Exploring Potential Benefits of 3D FPGA Integration

ABSTRACT

A new timing-driven partitioning-based placement tool for 3D FPGA integration is presented. The circuit is first divided into layers with limited number of inter-layer vias, and then placement is performed on individual layers, while minimizing the delay of critical paths. We use our tool, which will be available on the web for the research community, as a platform for exploring potential benefits in terms of delay and wire-length that 3D technologies have to offer for FPGA fabrics. We show that 3D integration results in wire-length reduction for FPGA designs. However, unlike the ASIC case, wire-length reduction does not automatically translate to much smaller circuit delays, unless multi-segment lengths are employed between layers. Our empirical analysis shows that wire-length can be reduced by up to 50% (20% on average) using 5 layers. Delay reductions are estimated to be up to 30% (15% on average) using the same number of layers.

1. INTRODUCTION

Smaller feature size and increasing transistor counts allow implementation of more complex and larger designs. However, a number of new design problems emerge and old problems become more difficult to solve. For example, global wires dominate the delay and power budgets of circuits, and signal integrity, IR-drops, process variations, and high temperature gradients pose new difficult design problems. Furthermore, shrinking time-to-market windows and ever-increasing mask costs have reduced profit margins alarmingly.

In response to mounting problems of the integrated circuit technology, various research groups have shown renewed interest in 3D IC integration, and a number of successful projects have shown the viability of the technology [15]-[21]. 3D integration can significantly reduce wire-lengths (and hence circuit delay), and boost yield[1]. Furthermore, there has been an inclination towards employing IP-based design and structured gate arrays (*e.g.*, FPGA blocks) to partially solve complex signal integrity and noise issues. 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.

On the standard cell arena, Das *et al.* recently proposed a placement and global routing tool as well as a 3D layout editor [13]. The placement algorithm is based on recursive min-cut partitioning of the circuit represented as a hypergraph and follows the same idea as in the Capo placer [23]. Interlayer via minimization is sought by min-cut partitioning for layer assignment. Wire-length minimization is done by considering aspect ratio during the partitioning. The user can select either hMetis [25] or PaToH [26] as the partitioning algorithm. Their global routing algorithm is a concurrent approach based on the idea in [24]. It was shown that 28% (51%) wire-length improvement could be obtained with two (five) layers, compared to [31] (the improvement is only 7% (17%) when inter-layer via minimization is the main objective). Wire-length reduction of up to 74% was reported in [33]. Deng and Maly showed – using a placement algorithm based on Capo – that the total wire-length can be reduced by 16% compared to flat placement, when two-layer integration is used [27]. It is important to note that current technologies allow for CMP substrate thinning down to about 5-10μm, hence allowing for multiple thin active device layers and interconnect levels be stacked on top of each other, resulting in short inter-layer vias with small aspect ratios [15].

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[2], and physical design considerations – remain to be addressed for some 3D architectures.

There has been some previous work proposing 3D FPGA architectures. Borrowing ideas from multi-chip module (MCM) techniques, Alexander *et al.* proposed to build a 3D FPGA by stacking together a number of 2D FPGA bare dies [1]. Electrical contacts between different dies would be made using solder bumps or vias passing through the die. The number of solder bumps that can fit on a die determines the width and separation of vertical channels between FPGA layers. Depreitere *et al.* proposed using optical interconnects to construct a multi-layer FPGA [2]. An straightforward extension of a 2D architecture [6] is found in the Rothko 3D architecture, which has routing-and-logic blocks (RLBs) placed in more than one layer [5]. Fine-grained interlayer connections were added outside each RLB, providing connections between cells above and below, using a specially designed technique [8], [9]. An improved version of Rothko architecture – which advocates placing the routing in one layer and logic on another for more efficient layer utilization – appears in [7]. It was shown that the percentage of routed connections increases with an increase in the flexibility of switch boxes. Also, computational density is higher compared to a 2D architecture. Universal switch blocks for 3D FPGA design were analyzed in [34]. It is important to point out that all of these works assume that the inter-layer connectivity is provided by vertical wire segments that connect each layer to its adjacent layers only.

