A Timing-driven Binding Algorithm for High-Level Synthesis of Three-dimensional Integrated Circuits

Vyas Krishnan
Department of Computer Science & Information Systems
Saint Leo University
Saint Leo, FL, USA

ABSTRACT
As semiconductor technology scaling approaches its limits, 3-D integrated circuits (3-D ICs) have been proposed as one solution to continue the push towards increasing transistor counts in VLSI circuits. Recent progress in the fabrication of three-dimensional (3-D) integrated circuits has opened the possibility of exploiting this technology to alleviate performance and power related issues raised by interconnects in advanced nanometer CMOS VLSI circuits. Physical synthesis for 3-D integrated circuits is substantially different from traditional planar integrated circuits due to the presence of additional constraints of placing circuit modules in multiple silicon layers. To realize the full potential offered by 3-D integrated circuits, high-level synthesis of these circuits must take layout-related issues unique to 3-D technology into account during the synthesis process. This paper presents a 3-D layout-aware timing-driven binding algorithm for design-space exploration during high-level synthesis. The algorithm tightly integrates the synthesis tasks of resource binding, assignment of modules to multiple silicon die, 3-D floorplanning, and through-silicon via (TSV) minimization. Elmore delay models incorporating distributed wire-delays, together with delays introduced by pins and TSVs in a 3-D integrated circuit are used to compute data-transfer delays in a data path. Accurate estimates of individual net delays, obtained from net topologies in 3-D floorplans, are used to compute wire delays. Our experimental results show that a timing-driven binding algorithm for high-level synthesis can improve delays by an average of 12.2% and a maximum of up to 20.65%.

General Terms
Computer Science – High-Level Synthesis, 3-D integrated circuits (IC), design space exploration, floorplanning, physical design;

Keywords
High-Level Synthesis;; Three-dimensional Integrated Circuits; Simulated Annealing; Timing-driven Synthesis;

1. INTRODUCTION
Aggressive scaling of CMOS technology under Moore's Law over the last three decades has enabled the realization of complex VLSI circuits with billions of transistors on a single silicon chip. However, as minimum feature sizes reach 10 nm and lower, this rapid pace of CMOS scaling has begun to slow due to technological challenges associated with semiconductor fabrication, interconnect delays, power dissipation, and circuit reliability. Three-dimensional (3-D) vertical integration of VLSI circuits is one of the technologies that can potentially alleviate some of these challenges in nanoscale CMOS VLSI [1].

A number of 3-D integrated circuit technologies have been proposed recently. In this work, we consider *wafer-stacking* technology [1, 2], due to its maturity and wider adoption when compared to other competing technologies. In wafer stacking technology, two or more layers of devices are fabricated separately, and then aligned and bonded together to form a 3-D stack. Circuit blocks are placed in each of these device layers, and interconnected through intra-layer and inter-layer wires. Conventional 3-D integration provides high-density vertical interconnects with through-silicon vias (TSVs) and can achieve higher transistor densities than conventional 2-D planar integrated circuits, thus enabling reducing wirelengths, interconnect delay and power, and chip area.

Physical synthesis of 3-D integrated circuits involves assigning circuit modules to different silicon layers in the 3-D stack. TSVs are introduced by nets connecting modules in different silicon layers. Floorplanning of 3-D integrated circuits involves assigning circuit modules to different active layers and minimizing the number of TSVs, which makes it significantly different from the floorplanning used in traditional planar integrated circuits.

High-level synthesis [3] of designs for 3-D integrated circuits must be made layout aware due to the nature of interconnects present in 3-D integrated circuits. In 3-D integrated circuits, the capacitance of TSVs is significantly higher than typical gate loads, and this has a significant impact on the signal delays, as well as the number of timing violations present in a design. In nanoscale CMOS technologies interconnect delays become a dominant part of signal delays. High-level synthesis engines must consider accurate estimates of path delays to ensure that the resulting designs satisfy timing constraints [3].

