A DSM Design Flow: Putting Floorplanning, Technology-Mapping, and Gate-Placement Together [*]

Amir H. Salek, Jinan Lou, Massoud Pedram
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089
{amir, jlou, massoud}@zugros.usc.edu

ABSTRACT - This paper presents an integrated design flow which combines floorplanning, technology mapping, and placement using a dynamic programming algorithm. The proposed design flow consists of five steps: maximum tree sub-structure formation, levelized cluster tree construction, minimum area implementation using 2-D shape functions, critical path identification, and repeated application of simultaneous floorplanning, technology mapping and gate placement along the timing critical paths. Experimental results obtained from an extensive set of benchmarks demonstrate the effectiveness of the proposed flow.

1. INTRODUCTION

The strong desire for ever-increasing levels of integration and higher performance often transcends the capabilities of traditional design tools and flows in the IC design arena. In particular, new deep-sub-micron (DSM) design considerations, such as the dominance of interconnect delays and signal integrity issues, have forced IC designers to re-evaluate the existing design methodologies and techniques. To address the DSM design challenges, one can increase the lookahead capability of high-level tools or develop new algorithms for optimally solving larger portions of the overall design problem simultaneously. This latter unification-based approach is, in our view, more promising. Indeed, the current state of CAD tools and algorithms has evolved to a point where it is both possible and necessary to combine certain steps of the logic synthesis and physical design processes.

This paper introduces an integrated approach for simultaneous floorplanning, technology-mapping, and detailed gate placement, of row-based standard-cell layout styles. The unique contributions of the paper are:

- A new design flow, *SiMPA, which simultaneously performs floorplanning, technology mapping, and placement.
- A new data structure, k-way levelized cluster tree, which represents the hierarchy of the circuit for *SiMPA.
- A new global area optimizer, *SiMPA-E, which optimizes chip area via simultaneous floorplanning, technology mapping, and gate placement.
- A new critical path optimizer, *SiMPA-R, which for a given number of critical paths effectively trades area for delay while simultaneously considering all floorplanning, technology mapping, and gate placement solutions.

The rest of this paper is organized as follows. In section 2, background and motivation behind this work are given. Section 3 describes our methodology for unifying floorplanning, technology mapping, and gate placement. In sections 4 and 5, our experimental results and concluding remarks are presented.

2. Background

Technology Mapping - It is well-known that the general technology mapping (TM) problem is NP-hard [HS96]. Keutzer in [Ke87] approximated the general TM problem by a set of tree-covering sub-problems and optimally mapped each, with respect to gate area, by a polynomial dynamic programming-based (DP) algorithm (KA). Later, Rudell in [Ru89] extended this idea to the minimum delay mapping problem. Subsequently, Touati et al. provided a mapper capable of minimizing area under delay constraints [TMBW90]. Chaudhary and Pedram presented a method for finding a set of all non-inferior mappings for a tree with different area-arrival time trade-offs [CP92]. All of the above mentioned methods assume the dominance of gate area and delay over interconnects.

Linear Placement - Linear placement (LP) is defined as the assignment of the vertices of a graph to the open slots located on a line so as to minimize a desired cost function. MINCUT is the LP problem where the cost function is the maximum cutwidth. It has been proven that this problem is NP-complete for general graphs [GJ79] but that it is polynomial in cases where the subject graph is a tree. Lengauer, in [Le82], developed an approximation algorithm (LA) for tree MINCUT problem whose cutwidth is within a factor of two of the optimal value. Later, Yannakakis introduced a polynomial optimal algorithm (YA) for tree MINCUT problem using the DP strategy [Ya85].

SiMPA - To address the high performance requirements of DSM designs, SiMPA (Simultaneous Technology Mapping and Linear Placement Algorithm) integrates technology mapping and linear placement by combining DP-based YA/LA with KA. This algorithm allows TM to access accurate physical information and LP to guide logic optimization. SiMPA-E ('E' stands for *exact total area*) is a combination

[*] This work was supported in part by SRC under contract No. 98-DJ-606, and NSF under contract No. MIP-9628999.

of YA and KA which optimally finds the minimum total (wiring+gate) area implementation of a given tree circuit [LSP97]. SiMPA-D ('D' stands for *disjoint combination based*) combines LA and KA and optimizes total area and total delay by generating three dimensional trade-off curves for tree circuits [LSP98]. FPD-SiMPA (FPD stands for *floorplan driven*) is a design flow which incorporates SiMPA-D and SiMPA-E (SiMPA-D/E) for building two dimensional implementations for DAG-structure circuits [LSP98]. The outline of this design flow is as follows: 1. Partition the initial DAG into a set of tree clusters. 2. Map and place each cluster. 3. Floorplan the clusters on a two dimensional plane. 4. Perform global routing followed by timing analysis. 5. Trade off area for delay using SiMPA-D/E for each timing or area critical cluster.