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

[1] True only in technologies that fabricate different layers separately, weed out the faulty layers from the layer wafers, and only integrate the working ones. However, wafer stacking technologies have been shown to be more practical and successful compared to techniques that "grow" different layers on top of each other in a single process.

[2] There is already previous work addressing thermal issues in physical design for 3D integration [32].

In the experimental methodology presented in [7], placement was performed with VPR [10] and routing was performed with a custom routing tool [12].

In this paper we present a fast placement tool for 3D FPGAs called TPR (Three dimensional Place and Route). Unlike previous works on 3D FPGA architecture and CAD tools, we investigate the effect of 3D integration on delay, in addition to wire-length. We show that wire-length alone cannot be relied on as a metric for 3D integration benefits.

Our placement algorithm is partitioning-based, and hence scalable in the face of explosive growth of design sizes. A circuit is first partitioned for min-cut minimization into a number of partitions equal to the number of layers for the 3D integration. Then, timing-driven partitioning-based placement is performed on every layer starting with the top layer and continuing downwards. Allowable bounding box for nets on a particular layer is decided by the layers above it, to minimize the 3D bounding-boxes of the most critical nets. Constraints for any given layer are set by the placement on layers above. The routing algorithm is currently being imported and adapted for the 3D architecture from the leading academic placement and routing tool for 2D architectures, VPR [10]. The main contribution of our work is as follows.

- We analyze the potential benefits, which can be obtained by 3D integration for FPGAs. More specifically, we place circuits onto 3D FPGA architectures and study the variation in circuit delay and total wire-length compared to their 2D counterparts, under different 3D architectural assumptions. The results of this study and similar studies in future could guide researchers in designing high performance 3D FPGA fabric architectures.
- We developed a tool, which will be available as source and executable on the web. 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.

2. OVERVIEW OF TPR

The philosophy of our tool closely follows that of its 2D counterpart, VPR [10]. In fact, much of the code related the parsing input files as well as the 2D FPGA architecture and routing definitions is imported and adapted from VPR. 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, which can be generated using SIS [28]. Then, the .blif netlist is converted into a netlist composed of more complex logic blocks with T-VPack [11]. The .net netlist as well as the architecture description file are the inputs to the placement algorithm. The placement algorithm first partitions the circuit into a number of balanced partitions equal to the number of layers for the 3D integration. The goal of this first min-cut partitioning is to minimize the connections between layers, which translates into minimum number of vertical (i.e., inter-layer) wires. The reason is that in 3D technologies, the architecture is not isotropic (i.e., vertical vias are not as dense – because of their higher pitch) and thus the placement and routing tools must judiciously use vertical routing resources. After dividing the netlist into layers, TPR continues with the placement of each layer in a top-down fashion.

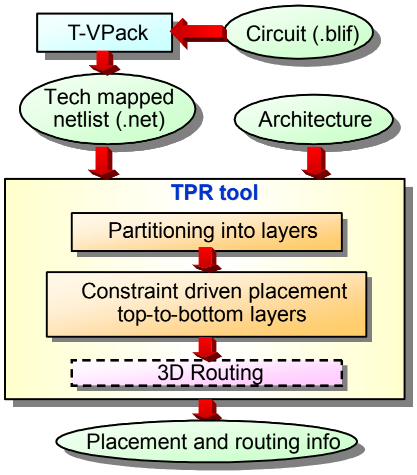
T-VPack
Circuit (.blif)
Tech mapped netlist (.net)
Architecture
TPR tool
Partitioning into layers
Constraint driven placement top-to-bottom layers
3D Routing
Placement and routing info

Fig. 1 Flow diagram of the 3D placement and routing tool

The top layer is placed by unconstrained recursive partitioning. Then, the rest of the layers are 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 layers, are placed on restricted placement-regions. The restricted placement-regions are defined by the projection onto the current layer of the bounding-boxes defined by the terminals placed already in the layers above. Hence, the 3D bounding-box of the critical nets is minimized[3]. Finally, global and detailed routing is to be performed using the adapted 3D version of the VPR routing algorithm.