This work presents a timing-driven high-level synthesis algorithm for 3-D integrated circuits that uses accurate Elmore delay based models applied to nets extracted from floorplans explored during synthesis. We model the timing-driven synthesis as an optimization problem with the aim of minimizing signal delays. Each of the high-level synthesis steps of scheduling, resource allocation, and binding significantly impacts the physical synthesis steps of placement and routing that follow [3]. The work described in this paper tightly integrates the high-level synthesis step of resource binding with the physical synthesis steps of 3-D floorplanning, with the goal of minimizing cycle-time and latency of a scheduled dataflow graph. The main contributions of this work are:

- A layout-aware binding algorithm for design-space exploration during high-level synthesis of 3-D integrated circuits that tightly integrates the synthesis tasks of resource binding, assignment of circuit modules to silicon layers in the 3-D stack, 3-D floorplanning, and minimization of the number of TSVs.

- A timing-driven data path resource binding algorithm that uses Elmore delay models incorporating distributed wire delays, together with delays introduced by pins and TSVs in a 3-D integrated circuit, during design-space exploration. Accurate estimates of individual net delays, obtained from net topologies in 3-D floorplans, are used to compute interconnect delays.

Previous work on layout-aware high-level synthesis mainly target 2-D planar integrated circuits [4−14]. Most of these use crude estimates (such as half-perimeter wire lengths) and simple closed-form equations [15] for net delays. However, work on high-level synthesis systems aimed at 3-D layouts is still in its infancy. Previous work on high-level synthesis for 3-D integrated circuits include [16], [17], [18], and [19]. The authors of [16] and [17] formulate the high-level synthesis task and the assignment of RTL modules to various 3-D layers, as a Linear Programming problem that generates constraints to run a 3-D constraint-driven floorplanner. However, their approach, separates the the high-level synthesis tasks from the floorplanning step. In addition, LP-based approaches do not scale well with problem size and complexity. The methods presented in [18] and [19] tightly couples the high-level and floorplanning steps of the synthesis process, where the high-level synthesis decisions are guided by an integrated incremental floorplanner.

Our approach differs from all of these by the use of accurate interconnect delays of critical nets to drive high-level design space exploration of 3-D integrated circuits, while most of previous work aimed to integrate high-level synthesis and physical synthesis of traditional 2-D planar layouts. Most of these approaches typically use simple point-to-point wire length modules. In our work, we use an Elmore-delay based on a star-net model to accurately estimate net delays [20]. The main advantage of this method is that it enables the estimation of individual delays between a source pin and each sink pin of a multi-terminal net. These net delays are then used to drive binding and floorplanning decisions of the high-level synthesis engine.

This paper is organized as follows. In Section 2, describes the nature of the module binding problem in high-level synthesis of 3-D integrated circuits. Section 3 introduces the timing model used to estimate net delays. Section 4 describes our floorplan-driven binding algorithm. Section 5 presents experimental results, and Section 6 concludes the paper.

2. PROBLEM DESCRIPTION

The input to the algorithm is a scheduled dataflow graph (DFG), and an allocated set of resources [3]. It is assumed that a library of components to be used for implementing the datapath is available. The output of the algorithm is an RTL binding [3] and its corresponding 3-D floorplan. The objective of the algorithm is to concurrently optimize the following design metrics:

- the cycle time,
- the footprint area,
- the difference in dimensions among 3-D floorplan layers,
- the total wire length,
- the through-silicon via (TSV) count.

The cycle time is determined by the longest register-to-register path in a scheduled step [3], which includes functional unit delays, multiplexor delays, register delays, and delays due to wires, pins, and TSVs in a 3-D floorplan.

Since a 3-D integrated circuit is created by vertically stacking several device layers, the floorplan areas of all the layers must closely match by minimizing the difference in floorplan dimensions among individual layers. This is necessary to ensure that the overall footprint area of the 3-D stack is minimized. For example, assuming two layers, L1 and L2, if the height of L1 is larger than L2, and similarly if the width of L2 is larger than that of L1, the need for matching layer dimensions to aid manufacturing would result in a significant portion of the silicon area to be wasted. Our synthesis algorithm aims to minimize the differences in the sizes of the floorplan layers in the 3-D stack. The footprint area of a 3-D floorplan is computed by determining the maximum dimension (width and height) among all active layers.

