Variation Aware Routing for Three-Dimensional FPGAs

Chen Dong, Scott Chilstedt, and Deming Chen
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
{cdong3, chilste1, dchen}@illinois.edu

Abstract

To maximize the potential of three-dimensional integrated circuit architectures, 3D CAD tools must be developed that are on-par with their 2D counterparts. In this paper, we present a statistical static timing analysis (SSTA) engine designed to deal with both the uncorrelated and correlated variations in 3D FPGAs. We consider the effects of intra-die and inter-die variation. Using the 3D physical design tool TPR as a base, we develop a new 3D routing algorithm which improves the average performance of two layer designs by over 22% and three layer designs by over 27%. To the best of our knowledge, this is the first physical design tool to consider variation in the routing and timing analysis of 3D FPGAs.

1. Introduction

In modern field programmable gate arrays (FPGAs), the majority of the chip area is devoted to the programmable interconnect used for the local and global routing of signals. As design complexities increase, signal paths become longer and have to connect across greater distances and through more programmable switches. The task of driving this capacitive interconnection fabric increases the critical path delay and power consumption of FPGA designs. One recognized solution to this problem is to move to a three-dimensional architecture, where layers of logic are stacked on top of each other instead of being spread across a 2D plane. Devices on each layer connect to devices on adjacent layers through the use of vias, increasing logic density and minimizing the average connection length.

There are three main strategies for manufacturing 3D integrated circuits: monolithic stacking, wafer based stacking, and die based stacking. In monolithic stacking, layers of logic are fabricated on top of existing layers of logic and interconnect. This is an ideal solution that allows layers to connect using minimally sized metal vias, but is difficult to achieve in practice because the heat required to manufacture transistors has the potential to destroy the metal routing in the layers below.

Another solution is wafer-based stacking, in which layers of transistors and wiring are fabricated on separate wafers. These wafers are bonded together to form a multilayer wafer, and then diced into 3D ICs. The problem with this scenario is that the yield decreases as the number of layers grows, since the failure of a die on any layer will render the final IC inoperable.

In die-based stacking, wafers are diced before bonding, allowing defective dies to be discarded. By using only known good dies, the 3D device yield can be maximized. Die-based stacking is especially attractive for homogeneous FPGA architectures, where logic tiles are replicated identically in each layer, allowing a single set of masks to be used.

To maximize the benefit of this new generation of integrated circuit design, 3D computer-aided design (CAD) tools must be developed that are on-par with their 2D counterparts. Current 3D tools use traditional static timing analysis, which assumes that all circuit elements have deterministic delay. To deal with variations, guard banding is employed to provide design margin. This is effectively a worst-case analysis which will satisfy a yield target, but does so at the price of performance. To achieve the optimal performance, 3D design tools should use variation-aware device models and calculate delay using statistical timing analysis (SSTA) techniques.

Variation comes from a number of sources and can be random in nature or correlated across multiple devices. Correlated variation can be considered across the devices on a die, across the dies on a wafer, or across wafers made with the same process. As technologies scale to increasingly small dimensions, the effects of variation are magnified because the size of the underlying atoms remains constant.

In this paper, we develop a new SSTA engine designed to deal with the uncorrelated and correlated variations in 3D FPGAs. We consider the effects of intra-die and inter-die variations to develop accurate timing models. Using the 3D placement and routing framework of TPR [1], we develop a new 3D routing algorithm which uses this engine as the basis for improving performance yield. To our knowledge, this is the first physical design tool to consider variation in the routing and timing analysis of 3D FPGAs.

We choose to address variation-aware routing instead of variation-aware placement and/or technology mapping because the most detailed timing and variation information is available during the routing phase. For instance, correlated variation between lookup tables is only known after the placement phase, when their locations have been fixed.

The rest of this paper is organized as follows: In Section 2, we review related work in 3D FPGAs and variation-aware FPGA routing. Section 3 discusses 3D architecture, including switch box and via density constraints. Our variation models for logic and interconnect are described in Section 4. We address the issues of applying statistical techniques to 3D CAD tools in Section 5, providing details on our variation-aware SSTA engine and new routing algorithm. Experimental results are presented in Section 6, and Section 7 concludes this paper.