Floorplanning **-** It refers to the placement and sizing of flexible blocks and is a common feature in many hierarchical design environments. In general, floorplanning algorithms are based on one of the following methods: mathematical programming, rectangular dualization, and combinatorial based methods [Sh93]. Recently, two novel floorplanning techniques based on sequence pair [MFNK96] and bounded slicing grid structures have been proposed [NFMK96]. In this paper, a combinatorial based floorplanning method, like that in [PK92], is combined with SiMPA through the use of an appropriately defined hierarchy.

Acyclic Partitioning **-** Acyclic partitioning prevents nets from creating directed cycles among the parts. A two-way acyclic partitioning algorithm was introduced in [IPFC93], then extended to multi-way acyclic partitioning in [CLB94]. The latter is used in this work when delay optimization is added to the proposed algorithm (*SiMPA-R). The acyclic property is necessary because it allows simple handling of the timing dependencies among the parts by following a topological ordering.

3. Simultaneous Floorplanning, Technology Mapping, and Gate Placement (*SiMPA)

SiMPA-D and SiMPA-E are only capable of manipulating tree-structure circuits. However, such a structure is atypical of most circuits and therefore in FPD-SiMPA, a DAG circuit is clustered into tree clusters and each cluster is manipulated by SiMPA-D/E separately. In other words, the design flow divides the problem into a set of smaller sub-problems and then combines the sub-solutions to build the final solution. This approach can be classified as a member of the divide-and-conquer family of algorithms and consequently inherits the intrinsic shortcomings associated with this family. Generally, FPD-SiMPA restricts SiMPA to work within the individual tree clusters only, and employs the floorplanner for taking advantage of the possible inter-cluster optimizations opportunities. The floorplanner, however, is subject to the optimization decisions already made within the clusters which are not necessarily appropriate for the overall circuit optimization.

*SiMPA, the proposed algorithm in this work, addresses this deficiency by combining SiMPA with the floorplanning design step. Hence, the flooplanner controls all the intra-cluster and inter-cluster optimization opportunities and so the decisions made inside and outside the clusters are the best in the global sense. Another important property of *SiMPA is that it eliminates delay budgeting (slack distribution) in the design flow. *SiMPA, due to its integrated nature, has no need for this step. It provides designers with a conceptually simpler and more precise design tool which outputs the global trade-off picture of the design using fewer heuristics.

The way FP is merged into SiMPA is different from the way TM and LP were merged in SiMPA. TM and LP both work at the same hierarchy level (inside the tree clusters) and so it was possible to implement their combination via interleaved calls from one to another at every step of the DP-based algorithm. FP on the other hand, works at a higher level of hierarchy (among the tree clusters), and therefore, its integration into SiMPA must be achieved by other means. The floorplanner uses SiMPA to capture the shape functions of all the tree clusters. These are subsequently used by the bottom-up floorplanner to calculate the global shape function for the whole circuit.

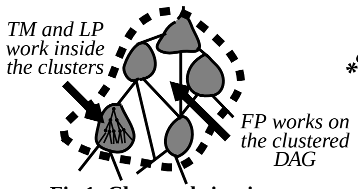


Fig.1: Clustered circuit

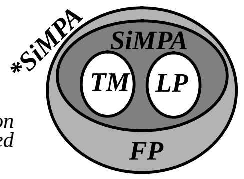


Fig.2: *SiMPA

*SiMPA comes in two flavors:

• *SiMPA-E which targets the total minimum area by generating a set of non-inferior solutions (w.r.t. shape) for the whole circuit.

• *SiMPA-R which removes the timing violations for a given implementation and a design hierarchy by simultaneously replacing and remapping the clusters on timing critical paths. This algorithm is capable of computing accurate shape/delay trade-off curves for the whole chip while speeding up one critical path of the circuit at a time.

In our flow, *SiMPA-E is first called to generate a minimum area implementation of the subject circuit, then *SiMPA-R is called to remove the timing violations one path at a time.

