Stochastic Physical Synthesis for FPGAs with Pre-routing Interconnect Uncertainty and Process Variation

Yan Lin and Lei He
Electrical Engineering Department
University of California, Los Angeles
{ylin, lhe}@ee.ucla.edu, http://eda.ee.ucla.edu *

ABSTRACT

Process variation and pre-routing interconnect delay uncertainty affect timing and power for modern VLSI designs in nanometer technologies. This paper presents the first in-depth study on stochastic physical synthesis algorithms leveraging statistical static timing analysis (SSTA) with process variation and pre-routing interconnect delay uncertainty for FPGAs. Evaluated by SSTA with the placed and routed layout and measured at the same clock frequency, the stochastic clustering, placement and routing reduce the yield loss from 50 failed parts per 10 thousand parts (pp10K) for the deterministic flow to 9, 12 and 35pp10K respectively for MCNC designs. The majority of improvements are achieved during clustering and placement while routing stage has much less gain. The gain mainly comes from modeling interconnect delay uncertainty for clustering and from considering process variation for placement. When applying all stochastic algorithms concurrently, the yield loss is reduced to 5pp10K (a 10X reduction) with the mean delay reduced by 6.2% and the standard deviation reduced by 7.5%. On the other hand, stochastic clustering with deterministic placement and routing is a good flow with little change to the entire flow, but the yield loss is reduced from 50pp10K to 9pp10K, the mean delay is reduced by 5.0%, the standard deviation is reduced by 6.4%, and the runtime is slightly reduced compared to the deterministic flow. Finally, while its improvement over timing is small, stochastic routing is able to reduce the total wire length for the same routing channel width by 4.5% and to reduce runtime by 4.2% compared to deterministic routing.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Types and Design Styles

General Terms

Algorithms, Performance

*This paper is partially supported by NSF grant CCR-0306682 and Actel under UC MICRO program. Address comments to lhe@ee.ucla.edu.

FPGA'07, February 18–20, 2007, Monterey, California, USA.

Keywords

FPGA, process variation, uncertainty, stochastic, physical synthesis

1. INTRODUCTION

Because interconnect delay is dominant in modern VLSI designs [1], pre-routing interconnect delay estimation is needed for early stages of design automation. For FPGAs, the existing timing-driven physical synthesis algorithms are all deterministic and leverage timing slack analyzed by static timing analysis (STA) as a guidance. The interconnect delay is estimated by various methods in different design stages for STA. The actual post-routing interconnect delay may differ from the estimated delay, which introduces pre-routing interconnect delay uncertainty. Recently, a probabilistic approach by modeling interconnect delay as a random variable has been presented for buffer insertion in ASICs [2]. Similar approach considering high-level power and delay estimation inaccuracy by modeling the supply voltage as a random variable has been presented for voltage scheduling to minimize risk [3]. However, no existing work in the literature has considered pre-routing interconnect uncertainty during physical synthesis for FPGAs.

Process variation has gained a growing impact on modern VLSI designs as devices scale down to nanometer technologies. FPGAs are subject to variations in the operation of transistors comprising logic functionalities and switching muxes. With process variation, any near-critical paths may actually be statistically critical. *Statistical criticality* of a timing edge/node is defined as the probability that this edge/node is statistically timing critical considering process variation [4]. It depends on not only slack magnitude, but also circuit topology and correlation between edges. Slack itself analyzed by STA is based on the single critical path and ignores near-criticality. Statistical criticality has recently been studied in [4, 5, 6, 7], and applied to gate sizing in [8, 9] for ASICs and placement in [10] for FPGAs. However, only placement stage but not the entire physical synthesis flow has been investigated in [10]. In addition, the interconnect uncertainty and spatially correlated process variation are not considered in [10].

Considering both pre-routing interconnect uncertainty and process variation, the traditional timing-driven physical synthesis algorithms based on STA may not optimize for near-critical paths and may not optimize timing statistically. In this paper, we study the stochastic physical synthesis algorithms leveraging statistical static timing analysis (SSTA) with statistical criticality calculation for FPGAs. The baseline FPGA synthesis flow (see Figure 1) consists of CutMap [11], T-VPack [12] and VPR [12], which are the commonly used algorithms. The same synthesis result is applied to all chips for the same application. The delay distribution is evaluated by SSTA after detailed placement and routing. Stochastic clustering, placement and routing are studied considering interconnect delay

uncertainty and process variation. We first replace each individual synthesis stage with the stochastic algorithm and study its impact on timing. We then replace multiple stages concurrently and study the interaction between these stochastic algorithms in a style similar to the study for power-aware algorithms in [13]. Although not done here, our methods can be extended to high-level synthesis and technology mapping algorithms for statistical timing optimization[1].

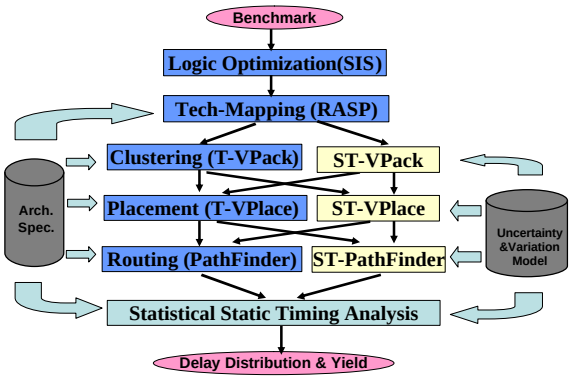

Figure 1: FPGA synthesis flow.

In order to quantify the benefit of stochastic algorithms, we use MCNC designs [15] in evaluation. When measured at the same clock frequency, the stochastic clustering, placement and routing reduce the yield loss from 50 failed parts per 10 thousand parts (pp10K) for the deterministic flow to 9, 12 and 35pp10K, respectively. Note that we use a heavily guard-banded delay as the cut-off delay such that the yield loss is very small (e.g. a few pp10K even for the baseline case) according to the usual practice in reality. The majority of improvements are achieved during clustering and placement. The gain mainly comes from modeling interconnect delay uncertainty for clustering and from considering process variation for placement. On the other hand, routing stage has much less gain. When applying all stochastic algorithms concurrently, the yield loss is reduced to 5pp10K with the mean delay reduced by 6.2% and the standard deviation reduced by 7.5%. Meanwhile, stochastic clustering with deterministic placement and routing is a good flow with little change to the entire flow, but the yield loss is reduced from 50pp10K to 9pp10K, the mean delay is reduced by 5.0%, the standard deviation is reduced by 6.4%, and the runtime is slightly reduced compared to the deterministic flow. While it improvement over timing is small, stochastic routing is able to reduce the total wire length for the same routing channel width by 4.5% and to reduce runtime by 4.2% compared to deterministic routing.

Note that the yield for FPGA devices is controlled by the manufacturer. Timing models for individual delay element must be guard-banded to consider process variation and environmental conditions. However, this guard-band can be either overly pessimistic (e.g. on long paths) or overly optimistic (e.g. on very short paths) for an unknown design, meaning that even some designs might actually run much faster than timing analysis says and others might fail to operate due to process variation outside the guard-banded value. The degree to which either occurs is based on the amount of guard-banding used in the timing model. Our study shows that by introducing statistical variation into the timing model and synthesis flow, we can improve timing and reduce yield loss for target designs implemented on the same FPGA fabric. Such improvement warrants more investigation of stochastic physical synthesis algorithms for FPGAs in the future.