2. Related Work

2.1. 3D FPGAs

A number of 3D FPGA architectures and CAD tools have been proposed in the literature. One of the first works to address the 3D FPGA placement and routing problem was [2], in which Alexander, et al. extended an iterated KMB algorithm into three dimensions. In another early work, Karro and Cohoon presented a simultaneous placement and global routing algorithm for 3D based on partitioning [3]. More recently, 3D routing tools were developed to characterize the performance of monolithically stacked 3D FPGAs by Lin, et al. in [4].

A 3D physical design engine called TPR was presented by Ababei, et al. in [1]. This engine is an extension of the popular VPR tool [5] into three dimensions, and the source code is freely available for academic use. Three placement algorithms were considered: a global min-cut partition to assign a netlist into layers, a timing driven intra-layer placement based on hMetis partitioning, and a 3D simulated annealing engine. TPR's routing tool is a Pathfinder negotiated congestion algorithm with added penalties to avoid vias.

Recently, Gayasen et al. addressed a number of 3D FPGA design issues in [6]. Area and critical path were compared for different combinations of layer number, bonding strategy, and via density. The routing resource utilization of a 5-layer stack was compared to an equivalent 2D stack and shown to be more efficient, assuming a 3µm via pitch. In addition, six potential 3D switchbox designs were evaluated. To analyze these designs, a timing driven placement and routing CAD flow was developed by extending VPR and adding a vertical channel congestion parameter into VPR's cost function.

While the progression of these works demonstrates a considerable evolution in the sophistication of 3D FPGA CAD, none of these tools addressed the significant impact of process variations on circuit performance.

2.2. Variation Aware Routing

On the 2D front, the statistical optimization of FPGA design tools has been receiving increasing attention. Previous works have demonstrated the particular effectiveness of such optimization during the physical design stage.

Sivaswamy and Bazargan presented a variation-aware routing algorithm in [7] that treats all sources of variation as spatially correlated, but ignores the effects of uncorrelated random variation. They conclude that while statistical optimizations are not currently as critical for FPGAs as they are for ASICs, process variations will become increasingly significant in future technologies, and statistical FPGA optimization techniques need to be explored.

In [8], Lin, et al. create a complete variation-aware physical synthesis flow that incorporates statistical clustering, statistical placement, and statistical routing. Since detailed routing information is not known during clustering, interconnect uncertainty is modeled as a random variable and used to characterize performance. For placement and routing, process variation is considered. When the effects of variations are considered during all three stages of physical synthesis, average yield improvements of 9.1% at a 95% yield, and 12.6% at a 90% yield are shown.

3. 3D FPGA Architecture

Like their 2-D counterparts, 3D FPGAs can adopt a traditional island-style FPGA architecture. This architecture contains a fabric of repeated tiles that consist of one switch block (SB), two connection blocks (CBs) and one configurable logic block (CLB). Fig. 1 illustrates two such tiles in a 3D stack. Connections are made between the layers using through-silicon vias (TSVs). For the architecture in this study, we only allow TSVs to connect in the SBs, as shown by the vertical lines in the figure.

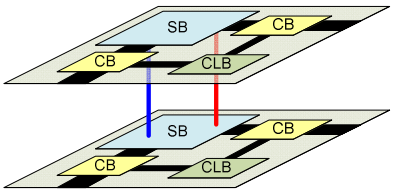

Figure 1. 3D Island based FPGA tiles

The 3D switch box is an important component in the 3D FPGA architecture which provides normal routing connections between the x and y horizontal routing channels, as well as vias to connect those channels vertically to additional layers in the 3D stack. Fig. 2 shows a possible 3D switch box design that demonstrates 3D switch point connections. For simplicity, switch points that only connect to the horizontal channels are not shown. The vertical vias are segmented to achieve electrical isolation between layers. This means that each switch box must contain independent connections for vias connecting to the upper and lower layers. This allows a signal from a horizontal wire to be directed to a specific layer instead of being sent both upwards and downwards. The pattern can be extended for any channel width by adding the appropriate number of switch points.