In *SiMPA, a special data structure is used for the representation of the design hierarchy. The first sub-section below describes and analyzes this representation. Next, *SiMPA-E and *SiMPA-R are presented and discussed.

3.1 K-LCT: A K-Way Levelized Cluster Tree

Any bottom-up technique requires, at least implicitly, a hierarchy to work with. For the case of BearFP (a DP-based floorplanner), the hierarchy is represented by a *cluster tree* which shows a grouping of macro-cells into first-level clusters, first-level clusters into second-level super clusters, and so on, until the root of the cluster tree is reached [PK92]. *SiMPA, which includes a floorplanner similar to

BearFP, likewise requires a data structure for the extraction and representation of the hierarchical organization of the tree clusters. In contrast to BearFP which uses a simple k-way tree by including the most connected blocks as sibling nodes under the same parent node, *SiMPA requires a more elaborate hierarchy and a corresponding suitable data structure to represent it. This novel data structure is called *k-way levelized cluster tree (k-LCT)* and is introduced and discussed below.

Definition1: For any given rooted tree, $T(V, E)$, $\forall v \in V$, the following terms are defined:

• *parent(v)*: immediate parent of *v*.

• *children(v)*: set of the immediate children of *v*.

• *descendents(v)*: the set of all nodes in the subtree rooted by *v*, including *v* itself.

• $leaves(v)=\{ u \mid u \in descendent(v) \wedge u$ is a leaf node $\}$.

Definition2: Given a directed acyclic graph $G(V', E')$, a rooted tree $C(V, E)$ with root r is a cluster tree for G if there exists a one-to-one function Γ: $leaves(r) \rightarrow V'$. Members of E and E' are called *cluster-edges* and *DAG-edges*, respectively. Note that for a given circuit, the corresponding DAG does not include the primary inputs or outputs which are later handled in a pad assignment phase.

Definition3: $C(V, E)$, a rooted cluster tree of a given $G(V', E')$, is called acyclic, if for any $v \in V$ and $1<i<|children(v)|$ the following statement holds: $\forall v_1,\ldots,v_i \in children(v)$, and $\forall (u_1,\ldots,u_i) \in leaves(v_1) \times \ldots \times leaves(v_i)$ then $\{(\Gamma(u_1),\Gamma(u_2)), \ldots, (\Gamma(u_{i-1}),\Gamma(u_i)), (\Gamma(u_i),\Gamma(u_1))\} \not\subset E'$.

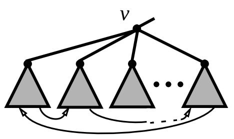
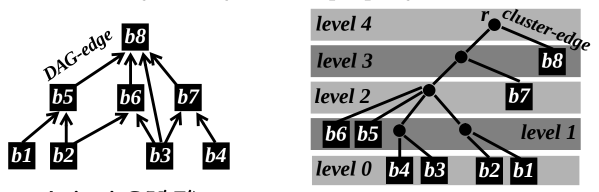


Fig.3: A cycle

Definition4: $C(V, E)$, a rooted cluster tree of a given $G(V', E')$, is a *k-LCT* if: **1)** $\forall v \in V$, $|children(v)| \leq k$, **2)**$\forall (v \in V \wedge v \notin leaves(r)), \forall (u \in leaves(r) \wedge u \notin descendents(v))$, $\forall w \in leaves(v)$, $(\Gamma(u),\Gamma(w)) \notin E'$.
In the following, $C(V, E)$ (with root r) is assumed to be a k-LCT of a given circuit $G(V', E')$ unless otherwise is stated.
Property two of Definition4 states that in a k-LCT, all the DAG-edges connect nodes to those in the same or higher level, hence guaranteeing a k-LCT to be an acyclic structure. This property empowers *SiMPA to calculate all needed wire lengths signal arrival times while synthesizing the circuit; details will be discussed later on in this paper. Fig.4. which demonstrates a k-LCT for a simple circuit can be used to verify the acyclicness property of k-LCT's.



A circuit G(V′,E′) *A 4-LCT for G, C(V,E)*

Fig.4

Lemma1: For any edge in $G(V', E')$, say (u,v) (let $\upsilon=\Gamma^{-1}(u)$ and $\omega=\Gamma^{-1}(v)$), we have $\upsilon \in descendents(parent(\omega))$.

Theorem1: $C(V, E)$, a k-LCT of a given $G(V', E')$ is an acyclic cluster tree.