Through-silicon vias (TSV) connect nets between circuit modules or gates that are located in different silicon layers, thus establishing an inter-layer connection. Minimizing the number of TSVs is important for two reasons. In current fabrication technology, TSVs are of much larger sizes than the regular vias between metal layers in each silicon layer [1]. This imposes an upper bound on the maximum number of TSVs that can be accommodated between any two silicon layers. Additionally, TSVs act as via blockages, impacting the routing congestion in the resulting layout.

3. TIMING MODEL

To estimate the net delays we use the Elmore delay, based on the star model for a multi-terminal net, similar to that proposed in [20]. The main advantage of this model is that it allows us to accurately estimate an individual delay between the source pin and each sink pin of a net. This is important since sink pin delays among pins on a net can differ significantly, especially for long nets, nets with a large fanouts, and nets that connect modules lying in different floorplan layers. Since, the capacitance of TSVs are substantially larger than regular vias, they significantly impact net delays. The computation of individual delays enables much greater accuracy of estimated net delays, when compared to techniques that treat a multi-bit net as a single wire.

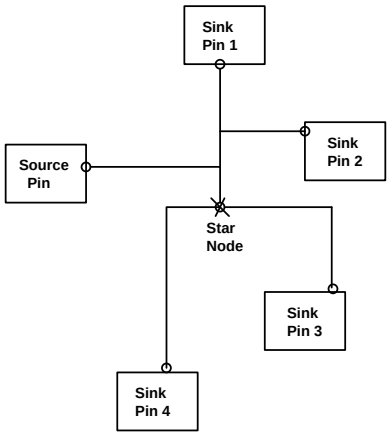

Figure 1. Star Tree Model for Multi-terminal Nets

A net is modeled by a star topology, where all the pins comprising a net are connected to a node termed the star node. Figure 1 shows an example of this net model for a 5-pin net. Given module coordinates on a floorplan, the star node is computed as the center gravity of all pins of the net, and the Manhattan distances of all pins from the star node is then

obtained. These distances are used to compute the equivalent π-model RC circuit of the wire segments connecting each of the sink pins to the source pin. The module delay is modeled as the sum of the intrinsic module delay, and a load dependent delay that is linearly proportional to the external load capacitance. The star model directly leads us to an RC-tree, for which the Elmore delay for each sink pin can be calculated in linear time.

We use the following notation in our Elmore delay model of the RC-tree of a star topology [20].

- p_0: the source pin of the RC-tree,
- e_i: the edge from a node n_i to its parent,
- r_i: the resistance of edge e_i,
- c_i: capacitance of edge e_i,
- C_i: the total capacitance of a tree rooted at n_i,
- r_0: driver resistance at source.

The Elmore delay from the source pin p_0 to a sink pin p_i is given by:

$$t_{p_0,p_i} = r_0 \cdot C_0 + \sum_{e \in \text{path } (p_0,p_i)} r_j \cdot \left(c_j/2 + C_j\right)$$

Assuming uniform wire width, the resistance and capacitance of an edge, *i.e.*, r_i and c_i of the edge e_i are proportional to its length. Approximating each of the wire segments in an RC-tree with its equivalent π-model, the Elmore delay equation implies that the delay from p_0 to a sink pin p_i is proportional to the square of the length of the wire segments between p_0 and p_i. This quadratic dependency suggests that in order to minimize the Elmore delay for a sink pin, the length of the path between the source and sink pins should be minimized. In the Elmore delay formulation, the load capacitances of pins are multiplied by the resistance of the wire segments on the path between source and sink pins. Therefore, pins with larger capacitance should be closer to the source. Further, to minimize delay to any sink pin, the total tree capacitance C_0 seen at the source pin should be minimized.

In a high-level synthesis flow, decisions made during binding determine the number of sink pins driven by a source pin. For example, a register shared by a larger number of variables may require the register to drive a large number of datapath modules. Similarly, the nature of datapath module binding could significantly impact the effective loads driven by the datapath modules. Binding also impacts the number and types of multiplexors needed in the datapath, which also influence the load capacitance seen by module pins. The lengths of wire segments to different sink pins in multi-terminal nets are determined by the relative placement of modules in a floorplan. Hence, to be effective, a timing-driven binding algorithm for high-level synthesis must be consider the impact of both binding and floorplanning on the estimated net delays.