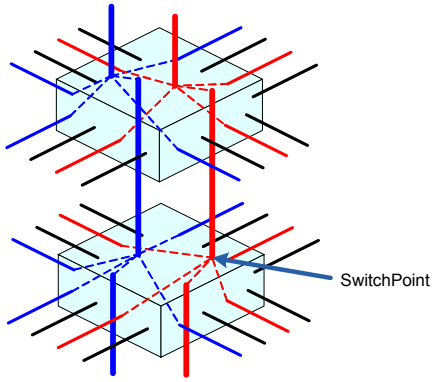

Figure 2. 3D subset switch box layout

In 3D circuit design, the vertical interconnects require special consideration. When two logic layers are connected in a face-to-face bonding process [9], their metallization layers are joined, and the size of the connecting vias is limited by the accuracy of the layer alignment technique used. For designs with more than two layers, face-to-back bonding is needed, which requires connections through the substrate. The properties of the connecting vias are determined by the substrate thickness. In a traditional bulk silicon process with a wafer thickness of 50–300 µm, through-silicon vias (TSVs) can be made with diameters as small as 10µm. However, if a silicon on insulator (SOI) process

is used, the substrate is reduced to a thickness of 1-50μm and vias can be made with diameters as low as 1μm [10].

In addition to via diameter, the fabrication method may also effect the via density. In the following calculations, we assume a 32nm process where the via pitch is only constrained by the switch point area. However, manufacturing limitations such as the substrate thickness and bonding technique will not scale in the same ratio as lithography-driven feature sizes, so it is possible that these limitations will dictate the via pitch below the 32nm technology node. This is especially true for a face-to-back bulk silicon process which requires larger vias than SOI.

To study the relationship between switch block area, vias density, and horizontal interconnection length, we define the parameter Z_{frac} according to equation 1.

$$Z_{frac} = Z_{width} / \max(X_{width}, Y_{width}) \quad (1)$$

Z_{frac} represents the number of vertical routing vias (Z_{width}) compared to the maximum horizontal channel width, and can be thought of as the percentage of switch points in a switch box that have vertical connections. In this study, we evaluate architectures with Z_{frac} = 20% and Z_{frac} = 30%.

Based on the routing switch design in [11], we estimate that each switch point takes 150.4T, where T is the minimum width transistor area. With a 0.0451μm^2 transistor area at the 32nm technology node [12], this gives us a switch point area of 6.783μm^2. Assuming a via diameter in range of 1.5μm [10], we model the area of a via-containing switch point as 1.5× larger than a horizontal-only switch point.

By adding the CLB area to the area of the switch blocks and connection blocks, we estimate the size of an FPGA tile for Z_{frac} = 20% and Z_{frac} = 30%. Table 1 illustrates the trade off between via density and routing delay. The results show that while increasing the number of vias in a switch box gives our physical design tools greater routing flexibility, it increases the tile size and requires longer horizontal interconnections.

Table 1. Via density vs. interconnect length

Via Density	Tile Size	Length-1 Wire
Z_{frac} = 20%	244.58 μm^2	15.64μm
Z_{frac} = 30%	267.32 μm^2	16.35μm

4. Variation Modeling

To develop an accurate statistical timing analysis engine, we must first characterize the variation model of three-dimensional FPGAs, considering both correlated and uncorrelated variation sources. Uncorrelated random variations can be approximated as normally distributed random variables with mean μ and standard deviation σ. Sources of uncorrelated variation include the random concentration of doping atoms in transistor source and drain, and the irregularities in interconnect width, thickness, and spacing.

Correlated variations are used to represent the trend of certain process parameters to vary according to the location on the chip. To characterize spatial correlation, the overall containing area can be divided into grids where every device in a grid square is assumed to be perfectly correlated. The variation relationships between the resulting *n* grid squares can then be represented by an *n x n* correlation matrix. Such a matrix is needed for performing principle component analysis, as described in [13].