The following lemmas introduce transformations on k-LCT's for the generation of a new equivalent k-LCT with different properties; such as a more balanced k-LCT which generally generates more efficient floorplans.

Lemma2: In C, any leaf node can be raised as high as the level where its lowest-level fanout resides.

Lemma3: In C, any leaf node can be lowered as low as the level where its highest-level fanin resides.

Lemma4: In C, any two nodes which have the same parent node and the total number of their immediate children is less than k can be merged together.

The 4-LCT in Fig.5. is equivalent to that shown in Fig.4, generated using a sequence of transformations as mentioned above. This tree is more desirable than that of Fig.4, since DP-based floorplanner tends to produce better results when the number of the internal nodes is $log_k(n)$. Interested readers are referred to [SLP98] for the proof of the theorems.

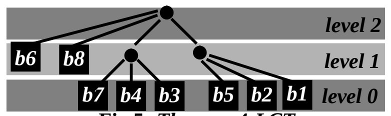


Fig.5: ***The new 4-LCT***

3.2 *SiMPA-E for Global Area Optimization

*SiMPA-E is a combination of SiMPA and floorplanning which targets total area optimization. If a bottom-up floorplanning scheme with shape propagation ability is used, *SiMPA-E is capable of finding a global height versus width trade-off curve for the whole network. This curve is then used in finding the solution satisfying the user-specified aspect ratio requirement.
Total area for a one-dimensional standard-cell layout is calculated by $A = W \cdot (h + \beta \cdot c)$, where W is the sum of the width of all the cells in a row, h is the cell height, a constant determined by the ASIC library being used, β is the minimum distance between the two adjacent wires' centers, and c is the maximum cutwidth (a.k.a. cut-density) of the design.

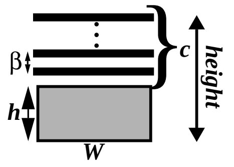
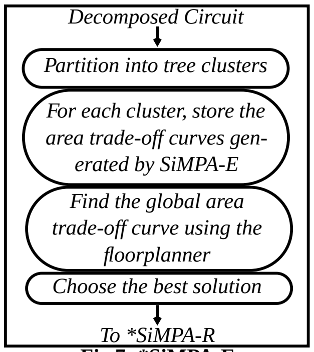


Fig.6: Area model

Lemma5: Cutwidth versus gate-area (c versus g) curves are transformed into height versus width curves, according to the following equations: $height=h+c\times\beta$ and $W=g/h$.
As shown in Fig.7, *SiMPA-E first partitions the given decomposed circuit into a set of maximal tree sub-networks (clusters). Then, SiMPA-E produces the cutwidth/gate-area trade-off curves, which are subsequently translated to height/width curve (shape function) using Lemma1, for each cluster. During the


Decomposed Circuit
Partition into tree clusters
For each cluster, store the area trade-off curves generated by SiMPA-E
Find the global area trade-off curve using the floorplanner
Choose the best solution
To *SiMPA-R
Fig.7: *SiMPA-E

third step, the set of all the shape functions is passed to a bottom-up floorplanner employed by the algorithm. The floorplanner takes these shape functions and generates the global 2-D trade-off curve. Finally, the best floorplan is picked from this curve. Note that, solutions for all the clusters are generated simultaneously from the best solution for the root of the cluster tree. Once the best global solution is found, the cluster sub-solutions which were used to construct the solution are traced back and assigned as the internal implementation of each cluster. Few observations relating to the operation of *SiMPA-E are:

Observation1: In the shape function for a cluster, the detailed physical and logical implementations of every solution are known prior to executing the floorplanner. This information is then used throughout the floorplanning process for the exact area and delay calculations of the whole or part of the design.

Observation2: *SiMPA-E achieves global area optimization by taking into account detailed design information for all the design choices and comparing solution qualities based on exact physical and logical information. Further, *SiMPA-E does not need to use any area estimators which are usually based on heuristic techniques.

Observation3: Any floorplanner which can properly use the set of calculated shape functions can be employed in this flow. However, the global height versus width trade-off curve is built most effectively using a bottom-up floorplanner with shape propagation capability. The use of this type of floorplanner, as mentioned earlier, needs a hierarchy on the clusters required for satisfying the *acyclicness* property if critical paths are intended to be resynthesized using *SiMPA-R. In this work, we employ k-LCT's (with k=4) to satisfy all these requirements.

