Three-dimensional Place and Route for FPGAs

Cristinel Ababei Hushrav Mogal Kia Bazargan

Department of Electrical and Computer Engineering, University of Minnesota,
200 Union St. SE, Minneapolis MN 55455
{ababei, mhush, kia}@ece.umn.edu

Abstract – We present timing-driven partitioning and simulated annealing based placement algorithms together with a detailed routing tool for 3D FPGA integration. The circuit is first divided into layers with limited number of inter-layer vias, and then placed on individual layers, while minimizing the delay of critical paths. We use our tool as a platform to explore the potential benefits in terms of delay and wire-length that 3D technologies can offer for FPGA fabrics. Experimental results show on average a total decrease of 21% in wire-length and 24% in delay, can be achieved over traditional 2D chips, when five layers are used in 3D integration.

I. INTRODUCTION

One possible future enabler of Moore's law is 3D integration, and a number of successful projects have shown the viability of the technology [8], [9]. 3D integration can significantly reduce wire-lengths and therefore hence circuit delay. 3D integration can particularly be useful for FPGA fabrics. It can address problems pertaining to routing congestion, limited I/O connections, low resource utilization and long wire delays. Even though the idea of 3D integrated circuits is not new, recent technological advances have made it a viable alternative. However, there is a lack of efficient 3D CAD tools that can exploit the potential gains that 3D integration has to offer. Furthermore, a number of important issues – such as heat dissipation, thermal stress [15], and physical design considerations – remain to be addressed for some 3D architectures.

Apart from previous work on tools for 3D standard cell technology [6], [12], [16] there has also been previous work on CAD tools for 3D FPGA integration. Alexander et al. proposed 3D placement and routing algorithms [2] for their architecture in [1]. Their placement algorithm is partitioning-based followed by a simulated annealing based refinement for total interconnect length minimization. They reported savings of up to 23% and 14% in total interconnect length at the placement and routing level respectively. An improved version of the placement algorithm appears as Spiffy, which performs placement and global routing simultaneously [3].

Our goal in this work is to present an efficient placement and detailed routing tool for 3D FPGAs. Unlike previous works on 3D FPGA architecture and CAD tools, we investigate the effect of 3D integration on delay, in addition to wire-length because wire-length alone cannot be relied on as a metric for 3D integration benefits. Furthermore, apart from the commonly used single-segment architecture, we also study multi-segment architectures in the third dimension.

The main contribution of our work is as follows.

- **TPR:** We developed a partitioning-based placement and maze routing toolset called TPR (Three-dimensional Place and Route). Its purpose is to serve the research community in predicting and exploring potential gains that the 3D technologies for FPGAs have to offer (similar to the role VPR played in the development of FPGA physical design algorithms). It shall be used as a platform, which can be used for further development and implementation of new ideas in placement and routing for 3D FPGAs.
- **SA-TPR:** In addition to the partitioning-based 3D placement tool, we have also developed a Simulated Annealing based version of TPR (called SA-TPR) to provide speed / quality tradeoffs.
- **Hybrid**: to provide more points on the runtime / quality tradeoff, we have developed a hybrid approach that uses partitioning to assign sub-circuits to different levels, and then uses simulate annealing to place individual cells within each layer.
- **Experiments**: we report the results of our tools on multi-segment routing in the third dimension. Furthermore, we model and report delay of placed and routed circuits. To the best of our knowledge, we are the first group to provide these features.

II. PARTITIONING BASED PLACEMENT ALGORITHM

A. *Overview of TPR*

The philosophy of our tool closely follows that of its 2D counterpart, VPR [4], [17]. The flow of the TPR placement and routing CAD tool is shown in Fig. 1. The design flow starts with a technology-mapped netlist in .blif format. Then, the .blif netlist is converted into a netlist composed of more complex logic blocks with T-VPack [5]. The .net netlist as well as the architecture description file are the inputs to the placement algorithm.

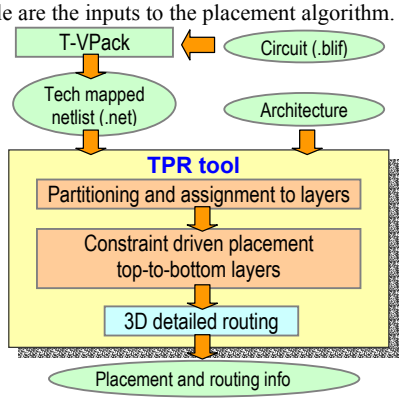

Figure 1 Flow diagram of TPR: 3D placement and routing tool

The placement algorithm first partitions the circuit into a number of balanced partitions equal to the number of layers for 3D integration. The top layer is placed by unconstrained recursive partitioning. The rest of the layers are then placed in turn by recursive partitioning, but constrained to reduce the delay on timing-critical nets: the terminals of the most critical nets, which span more than one layer, are placed on restricted placement

regions. A restricted placement region for a net n_i in layer l_j is defined as the smallest bounding box in layer l_j that encloses all projections of the terminals of net n_i which are placed in layers above l_j. Finally, global and detailed routing is performed using the adapted 3D version of the VPR routing algorithm.