The rest of the paper is organized as follows. Section 2 presents the background on the models of interconnect uncertainty and process variation, SSTA and the general experimental setting. Sections 3 to 5 present the stochastic clustering, placement and routing algorithms, and their results. Section 6 then combines the results from each individual stage and studies the interaction between them. Section 7 concludes the paper.

2. PRELIMINARIES

2.1 Interconnect Uncertainty Model

Similar to [2], we model the pre-routing interconnect delay as an independent Gaussian to consider interconnect uncertainty. The distribution of the Gaussian (i.e. the mean and the standard deviation) is approximated based on various methods in different design stages for the mean value and the statistics on post-routing interconnect delay for the standard deviation. To verify our modeling and algorithms, we evaluate our stochastic algorithms with the fully placed and routed layout.

2.2 Process Variation Model

Modern VLSI designs see a large impact from process variation as devices scale down to nanometer technologies. This variation can be classified as *global*, affecting all aspects of a given chip, *spatial/regional*, affecting geographic areas of the chip, or *local*, randomly affecting each individual transistor. Delay of a circuit element (e.g., an LUT or a routing switch) is a random variable under presence of process variation.

To model spatially correlated variation, we partition an FPGA chip into m grids and assume perfect correlation among the devices in the same grid. A standard Gaussian variable ΔS_k is associated with grid k. Given a set of correlated variables ΔS_k with the covariance matrix Q, principle component analysis (PCA) [16] can be used to transform ΔS_k into an uncorrelated set $\Delta S'_k$ (principle components) as

$$\Delta S_k = \sum_{l=1}^{m} \sqrt{\lambda_k} v_{kl} \Delta S'_l \qquad (1)$$

where λ_k is the k^{th} eigenvalue of the covariance matrix Q, v_{kl} is the k^{th} element of the l^{th} eigenvector of Q. In our study, we use the method from [17] to generate the covariance matrix. Global and local variations are also assumed as a set of independent random variables with PCA. Similar to [4], delay is then modeled in a first-order canonical form as

$$a_0 + \sum_{i=1}^{n} a_i \Delta X_i + a_{n+1} \Delta S_k + a_{n+2} \Delta R_a \qquad (2)$$

where a_0 is the nominal value, ΔX_i represents the variation for each global source of variation X_i (up to n sources), a_i represents the sensitivity to each global variation, ΔS_k represents the spatially correlated variation for grid k and can be represented by principle components using (1), a_{n+1} is the sensitivity to ΔS_k, ΔR_a is the variation of an independent random variable R_a from its mean value, and a_{n+2} is the sensitivity of R_a. Sensitivities are measured

[1] Leveraging the programmability of FPGA, FPGA chips for a same application can be programmed differently with respect to the individual variation maps (i.e., one customized configuration for a chip or a group of chips with a similar behavior). The so-called chip-wise FPGA design as first studied in [14] has a higher design cost but better design quality compared to stochastic design studied in this paper. Both are important as they offer different tradeoffs between cost and quality.

assuming that ΔX_i, ΔS_k and ΔR_a are standard Gaussians $N(0, 1)$. Although there are numerous sources of variation, only variations in lithographic effects affecting L_{eff} and dopant atoms in oxide layers affecting V_{th} are considered in this paper. SPICE simulation is performed to obtain sensitivities for each type of circuit element.

2.3 Statistical Static Timing Analysis

Statistical static timing analysis (SSTA) has recently been proposed to analyze timing with process variation [4, 16]. SSTA can however serve as a unified framework to handle both pre-routing interconnect uncertainty and process variation. The probabilistic equivalents of the "max", "min", "add" and "subtract" operations are involved in SSTA. With the delay in the canonical form, addition and subtraction are performed easily [4]. The max or min of two Gaussians is not a Gaussian, but is modeled as a Gaussian [18] and then expressed in the canonical form, which allows us to propagate the correlations due to global and spatial variations. With forward and backward traversals of the timing graph, the distribution of the arrival and requested arrival time for each node, and the statistical criticality for each node and edge can be calculated. The statistical criticality of an edge or node is defined as the probability that this edge or node is timing critical [4].

Given a cut-off delay T_{cut}, the *timing yield* is defined as the probability that the critical path delay is no longer than T_{cut} considering variation. Given the canonical form of the arrival time at the virtual sink in the timing graph, the mean T_μ and standard deviation T_σ of circuit delay can be calculated.[2] With a cut-off delay T_{cut}, the timing yield can then be computed using cumulative density function (CDF) of the standard Gaussian as $CDF((T_{cut} - T_\mu)/T_\sigma)$. The *yield loss* is defined as the number of parts that fail to meet the timing requirement out of 10,000 parts, in short, parts per 10K (pp10K). The yield loss can be easily calculated as $(1 - timing\_yield) \cdot 10K$.

2.4 General Experimental Setting

To quantify the benefit of our stochastic algorithms, we conduct the experiments on the largest MCNC designs [15]. We use the Berkeley predictive device model [19] at ITRS [20] $65nm$ technology node. Suggested in [21] for higher yield, we use the min-ED (energy-delay product) device setting (V_{dd} = 0.9v and V_{th} = 0.3v). The VPR FPGA toolset [12] implements an island style FPGA architecture resembling Altera's Stratix device [22] with 10 4-LUT clusters, and 60% length-4 and 40% length-8 wires. 1.2X of minimum routing channel width obtained by the deterministic synthesis flow is used for each design. The same routing channel width is used for stochastic algorithms. We implement a block-based SSTA from [4] with statistical criticality calculation for each timing edge/node. To model spatial correlation, each FPGA chip is partitioned into grids such that each grid contains five tiles in one dimension (around $0.5mm$ in $65nm$ technology). The correlation covariance coefficient decreases to 0.1 at $2mm$ distance. We assume a variation in each of L_{eff} and V_{th} of 10%, 10% and 6% at 3σ (i.e. a 99.73% chance that variation is within +/- 10% or 6% deviated from the nominal value) for global, spatial and local variations respectively unless specified otherwise. To evaluate the yield loss, we also assume a 2.5σ guard-banded delay [10] (i.e. the delay of each individual circuit element is modeled as $\mu + 2.5\sigma$[3]) in the deterministic flow evaluated by STA as the cut-off delay T_{cut}. Note that we use a heavily guard-banded delay as the cut-off delay T_{cut} such that the yield loss is very small (e.g. a few pp10K even for the baseline case) according to the usual practice in reality.

[2] Note that the mean delay T_μ may be larger than the nominal delay analyzed by STA due to the "max" operation with process variation.

[3] μ and σ are the nominal delay and standard deviation of delay for each circuit element with variation, respectively.