3. PLACEMENT ALGORITHM

The placement algorithm is primarily partitioning based. However, it also has integrated a simulated annealing engine (SA), which can be used for further refinement and improvement of the solution quality similarly to VPR, but at the expense of run-time increase. The simplified pseudo-code of the placement algorithm is shown in Fig. 2.

The initial "partitioning into layers" step is performed using the min-cut hMetis partitioning algorithm [25] and is further illustrated in Fig. 3. This is motivated by the limitations imposed by current technologies, which require us to minimize the usage of vertical connections (it was also observed in [14] that optimizing inter-layer interconnect is of key importance for 3D integration technologies).

[3] This is similar to the 2D terminal alignment proposed by Maidee *et al.* in [29].

```
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.      Do partitioning based placement of layer i
4.          Update timing slacks
5.          Re-enumerate critical paths
6.      Low temperature SA based refinement
7.      Greedy overlap removal
8.      Constraint generation for layers below (only for
        critical nets)
9.  Write .p placement output file
```

Fig. 2 Pseudo-code of the TPR placement algorithm

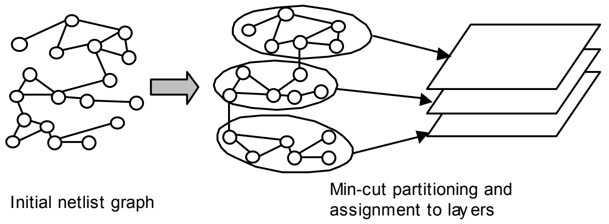

Fig. 3 Initial min-cut partitioning for minimization of the number of vertical wires

After the initial partitioning into layers we assign partitions to layers using a linear placement approach. The goal of this step is not only to minimize the total vertical wire-length but also the maximum cut between any two consecutive layers. Further illustration of this step is shown in Fig. 4.

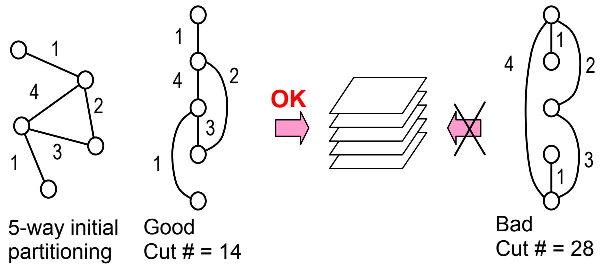

Fig. 4 Illustration of good and bad initial linear placement of partitions into layers

The actual placement is performed on each layer individually 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-weighting quad-partitioning using hMetis partitioning algorithm. Edge weights are commonly computed inversely proportional to the slack of nets. However, we also selectively bias weights of the most critical nets. The set of critical nets contains 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 [30]. The delay of the circuit (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 runtime / estimation accuracy trade-off.

The recursive partitioning of a given layer stops when each placement region has less than four blocks. A low-temperature SA can then be applied for further solution improvement as well as for partial overlap removal. 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 given layer is finished, we forward propagate placement constraints for the most critical nets. For example, assuming that layer k_1 is the first layer (from the top, as shown in Fig. 5) in which some terminals of a critical net are placed, the net bounding box of the net is projected to lower layers k_1+1 through L as a placement constraint for the rest of the terminals of the net in these layers.

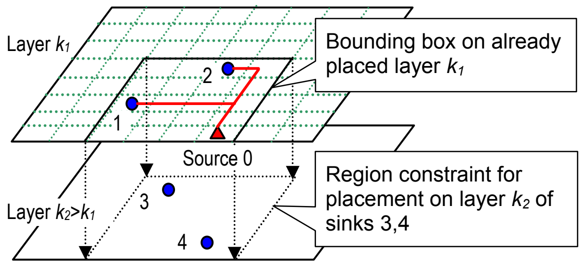
Layer k_1
Bounding box on already placed layer k_1
2
1
Source 0
Layer $k_2>k_1$
3
4
Region constraint for placement on layer k_2 of sinks 3,4

Fig. 5 Illustration of the restricted placement-region (i.e., projection of the bounding box of terminals 0, 1, and 2 placed on layer k_1) acting as a region constraint for sinks 3, 4 of a critical net, which has the source and sinks 1, 2 placed on a layer above

In layers that have net bounding box constraints, terminals that have placement restrictions are fixed in appropriate partitions before the call to the hMetis partitioning engine is made. 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.