B. Placement Algorithm

The simplified pseudo-code of the partitioning-based placement algorithm is shown in Fig. 2. The initial partitioning-into-layers step is performed using the min-cut hMetis partitioning algorithm [11]. This is motivated by the limitations imposed by current technologies, which require us to minimize the usage of vertical connections (it was also concluded in [7] that optimizing inter-layer interconnect is of key importance for 3D integration technologies).

```
Input:
       Tech mapped netlist .net G(V,E)
       Architecture description file
Algorithm:
1.  Initial min-cut partitioning into layers for
    via minimization
2.  For all layers i=0 to L-1 from top to bottom
3.      Partitioning based placement of layer i
4.          Update timing slacks
5.          Re-enumerate critical paths
6.      Greedy overlap removal
7.      Constraint generation for layers below
    (only for critical nets)
8.  Write .p placement output file
```

Figure 2 Pseudo-code of TPR placement algorithm

After the initial partitioning into layers we assign blocks (*i.e.*, partitions) to layers using a linear placement technique. The goal of this step is to minimize both the total (vertical) wire-length and maximum cut between any two adjacent layers. For example, in Fig. 3, we would like to assign the five blocks (as a result of the initial 5-way min-cut partitioning) to layers of the 3D architecture as in the case labeled "Good" rather than in the case labeled "Bad". That is because the good layer assignment minimizes both the total wire-length and maximum cut between adjacent layers.

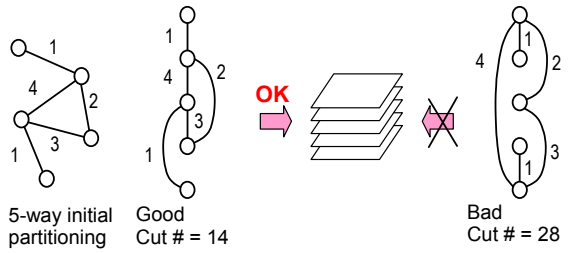

Figure 3 Illustration of good and bad initial linear placement of partitions into layers

We note that a solution for the layer assignment problem may not be optimal in terms of both objectives of wire-length and maximum cut between adjacent layers. Therefore, for this step, we use an efficient heuristic [19], which is able to find solutions with very good trade-off between wire-length and maximum cut. This technique is briefly described in what follows using the graph example shown in Fig. 5. First, we build the *EV-matrix*, which is an $m \times n$ matrix where m – the number of rows – is the number of edges in the graph and n – the number of columns – is the number of nodes. An element $a(i, j)=1$ in the matrix is non-zero if the *j-th* node is a terminal of the *i-th* net. If a node is not a terminal for a net, the corresponding *EV-matrix* element is zero. The order of the columns determines the order of the nodes in a linear placement. To minimize wire length and cut cost of the linear placement, the *EV-matrix* is transformed into a band matrix with the goal of minimizing the width of the band. This problem is denoted as *B(EV-matrix)-min* problem. The procedure to solve this problem uses row and column exchanges and is based on a sorting algorithm. The goal of getting the matrix to a band-form (which translates into a best linear ordering) serves two objectives:

1. Cutsize minimization – by having all 1's in the matrix clustered along the main diagonal, the cutsize (the number of nets cut by a vertical cut applied between any two consecutive nodes in the linear arrangement) is minimized everywhere in the linear arrangement.

2. WL minimization – by minimizing the width of the band (maximum distance spanned by any of the nets) of the *EV-matrix*, the total wire-length of all nets is minimized.