3. CLUSTERING

Modern island-style FPGAs have clustered logic blocks that contain multiple basic logic elements (BLEs). Each BLE consists of a pair of LUT and register. Clustering algorithm packs LUTs and registers into clusters under certain constraints, i.e. the number of BLEs, in/out pins and clocks of one cluster. The optimization goals of clustering including area, timing and routability have been studied in one of the representative timing-driven algorithms, $T\text{-}VPack$ [12], and have further been extended to reduce power in [13]. Below we first review the deterministic clustering algorithm T-VPack and then present our new stochastic algorithm, $ST\text{-}VPack$.

3.1 Timing-Driven Clustering T-VPack

During clustering, T-VPack first selects an unclustered BLE as the seed of a new cluster. An attraction function is calculated for all BLEs with respect to the current cluster. The BLE with the highest attraction value is then packed into the cluster until this cluster is fully utilized. If the cluster still has empty slots for BLE but lacks of cluster inputs, a hill-climbing technique is applied to look for BLEs that do not increase the number of inputs used by the cluster.

In order to optimize timing, BLEs on the critical path are packed into clusters since local connection delay within a cluster is much smaller than global interconnect delay. STA is performed using a constant delay model, i.e. 0.1 for logic and local connection delay and 1.0 for global interconnect delay. Given the slack analyzed by STA, the static criticality of edge i is defined as

$$ConnectionCriticality(i) = 1 - \frac{slack(i)}{MaxSlack} \qquad (3)$$

where $MaxSlack$ is the largest slack among all connections in the circuit. The static criticality of a BLE B is then defined as

$$\begin{aligned} Criticality(B) &= BaseBLECrit(B) + \\ &\quad \varepsilon \cdot TotalPathsAffected(B) + \\ &\quad \varepsilon^2 \cdot D_{source(B)} \end{aligned} \qquad (4)$$

where $BaseBLECrit(B)$ of BLE B is defined as the maximum criticality of edges connected to B if B is a seed BLE, or the maximum criticality of edges that are connected to both B and cluster C if B is not a seed BLE. The second and the third terms are the number of critical paths affected if B is packed into C, and the distance (the number of levels) from the timing graph source to B, respectively. These two terms only serve as tie-breakers with a small ε when two BLEs have the same $BaseBLECrit(B)$.

Based on (4), the attraction function between BLE B and cluster C is defined as

$$Attraction(B) = \lambda \cdot Criticality(B) + (1 - \lambda) \cdot \frac{Nets(B) \bigcap Nets(C)}{MaxNets} \qquad (5)$$

where the first term is for timing cost and the second term is for connection cost. $Net(B)$ and $Net(C)$ are sets of nets connected to B and C, respectively. λ is the tradeoff parameter between timing and connection, with a value of 0.75 adopted in T-VPack.

3.2 Stochastic Clustering ST-VPack

The deterministic timing model with constant interconnect delay used in T-VPack leads to some inaccuracy in estimation of where the critical path lies. T-VPack may try to shorten a path which is not part of the post-routing critical path due to this inaccurate estimation. Furthermore, any near-critical paths may become critical

considering process variation. Our new stochastic clustering algorithm, ST-VPack, leverages a statistical timing model and optimizes timing statistically.

To consider both interconnect uncertainty and process variation, we model interconnect delay for connection j as

$$d_j = d_0 + \sigma_p \Delta R_p + \sigma_i \Delta R_i \quad (6)$$

where ΔR_i models interconnect uncertainty[4] and is independent from each other for all interconnects, ΔR_p models the correlation between interconnects due to global and spatial process variations and is shared by all interconnects, and σ_i and σ_p are 0.2 and 0.1 (relative standard deviation of 20% or 10%) respectively. Any values of σ_p between 0.0 to 0.1 and σ_i between 0.1 and 0.3 work fairly well, however (to be presented in Table 1). d_0 is set to 1.0 for all global interconnects same as that in T-VPack. Both ΔR_p and ΔR_i are standard Gaussians $N(0, 1)$, and are independent from each other. (6) is in the first-order canonical form similar to (2). Note that the interconnect uncertainty may not be a Gaussian but is approximated as a Gaussian, ΔR_i.

With the delay model (6), SSTA can be performed with statistical criticality calculated for each timing edge/node. Similar to STA in T-VPack, SSTA is only performed once before clustering in ST-VPack. We then modify (4) for ST-VPack as

$$SCriticality(B) = SBaseBLECrit(B)^{\eta} + \varepsilon^2 \cdot D_{source(B)} \quad (7)$$

where $SBaseBLECrit(B)$ of BLE B is calculated as the maximum statistical criticality of edges that are connected to both B and cluster C if B is not a seed BLE. For a seed BLE, $SBaseBLECrit(B)$ is defined as the statistical criticality of B, which is different from the scenario in T-VPack. $TotalPathsAffected(B)$ in (4) depends on circuit topology and is removed from (7) since it has already been considered in statistical criticality. $D_{source(B)}$ is still kept in (7) as a tie-breaker such that BLEs with the same $SBaseBLECrit(B)$ are packed from one end of a chain of BLEs rather than from the middle. Similar to the cost functions in placement and routing (to be discussed), a new exponent parameter η is introduced to control the relative importance of connections with different criticalities. We experimentally select an η of 0.1 for the best timing yield. The new attraction function is then expressed as,

$$Attraction(B) = \lambda \cdot SCriticality(B) + (1 - \lambda) \cdot \frac{Nets(B) \bigcap Nets(C)}{MaxNets} \quad (8)$$

with λ of 0.75 for the same tradeoff between timing and connection as that in T-VPack.

3.3 Experimental Results

We first study the impact of the combination of two uncertainty sources, interconnect uncertainty σ_i and process variation σ_p, in delay model (6) on ST-VPack. Table 1 presents a few combinations[5] of different σ_i and σ_p and the corresponding post-routing mean (Tmean) and standard deviation (Tsigma) of circuit delay. In this table, group I does not consider interconnect uncertainty ($\sigma_i = 0$) while group II does not consider process variation ($\sigma_p = 0$). It is clear that group II leads to a smaller mean delay by modeling interconnect uncertainty while all combinations in both groups lead to a similar standard deviation. On the other hand, group III considers both interconnect uncertainty and process variation. Further considering process variation does not have a significant impact on the mean delay. Based on Table 1, the gain of ST-VPack is mainly due to modeling interconnect uncertainty. This is further explained in Figure 2, which compares the probability density functions (PDF) for post-routing delay normalized with respect to the estimated one during clustering (i.e. the delay variance introduced by interconnect uncertainty) and post-routing delay with process variation normalized with respect to the nominal one (i.e. the delay variance introduced by process variation). The statistics is based on all global interconnects of all designs. Clearly, interconnect uncertainty leads to a more significant delay variance (i.e. a much wider delay spread) in clustering stage. In the rest of the paper, σ_i and σ_p are set to 0.2 and 0.1, respectively.

	I		II		III	
σ_i	0.0	0.0	0.1	0.2	0.1	0.2
σ_p	0.0	0.1	0.0	0.0	0.1	0.1
Tmean (ns)	22.5	22.6	21.6	21.5	21.8	21.7
Tsigma (ns)	3.35	3.36	3.26	3.24	3.20	3.19

Table 1: The effect of standard deviation due to interconnect uncertainty σ_i and process variation σ_p (based on the geometric mean of 20 MCNC designs).

