This value is used as the cut-off delay when we calculate the yield loss of the conventional router and our variation aware router. A SSTA is then performed on the same physical design to estimate the mean and the standard deviation of the delay of the circuit that is optimized with guard-banding. These values are reported in columns 3 and 4. We retain the placement and route the circuit with our router and estimate the mean and standard deviation of the circuit. Columns 6 and 7 present these results. The yield loss reported in columns 5 and 8 is the number of chips that fail out of 10,000 chips with the delay reported in T_{guard_band} as the timing specification. The yield loss numbers are fractional since we estimate the yield from the cumulative distribution function of the output delay. Column 9 in the table, Yield

Circuit	T_{guard_band}	VPR			Variation Aware Router			Yield	Delay	Runtime
	(ps)	Mean(ps)	Sigma(ps)	Y_{loss}	Mean(ps)	Sigma(ps)	Y_{loss}	loss Impr.	Impr.(%)	Ratio
ex5p	12144.80	9034.83	979.54	7.49	8754.44	869.81	0.33	22.70 X	6.58	1.55
alu4	12052.90	9203.27	913.50	9.06	9193.98	861.79	4.54	1.99 X	1.50	1.12
misex3	12057.00	9122.83	957.92	10.95	8728.27	879.07	0.76	14.41 X	5.76	1.42
apex2	12354.20	9668.55	851.08	8.01	9161.15	880.68	1.44	5.56 X	3.24	1.70
apex4	11259.40	8171.35	964.10	6.80	8297.57	845.53	2.30	2.96 X	2.57	1.75
pdc	20223.70	15347.10	1547.15	8.11	14596.10	1492.22	0.81	10.01 X	4.74	1.28
seq	11429.60	8991.70	763.89	7.08	8226.58	831.44	0.60	11.80 X	4.37	1.47
des	11403.54	8940.83	788.36	8.93	8107.74	898.94	1.23	7.26 X	3.73	1.59
spla	22604.60	17358.80	1690.36	9.57	16373.40	1662.72	0.89	10.75 X	4.82	1.00
ex1010	19595.50	14868.50	1354.97	2.43	14658.80	1315.03	0.87	2.79 X	1.83	1.08
frisc	16877.60	12547.50	1333.58	5.83	12283.90	1273.81	1.55	3.76 X	2.84	1.36
elliptic	14467.00	10925.50	1122.55	8.03	10867.35	1132.86	7.43	1.08 X	0.17	1.41
bigkey	6381.37	5039.69	453.72	15.53	5025.27	450.96	13.19	1.18 X	0.35	1.52
s298	18843.30	14518.60	1340.72	6.28	13371.70	1297.13	0.12	52.33 X	7.10	1.12
tseng	7797.39	5969.87	583.967	8.76	5654.55	546.49	0.44	19.91 X	5.93	1.36
diffeq	9052.21	6850.52	726.60	12.22	6752.71	648.09	1.94	6.30 X	4.16	2.33
dsip	9805.97	7857.53	640.05	11.66	7237.09	615.09	0.15	77.73 X	7.37	1.16
s38417	14437.40	11132.90	1075.88	10.65	10647.97	927.28	0.22	48.41 X	7.56	1.14
s38584.1	13387.60	10671.80	886.90	10.99	10047.16	860.87	0.52	21.13 X	5.43	1.39
clma	22793.60	16828.10	1866.62	6.97	17534.50	1722.72	11.34	0.61 X	-1.17	1.56
Mean		**10103.99**	**979.52**	**8.27**	**9731.51**	**941.98**	**1.09**	**7.61 X**	**3.95**	**1.39**

Table 4: Results of our Variation Aware Router

```
Initialization: Stat_Crit(i,j) =1.0 for all nets i and sinks j;
while (overused resources exist) do
    for (each net, i) do
        Rip-up routing tree of net i;
        Update affected components in Equation (9);
        for (each sink j, of net(i) in decreasing Stat_Crit(i,j) order) do
            Find the least cost route of for sink j;
            for (all nodes in the path from i to j) do
                Update affected Components in Equation (9);
            end
            Update Elmore delay of the route;
        end
    end
    Update Historic_Congestion();
    Compute_NetDelay_mean_and_variance();
    Compute_Arrival_Tightness_Probabilities();
    Compute_Required_Tightness_Probabilities();
    Compute_Statistical_Criticalities();
    Update_Stat_Crit(i,j) ∀ i,j;
end
```

Figure 7: Pseudo-code for Variation Aware Router

Loss Impr., gives the improvement in yield loss that can be achieved by using our router over the deterministic router.