4. ANALYSIS OF PLACEMENT DELAY ESTIMATION

We analyze the relation between the circuit delay, which we estimate after placement and the circuit delay computed after detailed routing in the 2D case. In next section, we perform experiments on 3D placements. The goal of this 2D analysis is to study the reliability of the delay estimation during placement (which in turn affects slack calculations and the decision on which nets to tag as critical). To this end, we placed a set of ten combinational circuits – shown in Table 1, which come with the VPR package – using our TPR placement algorithm on a single layer and recorded the total wire-length and the circuit delay using our estimation[4]. Then, all placements are routed using the timing-driven VPR routing algorithm [11], after which again circuit delay and total wire-lengths are recorded.

Table 1 Statistics of simulated circuits

Circuit	No. CLBs	No. IOs
Ex5p	1064	71
Apex4	1262	28
Misex3	1397	28
Alu4	1522	22
Des	1591	501
Seq	1750	76
Apex2	1878	42
Spla	3690	62
Pdc	4575	56
Ex1010	4598	20

The correlations between the delay and the wire-length estimation after placement and the delay and the wire-length after detailed routing are shown in Fig. 6 .

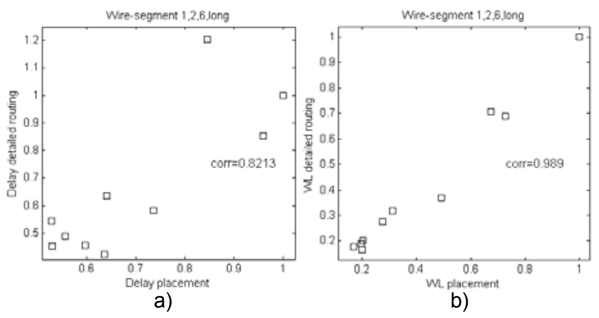

Fig. 6 Correlation between normalized delay (a) and normalized wire-length (b) after placement and after routing

Plot (a) in Fig 6 corresponds to delay, and plot (b) shows wire length correlations. Each point in the plots of Fig. 6 corresponds to one circuit from Table 1. A point with *(x,y)* coordinates indicates that for that circuit, delay (wire-length) estimation at placement was *x*, while the delay (wire-length) after detailed routing turned out to be *y*. The closer *x* and *y*, the more reliable the delay (wire-length) estimation is at the placement level. The plot in Fig. 6.a describes the relation between the normalized delay estimated after placement and the normalized delay computed after detailed routing. The routing architecture has wire segments of length one, two and six as well as long wires. The plot in Fig. 6.b shows the correlation between the normalized total wire-length after placement (computed based on half-perimeter of bounding-boxes) and the normalized total wire-length after detailed routing. We note that there is a good correlation in all cases, which tells us that comparing placed circuits based on our estimations should be representative of the results after routing. Our 3D routing tool is yet under development. Preliminary simulations with a first version of it demonstrate plots similar to those in Fig. 6, for a number of layers greater than two.

5. SIMULATION RESULTS

The goal of our simulations in this Section is to study the variation of the circuit delay and the total wire-length with the number of layers when the delay of an inter-layer wire (i.e., vertical via) has different values. The TPR implementation contains most of the VPR code (except the graphics), which was mostly ported to C++, as well as additional routines for the new partitioning-based placement algorithm, path enumeration engine, terminal propagation techniques, etc. The code was compiled with g++ on a Linux box running on a P4 – 2.54GHz - with 1G of memory.