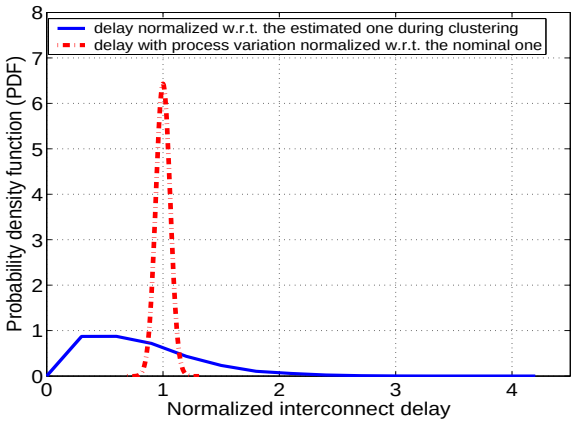

Figure 2: Comparison between probability density functions (PDF) for (i) post-routing delay normalized with respect to (w.r.t.) the estimated delay during clustering, and (ii) post-routing delay with process variation normalized w.r.t. the nominal one.

Figure 3 shows the normalized nominal (Tnorm), mean (Tmean) and standard deviation (Tsigma) of post-routing circuit delay obtained by ST-VPack for each benchmark. Each is normalized to its counterpart in T-VPack. The same deterministic placement and routing algorithms are used to generate detailed layouts. The nominal delay is evaluated by STA without considering process variation. The mean and standard deviation of delay are evaluated by SSTA with variation. Compared to T-VPack, ST-VPack reduces the nominal, mean and standard deviation of delay for most of benchmarks with few exceptions, which are due to the heuristic statistical cost function in ST-VPack. For example, ST-VPack increases the mean delay by 2% but reduces the standard deviation by 32% for $dsip$, which results in a much smaller yield loss compared to T-VPack. On average, ST-VPack reduces the nominal, mean and standard deviation of delay by 3.7% (up to 12.5%), 5.0% (up to 13.0%) and 6.4% (up to 31.8%), respectively. The impact of ST-VPack on timing distribution (Tmean and Tsigma) is larger than that on the nominal delay due to the fact that we are aiming to optimize timing statistically for ST-VPack.

[4] ΔR_i can be used to considered both interconnect uncertainty and local process variation, but interconnect uncertainty is the dominant component between the two.

[5] Although only a few combinations are presented in Table 1, our experimental results show a similar trend for all combinations for σ_p between 0.0 and 0.1, and σ_i between 0.0 and 0.3.

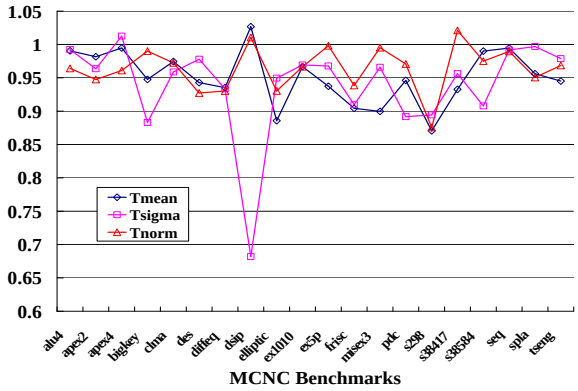

Figure 3: Normalized nominal, mean and standard deviation of circuit delay obtained by ST-VPack.

Table 2 compares T-VPack and ST-VPack in more details. On average, ST-VPack reduces the number of interconnect connections by 4.2%, delay (or timing graph depth) after packing in constant delay model (i.e. 1.0 for global interconnect and 0.1 for logic and local connection) by 4.0% compared to T-VPack. ST-VPack achieves almost the same number of clusters after packing. T-VPack and ST-VPack take 6 and 8 seconds to pack all 20 benchmarks, respectively. In the rest part of the paper, we only present the overall runtime (clustering, placement and routing), which is more meaningful. ST-VPack does not have significant impact on overall runtime. When evaluated with 2.5σ guard-banded delay, ST-VPack reduces the yield loss from 50pp10K for the deterministic flow to 9pp10K. Clearly, ST-VPack is able to effectively improve timing statistically and reduce yield loss without routability, area and runtime overhead compared to T-VPack.

	T-VPack	ST-VPack (% Diff.)
# of connections	5787	5545 (-4.2%)
# of clusters	381.3	381.8 (0.1%)
# of levels	9.3	8.9 (-4.0%)
nominal delay (ns)	21.2	20.4 (-3.7%)
mean delay (ns)	22.9	21.7 (-5.0%)
standard deviation (ns)	3.41	3.19 (-6.4%)
yield loss (pp10K)	50	9
runtime (s)	70.7	70.3 (-0.6%)

Table 2: Clustering results (based on the geometric mean of 20 MCNC designs).

4. PLACEMENT

After packing, clusters are placed to physical locations on the FPGA chip. For FPGAs, the typical placement algorithm is simulated annealing as in the timing-driven algorithm, *T-VPlace* [23], in VPR [12]. Below we first review T-VPlace and then present our new stochastic algorithm, *ST-VPlace*.

4.1 Timing-Driven Placement T-VPlace

Simulated annealing is a heuristic and iterative algorithm in which moves (swaps of logic cells) are accepted or rejected based on a cost function and an annealing temperature. T-VPlace considers both wiring and timing costs. Wiring cost is expressed as

$$Wiring\_Cost = \sum_{i=1}^{N_{nets}} q(i)[bb_x(i) + bb_y(i)] \quad (9)$$

where N_{nets} is the number of nets in the circuit. The cost of net i is determined by its horizontal and vertical spans, $bb_x(i)$ and $bb_y(i)$. Scaling factor $q(i)$ compensates for multi-terminal nets.

Timing cannot be optimized explicitly since it is too expensive to perform a timing analysis after each move. A heuristic timing cost is calculated based on the static criticality of each edge defined in (11), the delay of each edge $d(i,j)$ and the criticality exponent β. The timing costs of edge (i,j) and for a placement solution are

$$Timing\_Cost(i,j) = d(i,j) \cdot criticality(i,j)^{\beta} \quad (10)$$

$$criticality(i,j) = 1 - slack(i,j)/D_{max} \quad (11)$$

$$Timing\_Cost = \sum_{i,j} Timing\_Cost(i,j) \quad (12)$$

respectively, where $d(i,j)$ is obtained from the delay lookup matrix and the current placement, D_{max} is the critical path delay, and $slack(i,j)$ is the timing slack of each edge. Both D_{max} and slack are calculated by STA, which is performed once at every annealing temperature. The criticality exponent β is used to control the relative importance of connections with different criticalities.