4.1. Interconnect Delay

Interconnection delay is random based on the geometrical variation of wire width, wire thickness, and spacing [14]. In this study, we assume interconnect variation is an independent random variable, and model it for wires and vias based on the following equation from [15].

$$\sigma = 0.3836 \times \exp(-0.1537h) \times \mu_0 \quad (2)$$

In equation 2, σ is the standard deviation, *h* is the size of the driving buffer, and μ_0 is the nominal wire delay. For the size 10 and size 5 buffers in our architecture, this gives us standard deviations of 8.2% and 17.8% of μ_0, respectively.

4.2. Logic Delay

For logic devices, we consider both random and spatially correlated variations and express delay according to equation 3.

$$D = D_{nom} + \Delta D_{intra\_spatial} + \Delta D_{intra\_rand} + \Delta D_{inter} \quad (3)$$

In this equation, D_{nom} is the nominal delay value, and is adjusted based on the variation from intra-die ($D_{intra\_spatial}$, D_{intra_rand}) and inter-die (D_{inter}) sources.

Intra-die Variation

Intra-die variation is the variation that causes device performance to vary across a single die. Depending on the source of the variation, intra-die variation can be correlated or uncorrelated. To model the correlated intra-die variation, we consider both gate length and oxide thickness. We assume that these contributing process parameters are independent, and create separate correlation matrices for each. Additional process parameters could be modeled with additional matrices. Unlike 2D chips which need only one correlation matrix per parameter, 3D FPGAs are made from a number of separate dies and need a correlation matrix on each layer for each parameter. In our experiments, we consider intra-die variation with a 10% random component and a 10% spatially correlated component, with a correlation distance of 1mm [15]. We set the size of each correlation matrix by dividing the dies into grids such that each grid square contains a certain number of FPGA architecture tiles.

Since there is a lack of published variation information from device manufactures, we generate example data to fill the correlation matrices using a method from Xiong et al. in [16]. This method ensures that the correlation matrices are positive-semi definite, a requirement for the principle component analysis we use in Section 5. In an actual production flow, measured correlation data could be used.

Inter-die Variation

Inter-die variation is the variation correlated between dies. The stacking process used determines if there will be any inter-die correlation. In a homogenous die-based stacking process where each die in the stack comes from the same wafer, inter-die variation can be considered. One way to do this is to create large correlation matrices for each parameter that define relationships between the grid squares on all of the layers. The size of these matrices can be calculated by equation 4.

$$M_{size} = (n_{grid\_squares} \times n_{layers})^2 \quad (4)$$

Where $n_{grid\_squares}$ is the number of grid squares in each layer and n_{layers} is the number of layers in the device.

If wafer-based stacking is used instead of die-based stacking, the 3D devices will be made up of dies from different wafers, so inter-die correlation will be zero and should not be considered. However, wafer-to-wafer variation can be considered in its place to account for manufacturing processes that vary in the same way across each wafer.

In our experiments, we assume a homogenous die-based stacking process and a grid size of one tile. Since there are a large number of tiles in our benchmarks, this gives us a large $n_{grid\_squares}$. Considering correlated inter-die variations under these conditions would require a large M_{size}. Since operations performed on such matrices would be prohibitively expensive, we use a simpler model that treats inter-die variation as normally distributed with a standard deviation of 10% of the mean. This corresponds to the 11% random inter-die variation seen in [17].

5. CAD Tools

5.1. 3D SSTA

Efficient timing analysis generally requires the use of independent random variables. For our 3D SSTA, we follow the work of Chang et al. in [13]. This method leverages principle component analysis (PCA) to determine circuit behavior under correlated variations, and is widely used in 2D SSTA. PCA is a statistical technique that allows the transformation of a set of correlated random variables into a new set of uncorrelated random variables known as principle components. We use PCA to transform each process parameter's correlation matrix into a set of principle components (PCs) at the start of our CAD flow.

To perform 3D SSTA, each node in the timing graph must be made to store variation in addition to nominal delay. In our timing graph, we add normally distributed random variation and $n \times p$ sets of PCs to each node, where n is the number of layers in the stack and p is the number of independently correlated process parameters. If each die is divided into m grid squares, there will be m PCs in each set. The maximum number of PCs is therefore $n \times p \times m$. Since many of these PCs will remain empty, we allocate memory for a set of PCs only when a non-zero value is stored.