In this work we present a timing-driven high-level synthesis algorithm for three-dimensional integrated circuits that uses accurate net delay models derived from layout-level estimates of datapaths explored during synthesis. The pin to pin delays used in our delay computation takes into account sink pin capacitances, wire delays estimated from the placement of the modules, and the delays introduced by TSVs. In the star model used to estimate pin to pin delays for all source-sink pairs in a 3-D floorplan, the location of the star node is determined by computing the center of gravity of the x and y coordinates of all pins connected by a net. In our star model, the star node is always placed on the same floorplan layer as the source pin. This ensures that delays introduced by TSVs only affect sink pins located in floorplan layers other than that of the source pin.

4. TIMING-DRIVEN BINDING ALGORITHM

The timing-driven binding algorithm is based on a Simulated Annealing framework described in [21]. The algorithm accepts a scheduled data flow graph and a resource allocation for the schedule. A compatibility graph [3] for each resource type is then extracted from these two, and provided as an input to a Simulated Annealing based floorplan-driven binding algorithm then determines the best RTL datapath and its 3-D floorplan.

Our technique is a Simulated Annealing (SA) based iterative improvement algorithm that simultaneously performs a search for optimal module bindings and 3-D floorplans. We chose an SA-based approach primarily due to its proven performance when applied to floorplanning [22]. Since our algorithm aims to tightly integrate binding and floorplanning, the search for optimal bindings was also implemented as moves in the SA-framework. A unique feature of our algorithm is the use of two interleaved sequence of moves that alternately perturb resource bindings and datapath floorplans, where a fixed number of binding moves are attempted first, followed by a fixed number of floorplanning moves. This allows the algorithm to perform a neighborhood search of the binding space, followed by a search for an optimal floorplan for the resulting resource bindings. The need for performing a local search arises from the fact that often a combination of binding changes may be needed to improve a solution [21]. Likewise, a combination of floorplan moves may be needed on a given binding, to improve the solution. The number of floorplanning and binding moves attempted at every temperature can be set independently. In all our experiments, we set the number of binding moves per temperature to be 5 × (Number of DFG operations + Number of DFG edges), while the number of floorplanning moves was set to $10 \cdot M$, where M is the number of floorplan modules. By interleaving a chain of binding moves followed by a chain of floorplanning moves, the SA performs a neighborhood search of the floorplan and binding spaces independently, at every temperature.

Changes in binding can significantly affect the netlist topology in a datapath, and thus the resulting floorplan and wire length statistics [21]. By following a neighborhood search of the binding space with a neighborhood search floorplanning space, any changes in the netlist topology are immediately reflected in the actual floorplan. Due to the incremental floorplan update feature, the SA need only modify the current floorplan with every binding move sequence, without the need to perform a time-consuming floorplan generation step from scratch. This incremental update of the floorplan makes our SA is very efficient, since the new floorplan usually has only a small difference from the previous one.

4.1 Representation of 3-D Floorplans

The floorplanner used in this work is based on the sequence pair representation proposed by Murata et al. [23] The sequence pair (SP) representation can efficiently represent any topological placement of rectangular modules, mainly because of its non-slicing structure. While the original sequence pair representation was developed for the 2-D floorplanning problem, we extend the representation to handle the 3-D

floorplans used in this work. To perform the 3-D placement, we maintain a multi-level sequence pair data structure, with one sequence pair for each device layer of the 3-D stack. Thus, a separate sequence pair is used to represent the placement of RTL modules in each of these device layers. At any time during the search process, an allocated module or register can be placed in any one of the device layers. For a full design space exploration, we allow inter-layer moves, under which an RTL module can be removed from the SP of one device layer and inserted in the SP of another device layer. Five floorplan perturbation operators are used in our algorithm, as described below:

- **Module Rotate**, which rotates a module over 90 degrees in the clockwise direction.
- **Intra-layer move**, which moves a selected module from its current location in a sequence pair, to a different location in the same sequence pair. This move essentially relocates a module within the same silicon layer.
- **Intra-layer swap**, which swaps the positions of two modules in a sequence pair.
- **Inter-layer move**, which moves a module to a different device layer in the 3-D stack.
- **Inter-layer swap**, which swaps two modules located in different device layers.