The overall cost function is then shown in (13), where λ is the trade-off parameter between the timing and wiring costs. The previous timing and previous wiring costs are updated once every temperature. The temperature and ΔC are used to decide whether a move is to be accepted or rejected. It was shown in [23] that $\beta = 8$ and $\lambda = 0.5$ give the best timing and wiring trade-off.

$$\Delta C = \lambda \frac{\Delta Timing\_Cost}{Previous\_Timing\_Cost} + (1-\lambda)\frac{\Delta Wiring\_Cost}{Previous\_Wiring\_Cost} \quad (13)$$

4.2 Stochastic Placement ST-VPlace

The interconnect delay estimated in T-VPlace is based on 2-pin net routing for each pair of locations without considering congestion. The actual delay after routing may differ from the estimated delay in placement, mainly due to the impact of congestion and multi-terminal nets. This introduces interconnect delay uncertainty in placement. In addition, any near-critical paths may become critical with process variation. Figure 4 compares the PDFs for post-routing delay normalized with respect to the estimated one during placement (i.e. the delay variance introduced by interconnect uncertainty) and post-routing delay with process variation normalized with respect to the nominal one (i.e. the delay variance introduced by process variation). The statistics is based on near-critical interconnects (static criticality greater than 0.9 after routing) of all designs. As shown in this figure, more than 70% of interconnects have an estimation error within 1% while the relative standard deviation is 6% due to process variation. It is clear that process variation leads to a more significant delay variance (i.e. a much wider delay spread) and needs to be considered in placement. Because the study in Figure 2 of Section 3.3 has shown that only the dominant uncertainty source impacts the clustering results, in this section we assume that only the dominant uncertainty source, i.e. process variation, should be considered for placement.[6]

In order to consider process variation during placement, we calculate a delay matrix in the canonical form instead of the nominal delay matrix for each pair of locations. The delay in the canonical form for a routing path is calculated by performing statistical addition for the interconnect switches in that path. Given the delay in the canonical form for each edge, SSTA instead of STA is performed at

[6]Interconnect uncertainty can be modeled as an independent Gaussian with a small relative standard deviation with respect to the estimated delay. Our experimental results however show that such a small relative standard deviation, e.g. 0.5%, has little impact on the timing.

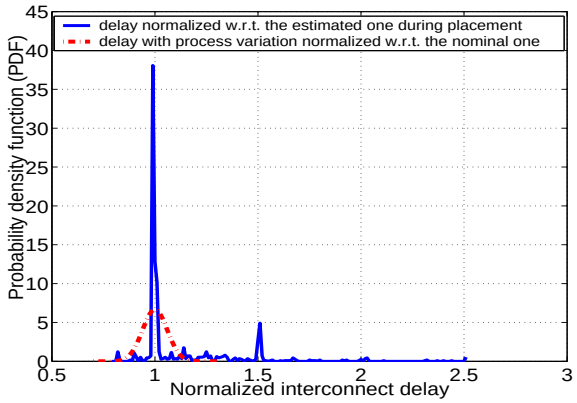

Figure 4: Comparison between PDFs for (i) post-routing delay normalized w.r.t. the estimated delay in placement, and (ii) post-routing delay with process variation normalized w.r.t. the nominal one.

each temperature to obtain the statistical criticality. Instead of using the static timing cost function (10), we define statistical timing cost functions for each edge (i, j) and a placement solution as

$$STiming\_Cost(i,j) = d_\mu(i,j) \cdot SCriticality(i,j)^{\beta'} \quad (14)$$

$$STiming\_Cost = \sum_{i,j} STiming\_Cost(i,j) \quad (15)$$

where $d_\mu(i,j)$ is the mean delay for each edge and $SCriticality(i,j)$ is the statistical criticality. $d_\mu(i,j)$ is the same as $d(i,j)$ in (10) as statistical "max" operation is not involved in a routing tree. Statistical criticality exponent, β', is a constant parameter. We experimentally tune β' to be 0.5 for the best timing yield. The overall cost function in ST-VPlace is as,

$$\Delta C = \lambda \frac{\Delta STiming\_Cost}{Previous\_STiming\_Cost} + (1-\lambda)\frac{\Delta Wiring\_Cost}{Previous\_Wiring\_Cost} \quad (16)$$

We use the same λ of 0.5 for the same timing and wiring tradeoff in ST-VPlace. The same annealing scheme in T-VPlace is also adopted in ST-VPlace. The goal of ST-VPlace is to perform placement considering process variation, and to optimize for the statistical timing leveraging the back-end SSTA.

4.3 Experimental Results

Figure 5 shows the normalized nominal, mean and standard deviation of post-routing circuit delay obtained by ST-VPlace for each benchmark. Each is normalized to its counterpart in T-VPlace. The same deterministic clustering and routing algorithms are used in both flows. Compared to T-VPlace, ST-VPlace reduces the mean and standard deviation of delay for most of benchmarks except for $dsip$. ST-VPlace increases the mean delay by 2% but reduces the standard deviation of delay by 23% for $dsip$, which results in a smaller yield loss. ST-VPlace reduces the nominal delay for most benchmarks except for des, $frisc$, $misex3$ and $s38417$. However, a smaller mean and standard deviation of delay is achieved for each of these benchmarks due to the heuristic statistical cost function in ST-VPlace. On average, ST-VPlace reduces the nominal, mean and standard deviation of delay by 3.3% (up to 12.3%), 4.0% (up to 14.2%) and 6.1% (up to 22.7%), respectively. Similar to the stochastic clustering ST-VPack, ST-VPlace has larger impact on timing distribution than that on the nominal delay due to its statistical fashion. The impact of ST-VPlace on timing is similar to that of ST-VPack. Nevertheless, the gain of ST-VPlace mainly comes from considering process variation, different from ST-VPack where the gain is mainly due to modeling interconnect uncertainty.

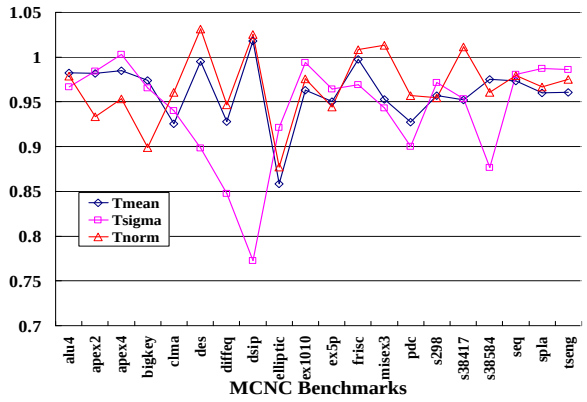

Figure 5: Normalized nominal, mean and standard deviation of circuit delay obtained by ST-VPlace.

	T-VPlace	ST-VPlace (% Diff.)
total wire length	27610	27983 (1.3%)
nominal delay (ns)	21.2	20.5 (-3.3%)
mean delay (ns)	22.9	22.0 (-4.0%)
standard deviation (ns)	3.41	3.20 (-6.1%)
yield loss (pp10K)	50	12
runtime (s)	70.7	219.8 (210.7%)

Table 3: Placement results (based on the geometric mean of 20 MCNC designs).