The pseudo-code of the procedure used for *EV-matrix* band-width minimization is shown in Fig. 4, and for example in Step 2, the *"Left"* array would be *{3,6,4,6,6}* for the example of Fig. 5. Sorting this array requires swapping the second and third elements, which translates into swapping second and third rows of *EV-matrix*.

```
Input:
       G(V,E)
Algorithm:
1.  Build EV-matrix
2.  Build "Left" array of indices of right-most
    non-zero elements. Sort array swapping rows
3.  Build "Top" array of indices of bottom-most
    non-zero elements. Sort array swapping
    columns
4.  Build "Right" array of indices of left-most
    non-zero elements. Sort array swapping rows
5.  Build "Bottom" array of indices of top-most
    non-zero elements. Sort array swapping
    columns
```

Figure 4 Pseudo-code of routine used for minimization of *EV-matrix* band

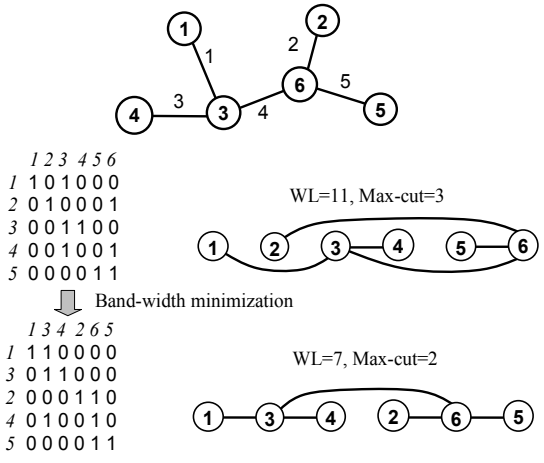

Figure 5 Illustration of good and bad initial linear placement of partitions into layers

After the initial layer assignment, placement is performed on each layer starting with the top layer (layer *0*) and continuing downwards till the last layer (layer *L-1*). The placement of every layer is based on edge-weighted quad-partitioning using the hMetis partitioning algorithm, and is similar to the approach in [13], which has the same quality as VPR but at 3-4 times shorter run times. Edge weights are computed inversely proportional to the timing slack of the corresponding nets. However, we also

selectively bias weights of the most critical nets. The set of critical nets is comprised of edges on the current *k*-most critical paths. The placement algorithm has an integrated static timing analysis engine as well as a path enumeration algorithm [14]. The delay of the circuit (and therefore slacks) and the set of the most critical paths are periodically updated based on the delay assigned to all current cut nets by the partitioning engine. This ensures accurate estimation of the circuit delay as the placement algorithm progresses. The rate of delay update and critical paths re-enumeration is dictated by the runtime / estimation accuracy trade-off.

The recursive partitioning of a given layer stops when each placement region has less than four blocks. Complete overlap removal is done using a greedy heuristic which moves non-critical blocks (*i.e.*, not on any critical paths) to the closest available empty location. When the placement of a layer is finished, we propagate placement constraints for the most critical nets. In layers that have net bounding box constraints, terminals that have placement restrictions are fixed in appropriate partitions before a call to the hMetis partitioning engine. This technique explicitly minimizes the 3D bounding-boxes of critical nets, which leads to minimization of the total wire-length and circuit delay. Steps 3 to 8 of the algorithm shown in Fig. 2 are performed for all layers, and when the last layer is finished the circuit is completely placed.

III. SA-TPR: SIMULATED ANNEALING BASED 3D PLACEMENT

In addition to the partitioning-based approach, we have also extended the simulated annealing based placement algorithm of VPR [4] to 3D (we call this engine SA-TPR, where SA stand for Simulated Annealing). As in VPR, our SA engine can place circuits with constraints of both wire-length and timing. SA-TPR can deliver better wire-length / delay quality at higher runtime costs compared to TPR.

Wire-length of a net is calculated as the weighted sum of its projected 2D bounding box and its vertical span. The weight on the vertical span is set to a high value to discourage usage of scarce vertical vias. The cost of a net e is described by the equation below.

$$Cost_{3D}(e) = q \cdot Cost_{2D}(e) + \alpha \cdot Span_z + \beta \cdot Num\_layers(e)$$

where q is a correction factor to 2D bounding box computation, which accounts for nets that have more than 3 terminals (the original VPR code uses this factor); $Cost_{2D}$ is the half-perimeter bounding box of the projection of all the terminals of the net; $Span_z$ is the vertical span of the net, and *Num_layers* is the number of layers on which terminals of net e are placed. Factors α and β are used to constrain the maximum length of vertical segments as well as the vertical channel density. To see the importance of using these factors, let us consider the two placements in Fig. 6.