Within a layer, statistic operations such as ADD and MAX are carried out by operating on the PCs for that layer. For example, consider two timing nodes: n1 and n2, that each contain a set of PCs for each layer: pc1 for layer 1 and pc2 for layer 2. If both n1 and n2 belong to layer 1, only pc1 is used during statistical operations. If n1 and n2 belong to layer 2, only the PCs in pc2 are used. When nodes n1 and n2 are located in two different layers, we have to consider the PCs from both layers.

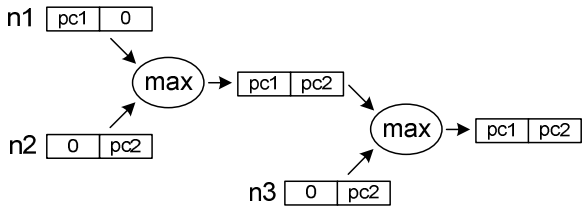

Figure 3. Multi-layer MAX operation

This is illustrated by the inter-layer MAX operation shown in Fig. 3, in which node n1 is on layer 1, and nodes n2 and n3 are on layer 2. Node n1 only has values for pc1, the spatially correlated variation within layer 1. Correspondingly, n2 only has PCs from layer 2. Therefore, the MAX operation of these two nodes must consider PCs representing intra-die correlations from both layers. To do this, each set of PCs in a node is statistically maxed with its counterpart in the other node. For example, pc1 set from n1 is maxed with the 0 from n2. Similarly, pc2 from n2 is maxed with the 0 in n1. The result then consists of PCs related to parameters on both layers. Using the same process, this result can then be maxed with additional nodes, such as n3. Note that this only accounts for intra-layer variation. We model inter-layer variation as an additional 10% random variation when performing inter-layer operations.

In a 3D architecture, nets can connect across multiple layers, spanning multiple spatially correlated variation domains. Using the techniques outlined above, the delay and slack calculations of our SSTA tool account for this by accumulating and combining sets of PCs for each of the correlated variation sources.

5.2. Variation Aware 3D Router

To perform variation aware routing, we modify the 3D FPGA placement and routing tool TPR [1]. TPR uses a routability-driven cost function focused on minimizing congestion. A routability-driven router is useful because the vertical connections of a 3D architecture increase the complexity and difficulty of achieving a legal routing solution. However, to achieve competitive performance, timing information must also be considered. The TPR router only uses static timing analysis to calculate the final critical path delay. We improve upon this baseline by adding a variation aware SSTA engine and incorporating timing information into the router's cost function. The pseudo code for this new router is shown in Fig. 4.

```
Compute correlation matrices;
Compute principle components and  generate routing, timing graphs.

Routing starts: set all nets i and sinks j, Crit(i,j) =1.0;
while (overused routing resources exist) do
    for (each net, i) do
        Rip-up routing tree of net i;
        for (each sink j of net i in decreasing Crit(i,j) order) do
            breadth first routing of sink j;
            for (all nodes in the path from i to j) do
                Update congestion;
            end
        end
    end
    Update historic congestion;
    Compute routing tree delay and variation.
    Update timing graph.
    SSTA compute mean and variation of arrival time and required time
                for each net(i);
    Update net Crit(i,j);
end
```

Figure 4. Pseudo-code of the modified TPR router

The first step in this algorithm is to capture spatial correlations in the form of correlation matrices. We assume there is no correlation between layers, so each layer has its own correlation matrix for each correlated process parameter. PCA is then used to generate the uncorrelated PCs for each node.