Table 3 compares T-VPlace and ST-VPlace in more details. When evaluated with 2.5σ guard-banded delay, ST-VPlace reduces the yield loss from 50pp10K for the deterministic flow to 12pp10K. On the other hand, ST-VPlace increases the total wire length after routing by 1.3% and takes $3.1X$ runtime compared to T-VPlace. Nevertheless, the block-based SSTA has linear time complexity of $O((n+m)\cdot(|V|+|E|))$ where n and m are number of global variation sources and number of grids, respectively. In addition, SSTA is only performed once at every annealing temperature. The average complexity of SSTA is still $O(1)$ same as STA in placement. ST-VPlace therefore has the same average complexity of $O(|C|^{4/3})$ as T-VPlace, where $|C|$ is the number of clusters.

5. ROUTING

After clusters are placed to physical locations on the FPGA, routing is performed to determine which programmable interconnect switches should be turned on to connect required interconnects. Below, we first review the deterministic routing algorithm, $PathFinder$ [24, 12], in VPR and then present our stochastic routing algorithm, $ST\text{-}PathFinder$.

5.1 Timing-Driven Routing PathFinder

The routing algorithm in VPR is developed based on an iterative algorithm PathFinder. During each iteration, routing is performed for one net at a time with congestion allowed. A wave expansion algorithm is invoked k times for a k-sink net, with the more critical sink being routed first. After one entire routing iteration, historical congestion costs are updated for routing resources and STA is performed to update the slack for each net. Considering the connection

to sink j of net i, the cost to include a routing resource node n is expressed as,

$$\begin{aligned} Cost(n) &= Crit(i,j) \cdot delay(n, topology) \\ &+ [1 - Crit(i,j)]b(n)h(n)p(n) \end{aligned} \quad (17)$$

where the first term is for timing cost and the second term is for wire congestion cost. $delay(n, topology)$ is the delay from the current partial routing to node n. $b(n)$, $h(n)$ and $p(n)$ are the base, historical and present costs for n, respectively. $Crit(i,j)$ is the criticality for each connection and is a tradeoff factor between timing and wire congestion for a resource node. $Crit(i,j)$ is defined as

$$Crit(i,j) = max([MaxCrit - \frac{slack(i,j)}{D_{max}}]^{\eta}, 0) \quad (18)$$

where $slack(i,j)$ is the slack of each connection and D_{max} is the critical path delay, both analyzed by STA. $MaxCrit$ is the maximum criticality that any connection can have, and η is the criticality component. Both are the parameters to control how the slack of a connection impacts the congestion and delay tradeoff in the cost function. Setting $MaxCrit$ to 0.99 prevents that the nets on critical path ignore congestion and can achieve a better routability without affecting circuit timing compared to a $MaxCrit$ of 1.0. In addition, η of 1 leads to the best circuit timing.

$PathCost(n)$ is the total cost of the path including the current partial routing tree and the node n, and is defined as

$$PathCost(n) = \sum_{l \in path\ from\ RT(i)\ to\ n} Cost(l) \quad (19)$$

The total cost of a routing tree includes the cost of current partial routing tree and the expected cost from node n to target sink j, and is defined as

$$TotalCost(n) = PathCost(n) + \alpha \cdot ExpectedCost(n,j) \quad (20)$$

where the expected cost $ExpectedCost(n,j)$ is based on the assumption that the same type of wires are used for the remaining routing without congestion. It has been shown that an α of 1.2 leads to the best timing.

5.2 Stochastic Routing ST-PathFinder

In the routing stage, the interconnect estimation occurs when predicting delay from the current partial routing to the target sink, and has the highest accuracy within all design stages. On the other hand, timing analysis is only performed after an entire routing iteration. Interconnect uncertainty has little impact on one of the key parameters, $Crit(i,j)$, in (18). We therefore only consider process variation in SSTA for ST-PathFinder. Similar to the stochastic placement, routing path delay in the canonical form is calculated by performing statistical addition for the interconnect switches in that path. SSTA is then performed to calculate the statistical criticality for each interconnect edge. We modify (18) as,

$$SCrit(i,j) = min(SCriticality(i,j)^{\eta'}, MaxCrit) \quad (21)$$

where $SCriticality(i,j)$ is the statistical criticality for each connection, criticality exponent η' is a constant parameter. In (18), $MaxCrit$ and "max" operation set the upper and lower bounds of static criticality to $MaxCrit$ and 0, respectively. Since the $SCriticality$ is non-negative in nature, we replace the "max" in (18) with the "min" operation in (21) to set the upper bound for $SCriticality$. Based on (21), we then have the statistical cost function for node n as,

$$\begin{aligned} SCost(n) &= SCrit(i,j) \cdot delay_{\mu}(n, topology) \\ &+ [1 - SCrit(i,j)]b(n)h(n)p(n) \end{aligned} \quad (22)$$

where $delay_{\mu}(n, topology)$ is the mean delay from the current partial routing to node n. $delay_{\mu}(n, topology)$ in (22) is same as $delay(n, topology)$ in (17) since the statistical "max" operation is not involved in a routing tree. Plugging (22) into (19) and then (20) gives us the new statistical cost function considering process variation for an entire routing tree. By using the statistical criticality, the routing order of sinks in one net and the tradeoff between timing and wire congestion costs for a resource node are changed. We experimentally tune η' to be 0.2 for the best timing yield. All other parameters in ST-PathFinder are the same as those in PathFinder.

5.3 Experimental Results

Figure 6 shows the normalized nominal, mean and standard deviation of post-routing circuit delay obtained by ST-PathFinder for each benchmark. Each is normalized to its counterpart in PathFinder. The same deterministic clustering and placement algorithms are used in both flows. Compared to PathFinder, ST-PathFinder reduces (or achieves as good as) the nominal and mean delay values (except for $misex3$ with 8% larger nominal delay) for most of benchmarks. On the other hand, ST-PathFinder has little impact on the standard deviation of delay. Note that ST-PathFinder achieves inferior results especially the nominal delay for some benchmarks (e.g. 8% nominal delay overhead for $misex3$) compared to PathFinder, which is due to the heuristic statistical cost function adopted in ST-PathFinder. On average, ST-PathFinder reduces the nominal, mean and standard deviation of delay by 1.4% (up to 7.8%), 1.4% (up to 7.8%) and 0.7% (up to 5.2%), respectively, compared to PathFinder. Compared to clustering and placement stages, the impact of stochastic routing on timing is much smaller due to the fact routing stage has the smallest design flexibility.

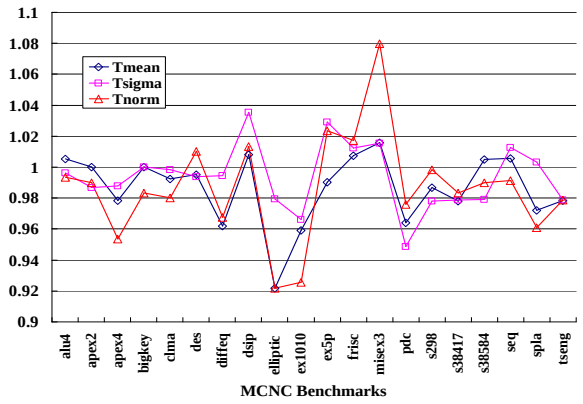

Figure 6: Normalized nominal, mean and standard deviation of circuit delay in ST-PathFinder.