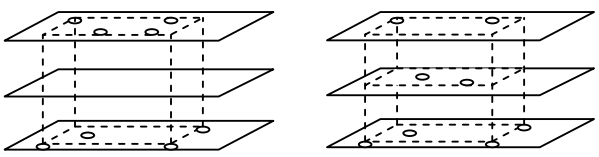

Figure 6 Two possible placements of the same net, showing different number of layers occupied

The two placement scenarios would be treated identically if we did not separately consider both the vertical span of a net, and the number of layers in which its terminals are placed. Each of these cost components are scaled by appropriate scaling factors: α, which discourages placing the terminals of a net far apart in the z dimension (otherwise the routing of the net would require longer vertical vias), and β, which restricts the number of vertical vias (vertical channel density is lower than the horizontal channel density and β reflects that ratio). In Fig. 6, the placement on the left is preferred to the one on the right, as it could potentially use only one vertical segment of length two to connect the terminals in different layers. But the placement on the right is likely to use more vertical routing resources.

Timing slack of a net determines its criticality weight. To compute the criticality of a net, the source-sink connection is projected onto 2D and its Δx and Δy separations in the 2D projection plane are calculated. Lookup tables are used to calculate the best-case 2D delay values, wherein unlimited routing resources are assumed. To accommodate a 3D structure, the separation of the connection in the third dimension is found and its delay is looked up using only one dimension of the delay tables (*i.e.*, a net that spans a distance of Δz in the vertical dimension, has the same delay as a 2D net with (Δz,0) bounding box).

The movement of cells in the third dimension is unrestricted in order to fully explore the vertical dimension. However, the annealing engine constrains movement in x and y directions more stringently as annealing proceeds (initially movement is allowed across the entire dimension of the chip and then gradually it is shrunk to neighboring CLB's).

IV. ROUTING ALGORITHM

Our 3D routing engine is shared by TPR and SA-TPR. The 3D FPGA architecture – described in the architecture file – is represented as a routing resource graph. Each node of the routing resource graph represents a wire (horizontal tracks in the x and y channels of all layers and vertical vias in the z channels) or a logic block (i.e., CLB) input or output pin. A directed edge represents a unidirectional switch (such as a tri-state buffer). A pair of directed edges represents a bi-directional switch (such as a pass transistor). An example of a routing resource graph construction is shown in Fig. 7.

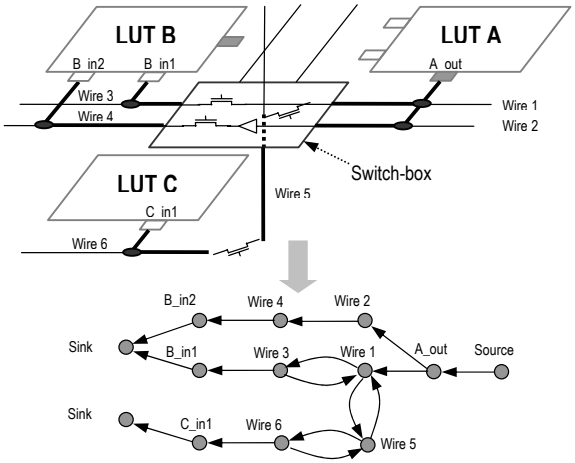

Figure 7 Illustration of the routing graph construction

TPR 3D detailed router is based on the Pathfinder negotiated congestion algorithm [18]. The routing is a rip-up and re-route iterative process, which routes every net by the shortest path using a breadth-first-search technique. The cost of overused routing

resources is gradually increased so that the algorithm forces nets with alternative routes to avoid overused routing resources, leaving behind only the net, which needs a given resource most. We add extra penalties to bends of a route created by a horizontal track and a vertical via as well as to vias themselves in order to discourage the routing engine to prefer vias and therefore to avoid a net placed totally in one layer to be routed using tracks in different layers. This will make, for example, the routing engine find the routing shown in Fig. 8.b rather than the routing solution shown in Fig. 8.a

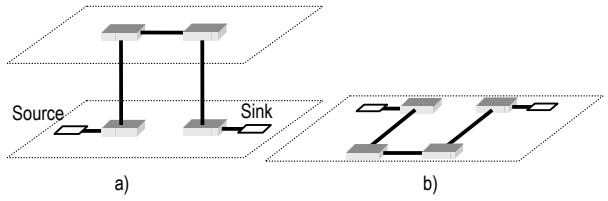

Figure 8 Illustration of two routings for a two terminal net

TPR router can find the minimum horizontal and vertical channel widths for which the circuit is fully routable. Vertical channel width starts with a value specified by the architecture file and is incremented every time when routing fails for a pre-determined number of different values for the horizontal channel width.