The first three moves are borrowed from a traditional 2-D floorplanning problem, and hence only affect the floorplan in a single layer. The last two moves allow the algorithm to explore the space of 3-D floorplans.

4.2 Representation of Module Bindings

For this work, we assume point-to-point multiplexer based interconnection among the data path modules [3]. The module bindings are determined by the compatibility graphs for each RTL module type derived from the scheduled DFG, and the allocated number of RTL resources. While the number of allocated resources are determined prior to module binding, the number and types of multiplexers can only be determined after the binding step. The number and types of multiplexers change with different bindings. Since the area and wiring overheads of the multiplexers can be significant, the binding and floorplanning steps are strongly inter-dependent. In 3-D layouts, the active layer assignments of modules and their bindings also strongly influence the number of TSVs needed to interconnect the modules. To determine feasible bindings for a given schedule, the SA maintains a compatibility graph for each resource type. All binding related SA moves are guided by this compatibility graph, to ensure that all bindings determined by the SA are legal, for the given schedule.

Three types of binding moves are defined, as described below:

- **Move Binding**, which reassigns the binding of a DFG operation from its current module, to another compatible module. Likewise, the same move is also applied to DFG variables and their register bindings.
- **Swap Binding**, which swaps the bindings of compatible DFG operations assigned to two different modules. The same operation is also applied to DFG variables and register bindings.
- **Swap inputs**, which interchanges the inputs of a module that performs a commutative operation.

In all of these SA moves, if the number of sources at the input of a resource (module or register) changes as a result of a change in the binding, multiplexers can vanish, appear, or change their input sizes. Any change to the number and types of multiplexers resulting from a binding move is immediately reflected in the floorplan. If a new multiplexer is added to the RTL data path as a result of a binding move, it is also added to the sequence pair of a randomly chosen layer lying in the three-dimensional bounding box enclosing the source and destination modules of the multiplexer. If a multiplexer vanishes as a result of a binding move, it is removed from the corresponding sequence pair containing it. Similarly, if the multiplexer size changes, its type is updated accordingly.

4.3 Cost Function

Our timing-driven binding algorithm concurrently optimizes four different metrics, namely, the cycle time for the schedule, the chip area, the total wirelength, and the number of TSVs. In addition, for 3-D floorplans, the final packed area of each floorplan layer must match, dictated by the need for the layer dimensions to match for fabrication. To ensure this, we use the concept of dimension deviation $dev(F)$ from [24]. Here, $dev(F)$ represents the deviation of the upper-right hand corner of a floorplan layer from the average of Ave_x and Ave_y values. The Ave_x value is computed as $\sum ux(f_i)/k$, where $ux(f_i)$ is the x-coordinate of the upper right-hand of floorplan i, and k represents the number of device layers. Ave_y is calculated in a similar manner. Thus, $dev(F)$ is formulated as

$$dev(F) = \sum_{i=1}^{N} (|Ave_x - ux(f_i)| + |Ave_y - uy(f_i)|)^2$$

The cost function used by the SA is,

$$cost = w_1 \cdot T_{norm} + w_2 \cdot A_{norm} + w_3 \cdot W_{norm} + w_4 \cdot V_{norm}$$

where,

$$T_{norm} = \frac{T_{new}}{T_{old}}$$

$$dev(F)_{norm} = \frac{dev(F)_{new}}{dev(F)_{old}}$$

$$A_{norm} = \frac{\text{Area}_{new}}{\text{Area}_{old}} + dev(F)_{norm}$$

$$V_{norm} = \frac{(\text{TSV count})_{new}}{(\text{TSV count})_{old}}$$

$$W_{norm} = \frac{\text{Wirelength}_{new}}{\text{Wirelength}_{old}}$$

In the cost function, T_{norm} represents the normalized cycle time, A_{norm} represents the normalized chip area, W_{norm} is the normalized total wirelength, and V_{norm} is the normalized TSV count. The terms of the cost function for a new solution is normalized with respect to the current solution. The subscripts "old" and "new" respectively refer to the solution before and after applying any of the SA moves. Based on a number of experiments, the following settings for the coefficients of the cost function were found to work well: w_1 = 0.50, w_3 = 0.25, and $w_2 = w_4$ = 0.125.