Table 4 compares PathFinder and ST-PathFinder in more details. When evaluated with 2.5σ guard-banded delay, ST-PathFinder reduces the yield loss from 50pp10K for the deterministic flow to 35pp10K. Although SSTA is more expensive than STA, SSTA or STA is only performed once after one entire routing iteration. ST-PathFinder reduces the average number of routing iterations required for a successful routing from 22 to 15 compared to PathFinder

and therefore consumes less runtime. It is due to the fact that ST-PathFinder uses statistical criticality to achieve a better balance between weights of timing and wire lengths in the cost function for each net. Besides of a 4.2% of runtime reduction, ST-PathFinder also reduces the total wire length after routing by 4.5% due to this better balanced cost function.

	PathFinder	ST-PathFinder (% Diff.)
total wire length	27610	26356 (-4.5%)
# of iterations	22	15 (-32.4%)
nominal delay (ns)	21.2	20.9 (-1.4%)
mean delay (ns)	22.9	22.5 (-1.4%)
standard deviation (ns)	3.41	3.38 (-0.7%)
yield loss (pp10K)	50	35
runtime (s)	70.7	67.7(-4.2%)

Table 4: Routing results (based on the geometric mean of 20 MCNC designs).

6. INTERACTION BETWEEN CLUSTERING, PLACEMENT AND ROUTING

The three previous sections study each individual stochastic algorithm in isolation and the impact of each one on timing improvement. Below we combine the stochastic algorithms and study the interactions between them. The results for yield loss in pp10K obtained by all eight possible combinations of algorithms are presented in Table 5. We also summarize other results including the nominal, mean and standard deviation of circuit delay, and runtime for all combinations of algorithms in Table 5. The delay values are presented in the difference compared to the flow consisting of all deterministic algorithms. The runtime is also normalized with respect to the deterministic flow.

Cluster	D	S	D	D	S	S	D	S
Place	D	D	S	D	S	D	S	S
Route	D	D	D	S	D	S	S	S
yield loss in pp10K								
alu4	33.3	25.2	16.5	36.3	10.7	23.7	16.0	9.2
apex2	39.3	18.9	23.1	35.3	9.4	17.4	15.9	9.6
apex4	65.8	63.9	48.0	37.1	21.4	67.6	43.1	28.8
bigkey	227.5	31.2	111.4	227.5	93.1	35.5	127.2	83.9
clma	32.1	11.9	2.5	26.6	1.9	8.3	3.3	1.5
des	177.8	55.3	87.5	159	43.5	102	93.8	26.2
diffeq	42.9	7.8	2.1	22.9	0.4	8.1	1.7	0.3
dsip	244.4	35.1	74.5	312	66.4	36.1	135	213
elliptic	33.1	1.5	0.5	5.4	0.3	0.6	0.7	0.3
ex1010	95.0	33.9	37.9	27.2	18.9	26.5	12.9	11.3
ex5p	37.8	6.1	8.3	38.3	4.8	16.3	3.6	8.7
frisc	24.0	1.1	17.2	30.1	1.1	1.5	23.6	0.8
misex3	6.7	0.2	0.8	11.6	0.9	0.9	0.6	0.5
pdc	28.5	1.9	1.2	6.9	0.5	5.1	0.4	1.0
s298	33.3	0.4	10.4	21.0	0.3	0.8	15.8	1.0
s38417	74.2	11.3	17.8	40.3	20.0	6.1	11.0	32.5
s38584	48.6	18.2	9.6	44.7	4.5	15.3	26.3	3.7
seq	43.1	35.5	19.4	53.5	17.3	48.7	28.1	24.4
spla	49.3	16.1	16.3	25.8	10.3	8.5	6.6	7.4
tseng	62.1	21.1	29.5	37.2	10.4	14.0	50.7	11.4
Geo.	50.2	9.3	11.8	35.2	5.3	10.3	11.0	5.4
Tnorm	21.2	-3.7%	-3.3%	-1.4%	-6.4%	-4.1%	-3.6%	-6.3%
Tmean	22.9	-5.0%	-4.0%	-1.4%	-5.9%	-4.7%	-4.0%	-6.2%
Tsigma	3.4	-6.4%	-6.1%	-0.7%	-8.8%	-6.1%	-6.3%	-7.5%
runtime	1.0X	0.99X	3.1X	0.96X	3.0X	0.97X	3.1X	3.0X
wire	27610	0.8%	1.3%	-4.5%	3.2%	-3.4%	-3.4%	-1.6%

Table 5: Combined results. 'D' and 'S' stand for deterministic and stochastic, respectively.

The majority of improvements are achieved during clustering and placement. The gain mainly comes from modeling interconnect delay uncertainty for clustering and from considering process variation for placement. When applying stochastic clustering and placement concurrently, we can achieve a smaller nominal, mean and standard deviation of delay than applying any one of them alone. However, there exists some overlap between gains in clustering and placement. On the other hand, routing stage has much less gain. When applying stochastic routing with other stochastic algorithms concurrently, the impact of routing is dominated by other algorithms. It is due to the fact that routing stage has the smallest design flexibility.

We also present the total wire length achieved by each flow in Table 5. Compared to the deterministic flow, the stochastic clustering and placement increase total wire length by 0.8% and 1.3%, respectively. When applying stochastic clustering and placement concurrently with deterministic routing, the wire length overhead increases to 3.2%. On the other hand, the stochastic routing reduces wire length by 4.5%. When applying stochastic routing with stochastic placement, clustering or both concurrently, the wire length is reduced by 3.2%, 3.4% and 1.6% respectively compared to the deterministic flow[7].

When all stochastic algorithms are applied concurrently, the yield loss is reduced from 50pp10K for the deterministic flow to 5pp10K measured with the 2.5σ guard-banded delay in deterministic flow as the cut-off delay. In addition, the stochastic flow reduces the nominal, mean and standard deviation of delay by 6.3% (up to 17.6%), 6.2% (up to 15.4%) and 7.5% (up to 22.7%) respectively but takes $3.0X$ runtime compared to the deterministic flow. Note that this stochastic flow achieves a larger yield loss for some design, e.g. *dsip*, compared to the flow using only stochastic clustering or placement due to the overhead introduced by stochastic routing for this particular design. For a good gain with less runtime and little change to the entire flow, we may apply only stochastic clustering with deterministic placement and routing. This flow reduces the nominal, mean and standard deviation of delay by 3.7% (up to 12.5%), 5.0% (up to 13.0%) and 6.4% (up to 31.8%), respectively, and reduces runtime slightly compared to the deterministic flow.

process variation settings (3σ)						
global	5.0%	10.0%	15.0%	20.0%	25.0%	30.0%
spatial	5.0%	10.0%	15.0%	20.0%	25.0%	30.0%
local	3.0%	6.0%	9.0%	12.0%	15.0%	18.0%
deterministic flow						
Tmean (ns)	21.7	22.9	24.4	26.2	28.2	30.2
Tsigma (ns)	1.8	3.4	5.0	6.4	7.8	9.2
stochastic flow						
Tmean (ns)	20.3 (-6.5%)	21.5 (-6.2%)	23.0 (-5.8%)	24.8 (-5.5%)	26.7 (-5.3%)	28.6 (-5.1%)
Tsigma (ns)	1.6 (-6.6%)	3.1 (-7.5%)	4.6 (-8.1%)	5.9 (-8.2%)	7.2 (-8.1%)	8.5 (-8.0%)

Table 6: Comparison of mean delay and standard deviation between deterministic and stochastic flows under various process variation assumptions (based on the geometric mean of 20 MCNC designs).