V. SIMULATION RESULTS

A. 3D Architectures

Our goal is to study the variation of the circuit delay and the total wire-length for a number of layers equal to five when the delay of an inter-layer wire (i.e., vertical via) has different values. We considered two different architectures: *Sing-Seg* and *Multi-Seg*. In both architectures, each plane has a routing architecture that resembles the Xilinx Virtex II architecture (they have wire segments of lengths 1, 2, 6, and long lines). However, *Sing-Seg* has vertical (inter-layer) vias of length one only, while *Multi-Seg* has vertical vias that span 1, 2, and all planes. Length one vertical segment is assumed to have the same delay and wire-length as 2D unit-length segments. This is a reasonable assumption, because 3D fabrication methods such as [8] can create inter-layer vias that are merely 5-10μm long. In such vertical segments, the switch delay dominates the delay of the segment, which is similar to the 2D case.

B. Experimental Results

We cannot compare our results to any of the previous works for a couple of reasons. First, our place and route tool is the first to report comprehensive results on wire-length and circuit delay as well as on all other metrics such as chip area, horizontal and vertical channel widths, and run-times on all twenty circuit benchmarks of the VPR package. We cannot compare to the only previous existing results reported in [2] because the authors of [2] used only six circuit benchmarks (unavailable to us) different from those we use (except *Apex2*). Moreover, the authors of [2] report only wire-length and minimum channel width results obtained for a very simple architecture, which only contains horizontal and vertical routing segments of length one. This is in contrast to our architectures, which have mixed – Virtex II-like – routing resources both horizontally and vertically.

We placed and detailed routed all circuits on 3D architectures with five layers. We recorded the average circuit delay and the average total wire-length of four different runs for each circuit. Results are presented in Tables 1 and 2. The *Average* row is the arithmetic mean of the values (in the HCW/VCW columns, only the HCW is averaged). The *ratio* row shows the ratio of the average values compared to TPR 2D (Table 1) and SA-TPR 2D (Table 2). Routing area is the total number of transistors used in all switchboxes in all layers (includes the third dimension switches). Footprint routing area of the 3D placements is the total routing area, divided by the number of layers (5 in our experiments). HCW (VCW) is the channel width in the planes (between layers).

We observe that when using the TPR algorithm, delay decreases on average by about 22% (24%) compared to the 2D case for *Sing-Seg* (*Multi-Seg*) architecture. In all cases, delay achieved using SA-TPR is smaller compared to TPR, which is not surprising, because annealing takes longer runtimes. When using SA-TPR, delay decreases by 19% (18%) compared to the 2D case placed by SA-TPR (which is the same as VPR). Note that these numbers are the *relative* improvements as a result of using a 3D architecture for the particular algorithm: either TPR or SA-TPR. We will compare all results in Table 3.

As shown in Tables 1 and 2, wire-length after detailed routing decreases by 21% and 10% on average using TPR and SA-TPR algorithms for both architectures (*Multi-Seg* and *Sing-Seg*). Wire-length is better minimized by SA-TPR. The smaller wire length results in smaller circuit delay. It can also have favorable impact on routing congestion (hence channel width), as well as power dissipation (especially because most of the power dissipated in FPGAs is due to interconnects, which account for more than 80% of the total area) as predicted by Rahman et al. [10].

Variations of the routing area and horizontal channel width are also presented in Tables 1 and 2. We observe that the overall area (i.e., chip foot-print area multiplied by the number of layers) slightly increases. This increase is due to the higher connectivity inside of a switch box (i.e., a track entering a 3D switch box will have to connect to 5 corresponding tracks as opposed to only 3 in the 2D case). Horizontal channel width decreases significantly in 3D placements.

Although not reported here (due to space limitations) we observed that, overall, run-times of SA-based placement are about twice the run-times of detailed routing and about an order of magnitude longer than run-times of partitioning-based placement. Therefore, partitioning-based placement can be used for efficient solution space exploration and different architectural feature exploration. The vertical channel widths, reported in Tables 1 and 2, are 1/3-1/4 of the horizontal channel widths, which demonstrates that our layer partitioning and linear placement as well as the routing algorithm are very well tuned to minimize the use of vertical tracks. Another advantage of using fewer vertical tracks greatly reduces the required area for switchboxes.

C. Experiments Using Mixed Partitioning- and SA- based Placement Algorithm (Hybrid)

We also implemented a mixed partitioning and simulated-annealing placement algorithm, called the *hybrid* algorithm. The reason for that is that the initial partitioning and assignment to layers does a very good job at minimizing the number of vertical vias. This technique combined with SA-based placement on each individual layer (under the restriction of not moving cells between layers) leads to high quality placements with minimum vertical connectivity. This strategy indeed leads to a decrease in wire-length whereas delay is virtually the same compared to full SA placement, which results in slightly smaller horizontal channel width (see Table 3). These results show that the quality of our layer partitioning and linear placement is very good. We can see that the hybrid algorithm leads to best results both in terms of

delay and wire-length but its run-time is the same as of SA-TPR and its area is slightly bigger.