The rest of the routing is iterative. During the first iteration, the criticality of each pin in every net is set to 1 (the highest criticality) to minimize the wire length of each pin. After all of the nets are routed, SSTA is performed by traversing the updated timing graph to calculate the new slack and critical path delays. The criticality of pin j in net i is then computed, considering variation in the slack and critical path according to equation 5.

$$Crit(i,j) = 1 - \frac{slack(i,j) - 3\sigma_s(i,j)}{t_{crit} + 3\sigma_{crit}} \quad (5)$$

We derive the criticality function in this way so that when two slacks have a similar mean but different variations, the $slack(i,j) - 3\sigma_s(i,j)$ term assigns larger criticality to the path with the greatest variation, weighting it more heavily in the next iteration. This is illustrated in Figure 5, where the distribution with slack variation σ_1 will be assigned a higher criticality than the distribution with slack variation σ_2, even though they have the same mean.

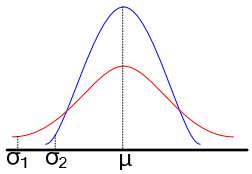

Figure 5. Criticality estimation

It is possible that some nets have positive slack but a large enough variation that the *slack* - $3\sigma_s$ term in equation 5 becomes zero or even negative. All of these nets are considered critical and will be assigned the maximum criticality.

Using this new criticality, we update the cost function used in TPR during maze routing. This function is used to select between multiple paths during breadth-first wave-front expansion. The function TPR uses to calculate the total cost, C_{total}, is shown in equation 6.

$$C_{total} = C_{path} + b(c)h(c)p(c) \quad (6)$$

In this equation, C_{path} is the path cost at the current routing node in the maze expansion wave front and b(c)h(c)p(c) is the congestion cost of the routing resource. The congestion cost consists of a base cost b(c), which is 1, 0.95, or 0, depending on the routing resource type, a historic congestion cost h(c), reflecting the overuse of this routing resource in the past, and a present congestion cost p(c), representing its current use.

Using the new criticality function in equation 5, we update this cost function by replacing the congestion cost by a timing and variation-aware routing resource cost, as shown in equation 7.

$$C_{total} = C_{path} + b(c) + \sqrt{(1 - crit_i)}\,h(c)p(c) \quad (7)$$

This differs from the original TPR cost function in that the base cost is separated from the congestion cost, and the congestion cost is scaled by $\sqrt{(1 - crit_i)}$. By adding the net criticality, $crit_i$, into equation 7, we allow nets with higher criticality to be less affected by congestion during maze expansion. The square root is used to preserve more of the congestion cost for lower criticality nets so that timing closure can also be achieved quickly. In this way, the impact of congestion on the critical path is greatly reduced, but the base cost remains the same.

In variation-aware routing, several nets are likely to be critical under statistical analysis. If each of these nets completely ignores congestion with a $crit_i$ = 1, overused resources could persist indefinitely, resulting in an unroutable situation. Therefore, we set the maximum net criticality value to be slightly less than 1 instead of using 1. This allows the congestion component to resolve such situations.

To further enhance the routing results, we set b(c) according to the statistical delay of the routing resources instead of using the base cost values suggested in [5] that do not consider timing information. This is shown in equation 8, where the delay of the routing resource node in question is rr_node_{mean} and rr_node_{var}.

$$b(c) = rr\_node_{mean} + 3 \times rr\_node_{var} \quad (8)$$

By considering timing in the base cost, b(c), and considering the base cost separately from the historic costs, a routing resource with a larger propagation delay will now have an appropriately larger cost. This differentiation of routing resources allows the new routability-driven router to select routing resources with lower propagation delay during maze expansion. As a result, our router achieves a large performance gain over the baseline routability-driven router used in TPR.

6. Experimental Results

6.1. Experiment Setup

In this experiment, we run simulations for 2-layer and 3-layer FPGA designs, using a standard set of 12 MCNC benchmarks. For each configuration, via densities of Z_{frac} = 20% and Z_{frac} = 30% are calculated to compare the impact of via density on routing results. Note that these techniques could also be scaled to a larger number of layers.

A traditional island style FPGA architecture is used, with size 4 lookup tables and a cluster size of 1. Delay of components such as LUTs and buffers are characterized in HSPICE using PTM 32nm models. We set a fixed routing channel width of 30 with buffered drivers. A mixture of length 1, length 2, and length 6 wires are used (wires spanning 1, 2 or 6 CLBs). Vias of both length 1 and 2 (crossing 1 or 2 layers) are used in the 3-layer case.