Minimizing the chip area encourages the SA to identify bindings with minimal multiplexer complexity in terms of number of multiplexers and their sizes. This also reflects on the resulting wiring complexity due to resource sharing. Minimizing the differences in the widths and heights of each of the device layers encourages the SA to search for floorplans that tend to match the dimensions of all the floorplan layers.

In our algorithm, two-terminal and multi-terminal nets are treated differently, when estimating wirelengths and net delays. The wirelengths of all 2-pin nets are estimated using traditional

Table 1: Comparison of proposed approach with traditional wirelength-driven synthesis

	Device Layers	Algorithm Type	Area	Total Wirelength	TSV Count	Cycle Time (ps)	Improvement over wirelength-driven synthesis
IIR	2	Timing-driven	128809	19381	128	1776	+14.65%
		Wirelength-driven	129756	17117	128	2063	
	3	Timing-driven	151133	16541	224	1766	+13.80%
		Wirelength -driven	129833	15151	248	2056	
	4	Timing-driven	129678	17268	320	1763	+14.34%
		Wirelength -driven	128861	14988	280	2058	
	5	Timing-driven	143051	19718	248	1771	+14.21%
		Wirelength -driven	129833	15152	248	2056	
EWF	2	Timing-driven	99578	36114	264	2410	+12.35%
		Wirelength -driven	99442	30166	216	2749	
	3	Timing-driven	109318	32815	472	2735	+0.78%
		Wirelength -driven	108156	29919	312	2756	
	4	Timing-driven	115694	24309	488	2393	+20.65%
		Wirelength -driven	113892	34475	440	3016	
	5	Timing-driven	114558	30909	504	2402	+11.78%
		Wirelength -driven	115473	29481	560	2722	
DCT	2	Timing-driven	228261	96436	624	2708	+10.72%
		Wirelength -driven	210236	107972	592	3033	
	3	Timing-driven	238043	87719	968	2690	+10.18%
		Wirelength -driven	214751	77481	856	2995	
	4	Timing-driven	257777	77886	1240	2670	+10.37%
		Wirelength -driven	258197	84139	1210	2979	
	5	Timing-driven	268043	81728	1816	2661	+12.35%
		Wirelength -driven	257669	80402	1752	3036	

half-perimeter bounding box approach. However, for multi-terminal nets, we use the star model proposed in [20] to represent their topology, where all the pins comprising a net are connected to a node termed the star node. Given module coordinates on a floorplan, the star node is computed as the center gravity of all pins of the net, and the Manhattan distances of all pins from the star node is then obtained. These distances are used to compute the length of the wire segments of a net, and hence the net length. The sum of all the nets in a floorplan represents the total wirelength. Minimizing the total wire length is complementary to finding minimal area implementations. This also guides the SA to exploit the additional placement freedom afforded by the availability of multiple layers in a 3-D architecture. Since wirelengths along the *z*-plane are significantly smaller than average wire lengths on the xy-plane, the wire length minimization metric is a very useful search parameter in minimizing average delays

between RTL modules, and in guiding the search towards implementations with minimal register-to-register delay values.

Though the presence of multiple device layers provides opportunities for minimizing total wirelengths, there is trade-off involved here, because the relatively large pitch of TSVs lead to increased routing congestion, the relatively large circuit parasitics introduced by TSVs impact net delays, and the presence of TSVs can impact the manufacturing yield [25].

5. EXPERIMENTAL RESULTS

The proposed algorithm was implemented in C++ and executed on a Linux workstation running on a 1.8 GHz Intel Core i5 processor with 6 GB of RAM. The RTL modules, used in our module library were created from behavioral Verilog descriptions and converted to structural Verilog using *BuildGates*, an RTL synthesis tool from *Cadence Design Systems*. They were then mapped to a 90 nm, 6-metal standard cell library, and placed and routed by *Cadence Encounter* and *WarpRoute*. The netlists extracted from these layouts were then analyzed for timing delays using *Synopsis PrimeTime*. The areas and delays from the actual layouts of these characterized RTL macro cells served as the inputs to our algorithm. The placed and routed macro cells are then exported in DEF format to the module library. These DEF files were used to create the floorplans for all the benchmark circuits synthesized by our algorithm. To assess the utility of our algorithm, we tested it on three benchmarks drawn from DSP applications. The characteristics of the DSP benchmarks are as follows:

- 8-tap IIR filter, with 9 DFG nodes, 19 DFG edges,
- Elliptic filter (EWF), with 34 DFG nodes, 43 DFG edges,
- 1-point 8X8 DCT filter, with 48 DFG nodes, 72 DFG edges.