Table 6 compares our stochastic flow with the deterministic flow under various process variation assumptions. Each flow consists of all deterministic or all stochastic algorithms. The 3σ of global/spatial/local variations are in the range between 5%/5%/3% and 30%/30%/18%. The mean and standard deviation of delay values in the stochastic flow are presented in the difference compared to those in the deterministic flow. The reduction ranges of mean and standard deviation are from 5.1% to 6.5% and from 6.6% to 8.2%, respectively. It is clear that the stochastic flow consistently achieves a smaller mean and standard deviation of circuit delay under various process variation assumptions, and therefore result in a smaller yield loss compared to the deterministic flow.

[7]Note that we assume the same routing channel width for both deterministic and stochastic flows and the wire length reduction due to stochastic routing could be converted to routing congestion reduction.

7. CONCLUSIONS AND DISCUSSIONS

In this paper, we have presented the first in-depth study on stochastic physical synthesis algorithms leveraging SSTA with process variation and interconnect delay uncertainty for FPGAs. We have studied stochastic clustering, placement and routing algorithms as well as the interaction between them. Evaluated by SSTA with the fully placed and routed layout and measured at the same clock frequency, the stochastic clustering, placement and routing reduce the yield loss from 50 failed parts per 10 thousand parts (pp10K) for the deterministic flow to 9, 12 and 35pp10K respectively. All achieve a smaller mean and standard deviation of circuit delay compared to the deterministic algorithms. The majority of improvements are achieved during clustering and placement. The gain mainly comes from modeling interconnect uncertainty for clustering and considering process variation for placement. On the other hand, routing stage has much less gain.

When applying all stochastic algorithms concurrently, the yield loss is reduced to 5pp10K (a $10X$ reduction) with the mean delay reduced by 6.2% and the standard deviation reduced by 7.5% compared to the deterministic flow. In addition, our stochastic algorithms consistently achieve a smaller mean and standard deviation of circuit delay, which result in a smaller yield loss, under various variation assumptions. We also show an overlap existed between gains of clustering and placement. Stochastic clustering with deterministic placement and routing is a good flow with little change to the entire flow, but the yield loss is reduced from 50pp10K to 9pp10K, the mean delay is reduced by 5.0%, the standard deviation is reduced by 6.4%, and the runtime is slightly reduced compared to the deterministic flow. The significant improvement observed by our study warrants more investigation on stochastic physical synthesis for FPGAs in the future. While its improvement over timing is small (1.4%), stochastic routing is able to reduce the total wire length for the same routing channel width by 4.5% and to reduce runtime by 4.2% compared to deterministic routing.

In the future, we plan to extend our stochastic algorithms to high-level synthesis and technology mapping to consider interconnect uncertainty. The statistical criticality calculation used in our paper is from [4], which assumes independence between timing edges and may be inaccurate. In addition, we assume Gaussian distribution to model interconnect uncertainty and process variation. We will also extend our stochastic algorithms for non-Gaussian SSTA and more accurate statistical criticality calculation such as that in [5, 6, 25].

8. ACKNOWLEDGEMENT

The authors thank Dr. Mike Hutton from Altera for stimulating discussions on stochastic placement in the early stage of this research.

9. REFERENCES

[1] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance optimization of VLSI interconnect layout," *Integration, the VLSI Journal*, vol. 21, pp. 1–94, 1996.

[2] V. Khandelwal, A. Davoodi, A. Nanavati, and A. Srivastava, "A probabilistic approach to buffer insertion," in *Proc. Intl. Conf. Computer-Aided Design*, November 2003.

[3] A. Davoodi and A. Srivastava, "Voltage scheduling under unpredictabilities: A risk management paradigm," in *Proc. Intl. Symp. Low Power Electronics and Design*, August 2003.

[4] 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.

[5] Y. Zhan, A. J. Strojwas, M. Sharma, and D. Newmark, "Statistical critical path analysis considering correlations," in *Proc. Intl. Conf. Computer-Aided Design*, November 2005.

[6] X. Li, J. Le, M. Celik, and L. T. Pileggi, "Defining statistical sensitivity for timing optimization of logic circuits with large-scale process and enviromental variations," in *Proc. Intl. Conf. Computer-Aided Design*, pp. 844–851, November 2005.

[7] J. Xiong, V. Zolotov, N. Venkateswaran, and C. Visweswariah, "Criticality computation in parameterized statistical timing," in *Proc. Design Automation Conf.*, July 2006.

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

[9] D. Sinha, N. Shenoy, and H. Zhou, "Statistical gate sizing for timing yield optimization," in *Proc. Intl. Conf. Computer-Aided Design*, November 2005.

[10] Y. Lin, M. Hutton, and L. He, "Placement and timing for FPGAs considering variations," in *Proc. Intl. Conf. Field-Programmable Logic and its Application*, August 2006.

[11] J. Cong and Y. Hwang, "Simultaneous depth and area minimization in LUT-based FPGA mapping," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 1995.

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

[13] J. Lamoureux and S. J. Wilton, "On the interaction between power-aware FPGA CAD algorithms," in *Proc. Intl. Conf. Computer-Aided Design*, pp. 701–708, November 2003.

[14] L. Cheng, J. Xiong, L. He, and M. Hutton, "FPGA performance optimization via chipwise placement considering process variations," in *International Conference on Field-Programmable Logic and Applications*, August 2006.

[15] S. Yang, "Logic synthesis and optimization benchmarks, version 3.0," tech. rep., Microelectronics Center of North Carolina (MCNC), 1991.

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

[17] J. Xiong, V. Zolotov, and L. He, "Robust extraction of spatial correlation," in *Proc. Intl. Symp. Physical Design*, April 2006.

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

[19] U. of Berkeley Device Group, "Predictive technology model," in *http://www.device.eecs.berkeley.edu/ ptm/mosfet.html*, 2002.

[20] International Technology Roadmap for Semiconductor in *http://public.itrs.net/*, 2003.

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

[22] Altera Corporation, "Stratix programmable logic device family data sheet," Aug 2002.

[23] A. Marquardt, V. Betz, and J. Rose, "Timing-driven placement for FPGAs," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, Feb 2000.

[24] L. McMurchie and C. Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, Feb 1995.

[25] J. Xiong, V. Zolotov, N. Venkateswaran, and C. Visweswariah, "Criticality computation in parameterized statistical timing," in *ACM/IEEE International Workshop on Timing Issues, San Jose, California*, Feb. 2006.