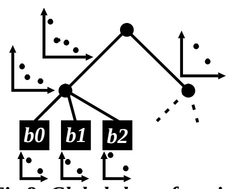


Fig.8: ***Global shape function generation***

Observation4: SiMPA-E calculates the shape function (including gate and wire areas) for every tree cluster and therefore all optimum non-inferior points are included in the shape function. However, the area taken by inter-cluster wiring is an entirely different issue. Bottom-up floorplanning can be made to estimate (probabilisticly or constructively) the inter-cluster (inter-block) connection lengths. These estimates are not exact and hence claims of optimality for *SiMPA-E are subject to inaccuracies in estimating the inter-block connection lengths and area requirements. The choice of a k-LCT hierarchy in *SiMPA-E leads not only to advantages in *SiMPA-R (discussed later), but also restricts inter-cluster interconnects which in turn reduces the impact of the inaccuracies suffered due to the use of a bottom-up approach.

Observation5: For every leaf and internal node of the cluster tree (hierarchy), *SiMPA-E stores the calculated shape functions for future reference by *SiMPA-R. This information is needed by the resizing process of the clusters on the timing critical path. The process may open up space for the timing critical clusters by optimally resizing the non-critical ones in order to allow trading space for better timing behavior inside the critical clusters.

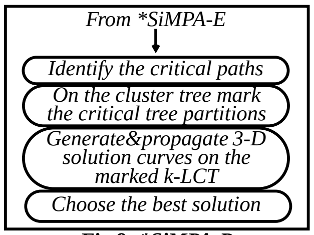
Theorem2: *SiMPA-E is a polynomial algorithm.

3.3 *SiMPA-R for Eliminating Timing Violations

Although the global shape function generated by *SiMPA-E captures the best implementation in terms of area, it is likely for the resulting implementation(s) to violate user specified timing requirements. In the *SiMPA flow, it is thus essential to employ a technique to eliminate the timing critical paths by resynthesizing and trading area for delay. After using *SiMPA-E and choosing an area optimized implementation, *SiMPA-R is employed for the purpose of fixing the timing violations.

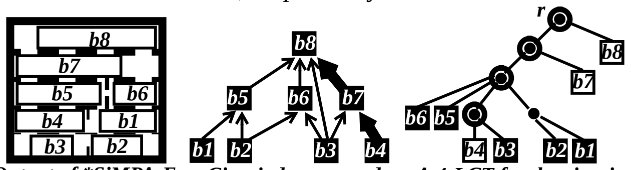
3.3.1 Critical Clusters Initial Manipulation

The output of *SiMPA-E is a fully mapped, two-dimensionally placed implementation of the given circuit. *SiMPA-R first identifies all critical paths and selects the one with the most negative slack and marks all the tree clusters on this critical path as critical. For every critical leaf node in the k-LCT, the internal nodes on the unique path to the root of the k-LCT are also marked so that floorplanning, technology mapping, and placement can be redone. We refer to the resulting representation as a marked k-LCT. Fig.10 demonstrates a simple example in which the critical path is shown by thicker arrows in the clustered circuit. In the 4-LCT the critical leaves and nodes are marked by white boxes and black circles, respectively.



From *SiMPA-E
Identify the critical paths
On the cluster tree mark the critical tree partitions
Generate&propagate 3-D solution curves on the marked k-LCT
Choose the best solution

Fig.9: ***SiMPA-R***



Output of *SiMPA-E ***Circuit decomposed into tree clusters*** ***A 4-LCT for the circuit***
Fig.10

Lemma6: In a marked k-LCT, there is at most one critical internal node and one critical leaf node among the children of every internal node.

3.3.2 3-D Curve Generation and Propagation

Having marked the k-LCT, *SiMPA-R begins the generation and propagation of the 3-D (height, width, and the critical signal ready time at the root of a subtree) solution curves from the lowest level marked internal node up towards the root. At each step, the algorithm detects the immediate children of the current internal node, *v*, and retrieves or generates their corresponding 2/3-D solution curves in order to examine the solution possibilities for all the floorplan templates. An immediate child of *v*, say *u*, is either a non-critical internal/leaf node, a critical internal node, or a critical leaf node. In the first case (a non-critical internal/leaf node), there is no marked node among the *descendents(v),* indicating the absence of a critical signal. Therefore, the corresponding solution curve is the 2-D curve already calculated and stored by *SiMPA-E. For the case of a critical internal node, a 3-D solution curve is already calculated by the previous calls to the same procedure. The situation is a bit different for a critical leaf node; in this case one of the incoming signals is the critical signal which may be provided by its marked sibling's descendents. Consequently, the arrival time of the signal depends on the implementation chosen for the sibling node. However, all the physical and logical implementation information up to that point are available and therefore for every floorplan template and block-to-room assignment, the possible critical signal arrival times can be precisely calculated.