Each of these benchmarks was specified as a control dataflow graph (DFG), capturing the behavioral description of the architecture to be synthesized. A behavioral synthesis tool developed by our group was then used to schedule and determine the resource allocations for these benchmarks.

A set of experiments was done to compare the performance of our timing-driven binding and floorplanning algorithm with that of a non-timing-driven floorplan-aware binding algorithm.

Table 1 compares the performance of the proposed timing-driven binding approach to that of a wirelength-driven approach. In the table, column 2 specifies the number of floorplan layers in a 3-D integrated circuit. Column 3 designates the algorithm used to synthesize the designs. The algorithm type (Timing-driven and wirelength-driven) indicates the two types of floorplan-aware binding algorithms compared in this work. Column 4 represents the total floorplan area of the 3-D stack representing the sum of the areas of all floorplan layers in the 3-D stack. The total wirelength in Column 5 is the sum of the all net lengths in the floorplan. The number of TSVs present in a floorplan is shown in Column 6. Column 7 shows the minimum estimated cycle-time for designs synthesized using these two approaches, and column 8 indicates the percentage improvement in the cycle-time obtained with our timing-driven binding algorithm.

From the table it can be seen that a timing-driven binding can achieve significant reductions in the clock cycle-time, when compared to a traditional area and wirelength driven flow. The overall average improvement was 12.2% for the benchmarks tested. These improvements in wirelengths are due to better wirelength distributions for critical nets as a result of our timing-driven binding approach.

Our experiments show that a smaller chip area or wirelength do not necessarily result in smaller delays. Minimizing delays requires that the binding step during high-level synthesis carefully consider the impact of its decisions on the wire delays, and interlayer via loads driven by the source pins. This is especially true for nets on the critical path.

6. CONCLUSIONS

In this work we address the problem of layout-aware timing-driven binding for three dimensional vertically integrated systems as part of a physical aware behavioral synthesis flow. We outline a simulated annealing based formulation for the combined binding and floorplanning problem. Our algorithm proposes a module binding algorithm that uses Elmore delay models to accurately estimate individual net delays using net topologies extracted from 3-D floorplans. A distributed wire-delay model is used to account for wire delays, together with delays introduced by pins and interlayer vias in a 3-D floorplan. These net delays are used to compute the register-to-register delays for all the data transfers in a dataflow graph, and the maximum achievable clock cycle time for data paths examined during design-space exploration. Experimental results show that our algorithm can obtain reductions in the critical path delays on average of 12.2%, with a maximum of up to 20.65% in the achievable minimum clock cycle times, when compared to traditional synthesis driven by area and wirelength minimization.

7. REFERENCES

[1] V.F. Pavlidis, I. Davidis, and E.G. Friedman, "Three-Dimensional Integrated Circuit Design, Second Edition" Morgan Kaufman Publishers, 2017.

[2] R. Reif, et al., "Fabrication Technologies for Three-Dimensional Integrated Circuits," *Proceedings of the Internal Symposium on Quality Electronic Devices* (*ISQED 2002*).

[3] D. Gajski *et.al*. "High-Level Synthesis: Introduction to Chip and System Design," Kluwer Academic Publishers, 1992.

[4] J.-P. Weng and A.C. Parker, "3D scheduling: High-level synthesis with floorplanning," *Proc. of the 28th ACM/IEEE Conference on Design Automation, 1991*.

[5] Y.M. Fang and D.F. Wong, "Simultaneous functional-unit binding and floorplanning," *Proceedings of the International Conference on Computer Aided Design* (*ICCAD 1994*).

[6] P. Prabhakaran and P. Bannerjee, "Simultaneous Scheduling, Binding, and Floorplanning in high-level synthesis," *Proc. Intl. Conf. VLSI Design 1998*.