We use T-VPACK [5] for timing-driven packing of our benchmarks, and then place the mapped netlists using the partition based placement algorithm in TPR [1]. We use the same placement files to evaluate both variation aware and baseline TPR routing.

TPR baseline results are generated by the original TPR deterministic router. An SSTA is then performed on the routed circuits to estimate the resulting mean and variation. Variation-aware routing is based on the algorithm described in Section 5 and nets are routed using dynamically updated criticality.

6.2. Results and Discussion

Table 2 details the performance of the baseline TPR and variation aware routing results for 2 and 3-Layer FPGAs. Due to space limitations, only the average values for the 3-layer experiments are shown.

This table shows that for both 30% and 20% via density, the new variation aware router improves the guard banded μ+3σ performance of a 2-layer FPGA by over 22%, and a 3-layer FPGA by over 27%. Note that using a higher via density (30%) will increase the routability of the architecture, but comes at a performance penalty. This is due to the corresponding increase in horizontal wire length and variation, as described in Section 3.

If only nominal values are considered in the baseline TPR routing, increasing the 3D stacking to 3-layers shows a 13% and 9.1% reduction in the critical path delay for 20% and 30% via densities respectively. However, once we consider the variation of the resulting 3D architectures, this advantage is degraded to 7.0%

Table 2. Performance of two and three layer FPGA routing (ns)

Benchmark Circuit (2 Layer)	Via Density 20%							Via Density 30%						
	TPR Baseline			Variation Aware			μ+3σ Reduction	TPR Baseline			Variation Aware			μ+3σ Reduction
	μ	σ	μ+3σ	μ	σ	μ+3σ		μ	σ	μ+3σ	μ	σ	μ+3σ	
tseng	10.1	1.47	14.5	7.21	0.68	9.24	36.3%	10.5	1.52	15.1	7.13	0.71	9.25	38.8%
ex5p	7.09	0.99	10.1	5.75	0.47	7.15	29.0%	7.88	1.10	11.2	6.11	0.56	7.79	30.3%
diffeq	7.43	0.91	10.2	6.36	0.76	8.65	14.8%	6.66	1.07	9.86	6.26	0.75	8.51	13.7%
misex3	8.48	0.73	10.7	6.60	0.20	7.21	32.4%	8.51	0.84	11.0	7.38	0.42	8.65	21.6%
apex4	8.02	0.71	10.2	7.37	0.31	8.29	18.4%	8.23	0.71	10.3	7.25	0.43	8.55	17.4%
alu4	7.93	0.67	9.93	7.17	0.35	8.21	17.3%	8.75	0.77	11.1	7.11	0.40	8.32	24.8%
seq	7.96	0.77	10.3	7.31	0.40	8.51	17.1%	7.40	0.70	9.50	7.18	0.53	8.76	7.72%
apex2	9.48	1.28	13.3	6.94	0.24	7.66	42.5%	10.3	1.41	14.5	7.79	0.34	8.82	39.2%
dsip	4.65	0.51	6.17	4.55	0.47	5.97	3.29%	4.44	0.51	5.97	4.22	0.43	5.50	7.91%
des	7.22	0.82	9.67	6.14	0.66	8.12	16.0%	6.78	0.97	9.69	5.47	0.61	7.31	24.6%
s298	22.0	2.28	28.8	15.9	2.09	22.1	23.3%	24.3	2.74	32.5	16.3	2.01	22.3	31.4%
bigkey	4.77	0.45	6.12	3.45	0.44	4.76	22.2%	4.83	0.51	6.35	4.59	0.44	5.91	7.01%
2 Layer Average	8.07	0.87	10.7	6.62	0.48	8.21	22.7%	8.19	0.95	11.1	6.81	0.56	8.56	22.0%
3 Layer Average	7.02	0.96	9.97	5.45	0.48	6.92	28.5%	7.44	1.02	10.6	5.75	0.55	7.45	27.4%

and 4.4% respectively. This is because the stacking of an additional layer increases the amount of inter-die variation.