Using these arrival times, SiMPA-D generates a 3-D solution curve for that critical cluster. These 3-D curves (for every template and assignment) are assembled into one curve and the inferior solutions are pruned out. The timing values on this curve are the critical signal's ready times at the output of *v* for each possible implementation of the subtree. Note that according to Lemma6, there exists at most one marked leaf node and therefore SiMPA-D is run for at most one node during processing of the immediate children of every *v*. If *v* has no immediate marked leaf node, the timing values of the critical internal node are the critical signal's ready times. As for the geometrical calculations, the height and width of the solutions of *v* are calculated from the solutions to the sub-problems in ways similar to *SiMPA-E's method.

Now, few remarks relating to *SiMPA-R:
•Every solution generated by *SiMPA keeps the pointers to its constituent sub-solutions in order to retrieve all the corresponding design information for the best solution selected in the final stage.
•For every template and room assignment, the relative locations of the blocks are known. Accordingly, interconnect routes within *v* are known (subject to using a bottom-up global routing algorithm. In addition, k-LCT provides *SiMPA with a proper structure which helps avoid signal cycles among *children(v)*, and therefore all timing information can be calculated according to the topological order of *children(v)*.
•This procedure proceeds recursively until it reaches the root in the k-LCT. The trade-off curve of the root reveals the global design options from which the best is selected and traced down using the stored pointers.

Theorem3: *SiMPA-R is a polynomial time algorithm.

4. Experimental Results

The sequence of steps consisting of technology independent optimization, technology mapping, placement, global and detailed routing is the conventional design flow. To verify the effectiveness of the simultaneous approach, results are presented in Table1 for two setups: I) Conventional flow. II) Technology mapping, maximal tree clustering, floorplanning (BearFP), linear placement of each cluster using YA, followed by TimberWolf and YACR. The presented results show only small differences between the two runs which points to the fact that *SiMPA's efficiency is due to its simultaneous approach to the design process rather than just improved mapping and placement algorithms.

Our next set of experiments compares the performance of the conventional design flow with our proposed DSM methodology on a number of benchmarks using a CASCADE standard cell library (0.5u HP CMOS process). Gate and wire delays are calculated using a 4-parameter delay equation (similar to that in [LSP97]) and the Elmore delay model, respectively. These experiments were run in the SIS environment [SSLM92] on an Ultra-2 Sun Sparc workstation with 256MB memory. The area and delay reported here are total chip area and delay after detailed routing. Table2 compares the results of *SiMPA with the conventional flow showing an average of 34% performance improvement with small area penalty. In these experiments, the floorplanning was performed by our preliminary k-LCT floorplanner prototype, TimberWolf was used for global routing, and YACR was used for detailed routing. The runtimes for *SiMPA remained comparable to the ones for the conventional flow, as shown in Table3 for a number of benchmark circuits.

	I.		*II.*		*II over I*	
Circuit	***Area***	***Delay***	***Area***	***Delay***	***Area Ratio***	***Delay Ratio***
alu2	*2459*	*13.13*	*2402*	*13.10*	*0.98*	*1.00*
apex7	*1254*	*6.45*	*1260*	*6.61*	*1.00*	*1.02*
cm150a	*226*	*3.22*	*221*	*3.18*	*0.98*	*0.99*
cm151a	*168*	*2.97*	*172*	*2.91*	*1.03*	*0.98*
cm162a	*229*	*2.50*	*234*	*2.57*	*1.02*	*1.03*
duke2	*3283*	*10.16*	*3241*	*9.75*	*0.99*	*0.96*
k2a	*7555*	*15.96*	*7570*	*19.51*	*1.00*	*1.22*
rot	*4748*	*8.91*	*4651*	*12.12*	*0.98*	*1.36*
table3	*5014*	*55.17*	*5030*	*66.34*	*1.00*	*1.20*
C1908	*4288*	*17.46*	*4477*	*16.67*	*1.04*	*0.95*
C880	*2673*	*9.53*	*2496*	*10.61*	*0.93*	*1.11*
C1355	*2452*	*8.84*	*2517*	*9.69*	*1.03*	*1.10*
C3540	*10101*	*27.63*	*9674*	*36.09*	*0.96*	*1.31*
				Average Ratios:	***1.00***	***1.09***