VI. CONCLUSION

Benefits which 3D FPGA integration can offer were analyzed using a new placement and detailed routing tool. Placement can be done using either partitioning-based or simulated annealing based approach. Simulation experiments, after detailed routing, showed potential total decrease of 21% (10%) for wire-length and 24% (18%) for delay using the partitioning-based algorithm (or the SA-based algorithm). We observed that the Multi-Seg architecture shows slightly better delay characteristics compared to the Sing-Seg architecture.

TABLE I DELAY, WL, HORIZONTAL CHANNEL WIDTH (HCW) VERTICAL CHANNEL WIDTH (VCW), AND AREA AFTER SUCCESSFUL ROUTING USING TRP

Circuit	*TPR 2D*				*TPR 3D (Sing-Seg arch, five layers)*				*TPR 3D (Multi-Seg arch, five layers)*			
	Delay ($\times 10^{-7}$)	*WL*	*Routing area* ($\times 10^{+6}$)	*HCW*	*Delay* ($\times 10^{-7}$)	*WL*	*Routing area* ($\times 10^{+6}$)	*HCW / VCW*	*Delay* ($\times 10^{-7}$)	*WL*	*Routing area* ($\times 10^{+6}$)	*HCW / VCW*
Ex5p	1.14	38506	2.58	24	0.85	31816	2.94	18 / 6	0.88	31816	2.72	18 / 6
Apex4	1.19	43732	2.98	23	0.93	34970	3.31	18 / 6	0.93	34797	3.48	18 / 6
Misex3	1.10	47248	3.11	22	0.84	37821	4.11	18 / 6	0.78	37864	3.93	18 / 6
Alu4	1.27	40992	3.11	19	0.93	37578	3.67	16 / 5	0.88	37354	3.49	16 / 5
Des	1.13	88034	8.11	21	0.68	44224	4.43	18 / 5	0.69	44590	4.16	18 / 5
Seq	1.22	61906	4.25	25	0.91	48936	4.55	18 / 5	0.99	48936	4.28	18 / 5
Apex2	1.43	70415	4.66	25	1.04	55650	5.01	18 / 5	1.14	55650	4.70	18 / 5
Spla	1.93	164648	11.30	32	1.69	125773	11.20	22 / 5	1.64	125010	11.40	22 / 5
Pdc	2.76	220518	14.70	33	2.08	172274	15.70	23 / 5	1.90	172056	16.40	23 / 7
Ex1010	2.10	158444	10.70	24	1.56	143885	13.30	20 / 7	1.71	143885	12.50	20 / 4
Dsip	1.25	53157	5.62	19	0.65	34533	3.73	17 / 4	0.55	32074	4.25	19 / 5
Tseng	0.76	26412	2.14	19	0.74	22375	1.99	13 / 5	0.69	22328	1.97	12 / 5
Diffeq	1.13	40384	2.96	19	0.91	32786	3.17	13 / 5	0.98	32015	3.04	13 / 5
Bigkey	0.92	59786	5.07	18	0.63	45222	5.32	19 / 5	0.57	42172	4.38	17 / 5
S298	2.26	46767	2.96	16	2.11	45638	4.04	14 / 5	2.04	45916	4.38	17 / 5
Frisc	1.88	138419	9.03	27	1.87	99999	8.81	18 / 5	1.75	98698	9.24	19 / 5
Elliptic	1.98	119692	7.90	22	1.63	92207	8.56	18 / 5	1.58	92204	8.27	18 / 5
S38417	1.77	173171	12.90	20	1.59	158876	18.90	19 / 5	1.46	155657	15.90	18 / 5
S38584.1	1.95	207449	14.70	23	1.41	148967	14.90	15 / 5	1.44	148870	13.80	14 / 5
Clma	3.18	342074	22.10	28	2.45	281293	25.20	21 / 5	2.21	283350	25.40	23 / 5
Average	**1.61**	**107088**	**7.54**	**22.95**	**1.27**	**84741**	**8.14**	**17.8**	**1.24**	**84262**	**7.88**	**17.95**
Ratio	**1.00**	**1.00**	**1.00**	**1.00**	**0.78**	**0.79**	**1.07**	**0.77**	**0.76**	**0.78**	**1.04**	**0.78**

TABLE II DELAY, WL, HORIZONTAL CHANNEL WIDTH (HCW) VERTICAL CHANNEL WIDTH (VCW), AND AREA AFTER SUCCESSFUL ROUTING USING SA-TRP