In the first set of simulations we assume that inter-layer vias are as long as single-length wire segments (i.e., the distance between two CLBs). This is a reasonable assumption, because 3D fabrication methods such as [15] can create inter-layer vias that are a mere 5-10μm long. The delay of the inter-layer via is assumed to be equal to the delay of a segment of length one as well. We placed all ten circuits of Table 1 on different number of layers and recorded the average circuit delay and the average total wire-length of three placement runs. The simulation results are shown in Fig. 7 and Fig. 8, where circle points represent the average of the ten circuits and square points represent the minimum and the maximum among the ten circuits of the delay and wire-length, respectively. It can be seen that wire-length decreases are significant (and are similar to the ASIC results from other researchers). However, delay does not decrease as much and it actually increases with increasing number of layers greater than three. This shows that a simplistic vertical routing architecture is not going to benefit 3D FPGA architectures.

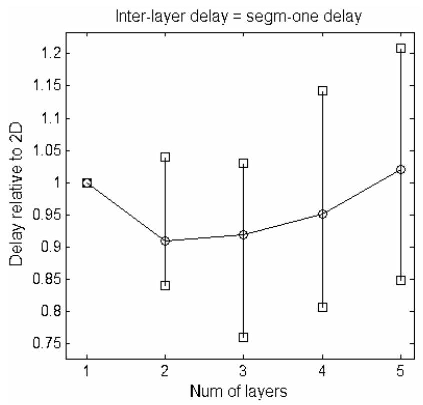

Fig. 7 Circuit delay as a function of the number of layers. The delay of an inter-layer connection that connects terminals that are *k* layers apart is *k* times the delay of a segment of length one

[4] Our rather optimistic delay estimation during and after placement is based on a matrix lookup table of best delays – as reported by VPR routing algorithm – which can be achieved for a route between two generic points say *(x_1,y_1)* and *(x_2,y_2)*. Then, the entry in the matrix for the route delay is *$(i,j) = (|x_1-x_2|,|y_1-y_2|)$*.

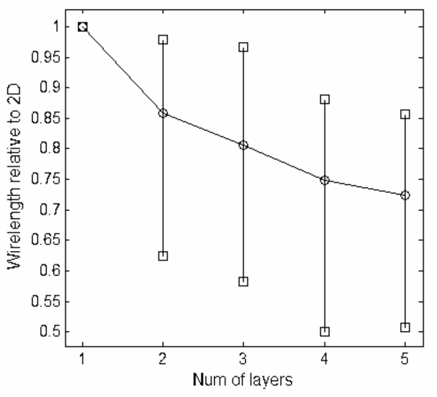

Fig. 8 Total wire-length as a function of the number of layers when the length of an inter-layer via is one

To better understand the results of Fig. 7, we studied the relationship between delay improvement and the internal structure of the circuits; we can see that for some circuits the delay decrease can be up to 23% for 3D integration into two and three layers, while it could be as bad as 20% when five layers are used. The main reason for the lesser decrease in circuit delay (compared to the potential significant decreases one can achieve with 3D integration of standard cell integrated circuits) is that in FPGAs the net delay is proportional to the number of *switches* on the net, rather than the Manhattan distance. Therefore, the “segmentation” due to switches necessary for connecting terminals of nets spanning more layers will impair the benefits of 3D integration for FPGAs with a simplistic vertical routing architecture. We also note that the increase in the delay of some circuits is due to their high internal connectivity, which in turn requires a significant fraction of nets to span at least two layers (see Fig. 4). To further probe this fact, Table 2 shows some statistics for the two circuits (*Pdc* and *Alu4*) corresponding to the minimum (delay is better with 3D) and to the maximum (delay is worse with 3D) square-points for a number of layers equal to three in Fig. 7.