7. Conclusion

In this paper, we developed a new SSTA engine designed specifically to deal with the variations in 3D FPGAs. We considered the effects of intra-die and inter-die variation to develop accurate timing models. Using the 3D placement and routing framework of TPR [1], we developed new 3D routing algorithms which used this engine as the basis for improving performance. Our experimental results showed that variation aware routing improves the average performance of two layer FPGA designs by over 22% and three layer FPGA designs by over 27%. This demonstrates the importance of advanced timing driven and variation aware physical design tools for evaluating 3D architectures.

Acknowledgements

This work is partially supported by NSF Career Award CCF 07-46608, NSF grant CCF 07-02501, and a gift grant from Altera Corporation. We would also like to thank Prof. Kia Bazargan of the University of Minnesota and Prof. Cristinel Ababei of North Dakota State University for their help with TPR.

References

[1] C. Ababei, H. Mogal, and K. Bazargan, "Three-dimensional Place and Route for FPGAs," *IEEE Transactions on CAD*, Vol. 25, Issue 6, pp. 1132-1140, June 2006.

[2] M. Alexander, J. Cohoon, et al., "Placement and routing for three-dimensional FPGAs," *Proc. Canadian Workshop on Field-Programmable Devices*, 1996, pp. 11-18.

[3] J. Karro and J. Cohoon, "A Spiffy Tool for the Simultaneous Placement and Global Routing for Three-Dimensional Field-Programmable Gate Arrays," 9th *Great Lakes Symposium on VLSI*, pp. 226-227, March 1999.

[4] M. Lin , A. El Gamal, Y. C. Lu, and S. Wong "Performance benefits of monolithically stacked 3-D-FPGA," *Int'l. Symp. on FPGAs*, 2006, p. 113.

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

[6] A. Gayasen, N. Vijaykrishnan, M. Kandemir, A. Rahman, "Designing a 3-D FPGA: Switch Box Architecture and Thermal Issues," *IEEE Transactions on VLSI Systems*, vol.16, no.7, pp.882-893, July 2008.

[7] S. Sivaswamy and K. Bazargan, "Variation-aware routing for FPGAs," *Int'l. Symp. on FPGAs,* 2007, pp 71-79.

[8] Y. Lin, L. He, and M. Hutton, "Stochastic Physical Synthesis Considering Prerouting Interconnect Uncertainty and Process Variation for FPGAs," *IEEE Transactions on VLSI Systems,* vol.16, no.2, pp.124-133, Feb. 2008.

[9] S. J. Koester, et al., "Wafer-Level 3D Integration Technology," *IBM J. Res. & Dev.*, vol. 52, No. 6, Nov. 2008.

[10] J. U. Knickerbocker et al., "Three-dimensional silicon integration," *IBM J. Res. & Dev.*, vol. 52, No. 6, Nov. 2008.

[11] G. Lemieux and D. Lewis, "Circuit Design of FPGA Routing Switches," *Int'l. Symp. on FPGAs*, Feb. 2002, pp. 19-28.

[12] International Technology Roadmap for Semiconductors, http://www.itrs.net/.

[13] H. Chang, S. S. Sapatnekar, "Statistical Timing Analysis Considering Spatial Correlations using a Single PERT-like Traversal," *IEEE Int'l. Conference on CAD*, 2003.

[14] D. Boning and S. Nassif, "Models of Process Variations in Device and Interconnect," *Design of High-Performance Microprocessor Circuits*, Chapter 6, 2000.

[15] G. Lucas, S. Cromar, and D. Chen, "FastYield: Variation-Aware, Layout-Driven Simultaneous Binding and Module Selection for Performance Yield Optimization," *ASP-DAC*, Jan. 2009, pp. 61-66.

[16] J. Xiong, V. Zolotov, and L. He, "Robust extraction of spatial correlation," *ISPD*, 2007, pp. 2–9.

[17] R. Rao, et al., "Statistical Estimation of Leakage Current Considering Inter- and Intra-Die Process Variation," *Int'l. Symp. on Low Power Design*, 2003, pp. 84-89.