Circuit	*SA-TPR 2D*				*SA-TPR 3D (Sing-Seg arch, five layers)*				*SA-TPR 3D (Multi-Seg arch, five layers)*			
	Delay ($\times 10^{-7}$)	*WL*	*Routing area* ($\times 10^{+6}$)	*HCW*	*Delay* ($\times 10^{-7}$)	*WL*	*Routing area* ($\times 10^{+6}$)	*HCW*	*Delay* ($\times 10^{-7}$)	*WL*	*Routing area* ($\times 10^{+6}$)	*HCW*
Ex5p	0.97	30319	2.16	20	0.77	26780	2.58	19 / 5	0.77	26989	2.50	19 / 5
Apex4	1.02	35108	2.58	20	0.96	32021	2.87	18 / 5	0.91	32124	2.77	18 / 5
Misex3	0.92	36634	2.76	19	0.85	34900	3.22	18 / 5	0.81	34455	3.12	18 / 5
Alu4	1.14	37335	2.68	18	0.89	33462	2.96	14 / 5	0.87	33760	2.82	14 / 5
Des	0.94	51893	5.90	16	0.58	38351	3.32	17 / 5	0.64	38619	3.08	16 / 5
Seq	0.95	46563	3.43	19	0.81	44541	4.08	19 / 5	0.86	45182	3.87	18 / 5
Apex2	1.13	51507	3.75	19	1.00	50472	4.37	18 / 5	0.89	50963	4.28	18 / 5
Spla	1.82	126218	8.91	26	1.44	108559	10.50	23 / 5	1.48	106293	9.60	21 / 5
Pdc	1.97	176412	12.30	29	1.78	149940	16.50	32 / 5	1.84	147254	13.60	26 / 5
Ex1010	2.23	123461	8.51	19	1.45	116726	10.20	18 / 5	1.48	116597	9.87	18 / 5
Dsip	1.11	29699	3.83	13	0.57	25581	2.55	13 / 5	0.55	25129	2.45	13 / 5
Tseng	0.98	131000	10.40	17	0.82	126000	12.80	16 / 5	0.84	129144	12.30	17 / 5
Diffeq	0.77	30898	2.16	15	0.75	29739	2.85	13 / 5	0.85	29549	2.74	13 / 5
Bigkey	1.17	36451	3.83	13	0.63	31909	3.17	13 / 5	0.60	32215	3.04	13 / 5
S298	1.81	39933	2.71	14	1.64	37893	3.57	14 / 5	1.59	37843	3.40	14 / 5
Frisc	1.72	105071	7.48	22	1.54	99120	8.72	19 / 5	1.47	99213	8.49	20 / 5
Elliptic	1.23	87174	7.14	19	1.33	87779	7.97	18 / 5	1.25	88105	7.70	18 / 5
S38417	1.09	136116	9.85	16	1.25	121736	11.30	14 / 5	1.38	128257	11.20	14 / 5
S38584.1	0.58	18822	1.46	13	0.60	19090	1.78	12 / 5	0.60	18969	1.78	13 / 5
Clma	2.38	260461	19.00	24	1.89	221749	19.90	19 / 5	1.75	223908	19.20	19 / 5
Average	**1.29**	**79554**	**6.04**	**18.55**	**1.07**	**71817**	**6.76**	**17.35**	**1.07**	**72228**	**6.39**	**17.00**
Ratio	**1.00**	**1.00**	**1.00**	**1.00**	**0.83**	**0.90**	**1.11**	**0.93**	**0.82**	**0.90**	**1.05**	**0.91**

TABLE III AVERAGE VALUES AND RATIOS RELATIVE TO SA-TRR 2D CASES

	Averages								*Ratio of averages (divided by SA-TPR 2D, i.e., VPR)*						
	SA-TPR 2D	*SA-TPR 3D Sing-Seg*	*SA-TPR 3D Multi-Seg*	*TPR 2D*	*TPR 3D Sing-Seg*	*TPR 3D Multi-Seg*	*Hybrid 3D Sing-Seg*	*Hybrid 3D Multi-Seg*	*SA-TPR 3D Sing-Seg*	*SA-TPR 3D Multi-Seg*	*TPR 2D*	*TPR 3D Sing-Seg*	*TPR 3D Multi-Seg*	*Hybrid 3D Sing-Seg*	*Hybrid 3D Multi-Seg*
Delay ($\times 10^{-7}$)	1.29	1.07	1.07	1.61	1.27	1.24	1.06	1.06	0.82	0.82	1.24	0.98	0.96	0.82	0.82
WL	79554	71817	72228	107089	84741	84262	68154	66798	0.90	0.90	1.34	1.06	1.05	0.85	0.83
Routing area ($\times 10^{+6}$)	6.04	6.76	6.39	7.54	8.14	7.88	7.16	6.83	1.11	1.05	1.24	1.34	1.30	1.18	1.13
HCW	18.55	17.35	17	22.95	17.8	17.95	15.81	15.35	0.93	0.91	1.23	0.95	0.96	0.85	0.82

ACKNOWLEDGEMENTS

This work was supported in part by DARPA under grant number N66001-04-1-8909.