Table 2 Placement statistics for Pdc and Alu4

Circuit	Obs.	% nets spanning more layers	Avg. net WL	Avg. no pins / net	Avg. no nets / block
Pdc	Min for *x=3* in Fig. 7	8.79%	11.73	4.74	4.7
Alu4	Max for *x=3* in Fig. 7	11.73%	7.58	4.52	4.49

Indeed, we can see that a larger fraction of nets of *Alu4* span two or more layers (as a consequence of higher internal connectivity) and therefore the delay cannot be improved by 3D integration when a simple vertical routing architecture is used.

In the second set of simulations we placed all circuits and recorded the average circuit delay for the ideal situation where the delay of an inter-layer wire is assumed to be equal to the delay of a segment of length one, regardless of how many layers separate the net terminals. This setup serves the purpose of analyzing the upper bound of maximum potential delay improvements using our method, which can be achieved by the 3D integration[5]. The simulation results are shown in Fig. 9. It can be seen that delay improvements of up to 30% (average of 15%) can be achieved using 5 layers.

We note that the total wire-length decreases as more layers are used (see Fig. 8). This decrease can be up to 50% for some circuits, mainly depending on the internal connectivity of the circuits. If the length of the inter-layer via increases, then the total wire-length decrease will be less. That is mainly because the fraction of the vertical wire-length relative to the total wire-length will become significant and also the average net delay will increase due to bending (i.e., switches) of nets spanning more layers. It has to be noted that the decrease in total wire-length can have favorable impact on the routing congestion (hence channel width), as well as power dissipation (especially due to the fact that 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.* in [22].

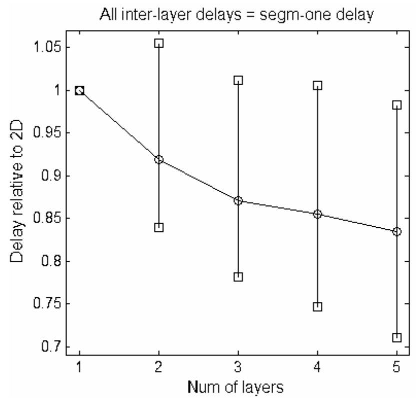

Fig. 9 Circuit delay as a function of the number of layers when the inter-layer delay is independent of distance between layers (equal to the delay of a unit segment)

On the other hand, the potential gain in terms of circuit delay is smaller (see Fig. 7) unless the 3D technology offers a rich vertical routing architecture that has vias that span multiple layers without using a switch (similar to segments of length, *e.g.*, two, six, and long in the 2D architectures). Technologically, this is doable. However, further research needs to be done to find the exact patterns and lengths of such vertical routing segments. The 2D routing architecture cannot be simply extended to the 3D due to the more restrictive limitations on the number of vertical segments.

6. CONCLUSION

Benefits which 3D FPGA integration can offer were analyzed using a new placement algorithm. The placement algorithm is partitioning-based and has integrated techniques for minimization

[5] We should emphasize, however, that this bound is not going to be far off from a post-routing analysis. In the 2D FPGA architectures of today, the delay of longer segments is comparable to the delay of unit segments. For example, in the Xilinx Virtex architecture, a hex segment of length six has a delay of 1.1 times the delay of the unit segment.

of the 3D bounding-boxes of nets and of the delays of the critical paths. Simulation experiments showed potential total wire-length decrease up to 50% for some circuits and up to 30% decrease in delay.

Our TPR placement and routing tool is meant to be the 3D counterpart of VPR and will be available online for the research community.

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] J. Dpreitere, H. Neefs, H. V. Marck, J. V. Campenhout, D. B. R. Baets, H. Thienpont, and I. Veretennicoff, "An Optoelectronic 3-D Field Programmable Gate Array", *Proc. Intl. Workshop on Field-Programmable Logic and Applications*, 1994.

[3] 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, Toronto, Canada, May 1996.

[4] 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, March 1999.

[5] M. Leeser, W. Meleis, M. Vai, S. Chiricescu, W. Xu, and P. Zavracky, "Rothko: A Three-Dimensional FPGA"*, IEEE Design Test Computers*, pp. 16-23, 1998.