On average, our router reduces the yield loss from about 8.27 chips out of 10,000 chips to about 1.08, which is a gain of 7.61X for the benchmarks considered. The column labeled Delay Impr. presents the improvement in circuit delay that can be obtained with our router if we maintain the yield same as that given by the deterministic router. We get an average improvement of about 3.95% in circuit delay. This is possible because our router, for the most part reduces both the mean and standard deviation of the circuit delay. This means that we can run the design with a slightly faster clock if we are willing to tolerate a few more chips failing. The last column gives the run time penalty incurred by our router. Though asymptotically SSTA has the same time complexity as STA, the constant involved is much higher. If we divide the chip into k grids, there will be k principal components for every process parameter and if we consider m process parameters there will be a total of mk coefficients in Equation 3. Thus the total time complexity is $O(mk(E+V))$, where E is the number of nets and V is the number of logic blocks. From the last column in the table, our router is 1.39X slower than VPR's timing driven router. We believe that this runtime penalty is not too high considering the improvement in yield loss.

5. CONCLUSION

In this paper, we studied the difference between the impact of process variations on designs mapped to standard-cells and FPGAs. We considered spatial correlation between process parameters of devices on different regions of the chip and adopted a simple piece-wise linear model to model its effects. We enumerated critical and near-critical paths in FPGA and standard-cell designs, first by ignoring process variations and then by considering variations. We performed

Monte Carlo simulations to obtain the criticalities of different paths in the presence of variations. This was done to compare the effectiveness of using STA to identify critical paths. We observed that STA surprisingly did a much better job of identiying critical paths in FPGAs than in standard-cell designs in the presence of variations. We also proposed a variation aware router that reduces the average loss by 7.61X when compared to the router in VPR which uses deterministic timing analysis in its optimization engine. We observed that we can also reduce the delay by about 3.95% with our router if we maintain the same yield as VPR's timing driven router.

In the future, we will try to develop heuristics that can exploit the results presented in Table 3 to speed up computation in SSTA to reduce the run time penalty. We will also explore post-layout yield optimization techniques to maximize timing yield in the presence of variations.

6. REFERENCES

[1] H.Y. Wong, L.Cheng, Y.Lin, L.Hei, "FPGA Device and Architecture Evalutaion Considering Process Variations",*International Conference on Computer-Aided Design*,2005.

[2] Y. Lin, M.Hutton, L.Hei, "Placement and Timing for FPGAs Considering Variations", *International Conference on Field Programmable Logic and Applications*, August 2006.

[3] H. Chang, S. Sapatnekar, "Statistical Timing Analysis Under Spatial Correlations", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24 No. 9, pp. 1467-1482.

[4] P. Froedberg, Y. Cao, J. Cain, R. Wang, J. Rabaey, C. Spanos, "Modeling Within-Die Spatial Correlation Effects for Process-Design Co-Optimization", *International Symposium on Quality of Electronic Design* pp. 516-521, 2005.

[5] C. Clark, "The greatest of a finite set of random variables,", Operations Research, vol. 9, pp. 85-91, 1961.

[6] I.T. Jolliffe, "Principal Component Analysis". SpringerVerlag, New York, 1986.

[7] Berkeley Device Group, "Predictive Technology model", in http://www.eas.asu.edu/ ptm

[8] V. Betz, J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research", International Conference of Field Programmable Logic and Applications, 1997.

[9] OpenAccess Gear: Library Core of EDA algorithms and infrastructure for OpenAccess, in http://www.openedatools.org/projects/oagear

[10] "Capo,
A large scale fixed-die placer from UCLA," Available at http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Placement.

[11] SIS: A System for Sequential Circuit Synthesis. Release 1.2. UC Berkely Software Distribution.

[12] C.J. Alpert, J. Hu, S.S. Sapatnekar, P.G. Villarubia, "A Practical Methodology for Early Buffer and Wire Resource Allocation", Design Automation Conference, 2001, pp. 189-194.

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

[14] C. Visweswariah, K. Ravindran, K. Kalafala. S.G. Walker, S. Narayan, "First-Order Incremental Block-Based Statistical Timing Analysis", *Design Automation Conference*, 2004.

[15] M.R. Guthaus, N. Venkateswaran, C.Visweswariah, V.Zolotov, "Gate Sizing using Incremental Parameterized Statistical Timing Analysis", *International Conference on Computer-Aided Design*, 2005.

[16] K.Chopra, S.Shah, A.Srivastava, A.Blaauw, D.Sylvester, "Parametric Yield Maximization using Gate sizing based on efficient Statistical Power and Delay gradient computation", *International Conference on Computer-Aided Design*, 2005.

[17] S.Raj, S.Vrudhula, J.M.Wang, "A methodology to improve timing yield in the presence of process variations", *Design Automation Conference*, 2004.