REFERENCES

[1] A. J. Alexander, J. P. Cohoon, Jared L. Colflesh, J. Karro, and G. Robins, "Three-Dimensional Field-Programmable Gate Arrays", *Proc. Intl. ASIC Conf.*, pp. 253-256, 1995.

[2] A. J. Alexander, J. P. Cohoon, Jared L. Colflesh, J. Karro, E. L. Peters, and G. Robins, "Placement and Routing for Three-Dimensional FPGAs", *Fourth Canadian Workshop on Field-Programmable Devices*, pp. 11-18, 1996.

[3] J. Karro and J. P. Cohoon, "A spiffy tool for the simultaneous placement and global routing for three-dimensional field-programmable gate arrays", *Ninth Great Lakes Symposium on VLSI*, pp. 226-227, 1999.

[4] V. Betz and J. Rose, "VPR: A New Packing Placement and Routing Tool for FPGA Research", *Field-Programmable Logic App.*, pp. 213-222, 1997.

[5] A. Marquardt, V. Betz, J. Rose, "Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density", *FPGA*, pp. 37-46, 1999.

[6] S. Das, A. Chandrakasan, and R. Reif, "Design Tools for 3-D Integrated Circuits", *Proc. ACM/IEEE ASP-DAC*, 2003.

[7] S. Das, A. Chandrakasan, and R. Reif, "Three-Dimensional Integrated Circuits: Performance Design Methodology and CAD Tools", *Proc. ACM/IEEE ISVLSI,* 2003.

[8] R. Reif, A. Fan, K. - N. Chen, and S. Das, "Fabrication Technologies for Three-Dimensional Integrated Circuits", *Proc. International Symposium on Quality Electronic Design (ISQD)*, 2002.

[9] K. W. Lee, T. Nakamura, T. Ono, Y. Yamada, , T. Mizukusa, H. Hashimoto, K. T. Park, H. Kurino, and M. Koyanagi, "Three-dimensional shared memory fabricated using wafer stacking technology", in *Technical Digest of the International Electron Devices Meeting*, pp. 165-168, 2000.

[10] A. Rahman, S. Das, A. Chandrakasan, and R. Reif, "Wiring Requirement and Three-Dimensional Integration of Field-Programmable Gate Arrays", *Proc. ACM/IEEE SLIP*, 2001.

[11] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multi-level Hypergraph Partitioning: Applications in VLSI Design", *Proc. ACM/IEEE DAC*, pp. 526-529, 1997.

[12] Y. Deng and W. P. Maly, "Interconnect Characteristics of 2.5-D System Integration Scheme", *Proc. ACM/IEEE ISPD*, pp. 171-175, 2001.

[13] P. Maidee, C. Ababei and K. Bazargan, "Fast Timing-driven Partitioning-based Placement for Island Style FPGAs", *Proc. ACM/IEEE DAC*, pp. 598-603, 2003.

[14] Y-C. Ju, R.A. Saleh, "Incremental Techniques for the Identification of Statically Sensitizable Critical Paths", *Proc. ACM/IEEE DAC*, 1991.

[15] B. Goplen and S. Sapatnekar, "Efficient Thermal Placement of Standard Cells in 3D ICs using a Force Directed Approach", *Proc. ACM/IEEE ICCAD*, pp. 86-89, 2003.

[16] S. T. Obenaus and T. H. Szymanski, "Gravity: Fast Placement for 3-D VLSI", *ACM Trans. on Design Automation of Electronic Systems (TODAES)*, Vol. 8, No. 3, pp. 298-315, July 2003.

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

[18] C. Ebeling, L. McMurchie, S. A. Hauck, and S. Burns, "Placement and Routing Tools for the Trptych FPGA", *IEEE Trans. VLSI Systems*, Vol. 3, No. 4, pp. 472-483, Dec. 1995.

[19] C. Ababei and K. Bazargan, "Non-contiguous Linear Placement for Reconfigurable Fabrics", *Proc. Reconfigurable Architectures Workshop (RAW)*, 2004.