[6] G. Borriello, C. Ebeling, S. Hauck, and S. Burns, "The Triptych FPGA Architecture", *IEEE Trans. On VLSI Systems*, Vol. 3, No. 4, pp. 491-501, 1995.

[7] S. Chiricescu, M. Leeser, and M. M. Vai, "Design and Analysis of a Dynamically Reconfigurable Three-Dimensional FPGA", *IEEE Trans. VLSI Systems*, Vol. 9, No. 1, pp. 186-196, Feb. 2001.

[8] P. Zavracky, M. Zavracky, D. Vu, and B. Dingle, "Three-Dimensional Processor Using Transferred Thin Film Circuits", *U.S. Patent Application*, 08-531-177, Jan. 1977.

[9] P. Sailer *et al.*, "Three-Dimensional Circuits Using Transferred Films", *IEEE Circuits and Devices*, Vol. 13, No. 6, Nov. 1997, pp. 27-30.

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

[11] 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.

[12] S. Chiricescu, "Parametric Analysis of a Dynamically Reconfigurable Three-Dimensional FPGA", *Ph.D. Dissertation*, Northeastern Univ., Boston, MA, Dec. 1999.

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

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

[15] 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.

[16] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat, "3-D ICs: A novel chip design for improving deep submicron interconnect performance and systems-on-chip integration", *Proceedings of the IEEE*, vol. 89, pp. 602 - 633, May 2001.

[17] J.A. Davis, R. Venkatesan, A. Kaloyeros, M. Beylansky, S.J. Souri, K. Banerjee, K.C. Saraswat, A. Rahman, R. Reif and J.D. Meindl, "Interconnect limits on gigascale integration (GSI) in the 21st century", *Proc. IEEE*, vol. 89, pp. 305 - 324, Mar. 2001.

[18] "SoCs are `dead,' Intel manager declares", February 12, 2002. (available at http://www.eet.com/semi/news/OEG20030212S0038).

[19] J. Burns, L. McIlrath, J. Hopwood, C. Keast, D. P. Vu, K.Warner, and P. Wyatt, "An soi-based three dimensional integrated circuit technology", in *IEEE International SOI Conference*, pp. 20 - 21, Oct. 2000.

[20] 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.

[21] K. W. Guarini, A. W. Topol, M. Leong, R. Yu, L. Shi, M. R. Newport, D. J. Frank, D. V. Singh, G. M. Cohen, S. V. Nitta, D. C. Boyd, P. A. O'Neil, S. L. Tempest, H. B. Pogpe, S. Purushotharnan, and W. E. Haensch, "Electrical integrity of state-of-the-art 0.13u m SOI CMOS devices and circuits transferred for three-dimensional (3D) integrated circuit (IC) fabrication", in *Technical Digest of the International Electron Devices Meeting*, pp. 943 - 945, 2002.

[22] 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.

[23] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Can Recursive Bisection Alone Produce Routable Placements?", *Proc. ACM/IEEE DAC*, pp. 477-482, 2000.

[24] M. Burnstein and R. Pelavin, "Hierarchical Wire Routing", *IEEE Trans. CAD*, Vol. 2, No. 4, pp. 223-234, 1983.

[25] 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.

[26] U. V. Catalyurek and C. Aykanat, "Hypergraph-partitioning-based Decomposition for Parallel Sparse-matrix vector Multiplication", *IEEE Trans. Parallel and Distributed Systems*, Vol. 10, no. 7, pp. 673-693, 1999.

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

[28] E. M. Sentovich *et al.*, "SIS: A System for Sequential Circuit Synthesis", *Technical Report UCB/ERL M92/41*, University of California, Berkeley, May 1992.

[29] 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.

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

[31] P. H. Madden, "Reporting of Standard Cell Placement Results", *Proc. ACM/IEEE ISPD*, pp. 30-35, 2001.

[32] 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.

[33] 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.

[34] G. – M. Wu, M. Shyu, and Y. – W. Chang, "Universal Switch Blocks for Three-Dimensional FPGA Design", *FPGA*, pp. 254-259, 1999.