[7] S. Tarafdar, M. Leeser, and Z. Yin, "Integrating Floorplanning in Data Transfer Based High-Level Synthesis," *Proceedings of the International Conference on Computer Aided Design* (*ICCAD 1998*).

[8] D. Kim, J. Jung, S. Lee, J. Jeon, and K. Choi, "Behavior-to-Placed RTL Synthesis with Performance- Driven Placement," *Proc. of ICCAD 2001*, pp. 320-325.

[9] M.Xu and F.J.Kurdahi, "Layout-driven RTL binding techniques for high-level synthesis using accurate estimators," *ACM Trans. Design Automation of Electronic Systems*, vol.2, no.4, pp.312-343, 1997.

[10] W. E. Dougherty and D. E. Thomas, "Unifying behavioral synthesis and physical design," in *Proceedings of the Design Automation Conference* (*DAC 2000*).

[11] A. Davoodi and A.Srivastava, "Power-Driven Simultaneous Resource Binding and Floorplanning: A Probabilistic Approach, *IEEE Trans. Computer Aided Design of Integrated Circuits & Systems*, 2005, pp. 934-942.

[12] A. Stammermann, *et.al.*, "Binding, Allocation and Floorplanning in Low Power High-Level Synthesis", *Proceedings of the International Conference on Computer Aided Design* (*ICCAD 2003*).

[13] L. Zhong and N. K. Jha, "Interconnect-aware high-level synthesis for low power," in *Proceedings of the International Conference on Computer Aided Design* (*ICCAD 2002*).

[14] Z. Gu, et. al., "Incremental Exploration of the Combined Physical and Behavioral Design Space," in *Proc. DAC 2005*.

[15] H.B.Bakoglu, *Circuits, Interconnects, and Packaging for VLSI*. Reading, MA: Addison-Wesley, 1990.

[16] M. Mukherjee and R. Vemuri, "Simultaneous Scheduling, Binding and Layer Assignment for Synthesis of Vertically Integrated 3D Systems," in *Proceedings of the International Conference on Computer Design* (*ICCD 2004*).

[17] M. Mukherjee and R. Vemuri, "On Physical-Aware Synthesis of Vertically Integrated 3D Systems," in *Proc. International. Conference on. VLSI Design 2005*.

[18] V. Krishnan and S. Katkoori, "A 3-D Layout Aware Binding Algorithm for High-Level Synthesis of Three-Dimensional Integrated Circuits," in *Proceedings of the International Symposium on Quality Electronic Devices* (*ISQED 2007*).

[19] Y. Chen, et. al., "3DHLS: Incorporating High-Level Synthesis in Physical Planning of Three-Dimensional (3D) ICs," *Proceedings of the Design Automation and Test in Europe Conference* (*DATE 2012*).

[20] F.Mo, A.Tabbara, and R.K.Brayton,"A Timing-driven Macrocell Placement Algorithm", in *Proceedings of the International Conference on Computer Design* (*ICCD 2001*).

[21] V. Krishnan and S. Katkoori, "Minimizing Wire Delays by Net-Topology Aware Binding during Floorplan-Driven High Level Synthesis," in *Proceedings International Conference on Very Large Scale Integration* (*VLSI-SoC 2007*).

[22] J. Cong et al., "A Thermal-Driven Floorplanning Algorithm for 3D ICs," in *Proceedings of the International Conference on Computer Aided Design* (*ICCAD 2004*).

[23] H. Murata, et al., "VLSI Module Placement Based on Rectangle Packing by the Sequence Pair" in , *IEEE Trans. Computer Aided Design of Integrated Circuits & Systems*, 1996, vol 15(12), pp. 1518-1524.

[24] P. H. Shiu, and S. K. Lim, "Multi-layer Floorplanning for Reliable System-on-Package.", in *Proceedings of International Symposium on Circuits and Systems* (*ISCAS 2004*).

[25] X. Dong, J. Zhao, and Y. Xie, "Fabrication cost analysis and cost-aware design space exploration for 3-D ICs," *IEEE Trans. Computer Aided Design of Integrated Circuits & Systems*, 2010, pp. 1959-1972