Table 1: Verifying the effectiveness of the simultaneous approach

5. Conclusion

This paper presents a novel design flow *SiMPA, which performs simultaneous floorplanning, technology mapping and linear placement. *SiMPA is a dynamic-programming based algorithm which first computes the shape function of every tree cluster using SiMPA-E and then utilizes a bottom-up floorplanner to generate higher level shape based on a k-way levelized cluster tree. After the minimum area solution is chosen and the critical paths are identified, the clusters on each critical path are re-floorplanned, re-mapped and re-placed simultaneously to achieve improved timing. This new design flow yields very high quality circuits in terms of post layout chip area and delay.

	Conventional Flow		**SiMPA*		*Ratios*	
Circuit	**Area**	**Delay**	**Area**	**Delay**	**Area**	**Delay**
alu2	*2459*	*13.13*	*2017*	*8.60*	*0.82*	*0.65*
alu4	*4258*	*22.89*	*3725*	*11.89*	*0.87*	*0.52*
apex6	*4554*	*12.01*	*5106*	*6.82*	*1.12*	*0.57*
apex7	*1254*	*6.45*	*1317*	*4.31*	*1.05*	*0.67*
b9	*620*	*3.62*	*598*	*2.57*	*0.96*	*0.71*
cm150a	*226*	*3.22*	*215*	*1.42*	*0.95*	*0.44*
cm151a	*168*	*2.97*	*94*	*1.46*	*0.56*	*0.49*
comp	*641*	*2.84*	*544*	*2.22*	*0.85*	*0.78*
con1	*83*	*1.55*	*93*	*0.76*	*1.12*	*0.49*
cordic	*376*	*2.74*	*266*	*1.90*	*0.71*	*0.69*
dalu	*6356*	*19.45*	*5959*	*14.75*	*0.94*	*0.76*
duke2	*3283*	*10.16*	*2708*	*8.64*	*0.82*	*0.85*
f51m	*300*	*5.76*	*370*	*4.42*	*1.23*	*0.77*
k2a	*7555*	*15.96*	*9421*	*12.37*	*1.25*	*0.78*
lal	*560*	*4.09*	*537*	*2.60*	*0.96*	*0.64*
misex3	*3374*	*12.89*	*3559*	*10.73*	*1.05*	*0.83*
mux	*221*	*3.03*	*234*	*1.46*	*1.06*	*0.48*
pcle	*471*	*3.66*	*300*	*2.41*	*0.64*	*0.66*
pcler8	*602*	*3.90*	*570*	*3.00*	*0.95*	*0.77*
ritex	*337*	*2.16*	*299*	*1.84*	*0.89*	*0.85*
ritex1	*90*	*1.09*	*94*	*0.77*	*1.04*	*0.71*
rot	*4748*	*8.91*	*4864*	*8.20*	*1.02*	*0.92*
term1	*711*	*3.92*	*778*	*2.72*	*1.09*	*0.69*
z4ml	*267*	*3.45*	*182*	*1.66*	*0.68*	*0.48*
5xp1	*641*	*7.66*	*546*	*4.23*	*0.85*	*0.55*
9sym	*811*	*7.71*	*1019*	*5.62*	*1.26*	*0.73*
Z5xp1	*398*	*7.08*	*459*	*5.79*	*1.15*	*0.82*
Z9sym	*876*	*7.77*	*1068*	*5.14*	*1.22*	*0.66*
b12	*393*	*3.81*	*355*	*1.97*	*0.90*	*0.52*
bw	*1017*	*9.33*	*845*	*5.77*	*0.83*	*0.62*
clip	*689*	*7.85*	*768*	*3.61*	*1.11*	*0.46*
e64	*1602*	*8.29*	*1648*	*9.13*	*1.03*	*1.10*
inc	*587*	*7.78*	*577*	*3.45*	*0.98*	*0.44*
misex1	*293*	*5.90*	*313*	*2.61*	*1.07*	*0.44*
misex2	*517*	*4.03*	*514*	*2.48*	*0.99*	*0.62*
misex3c	*2629*	*19.50*	*2765*	*8.54*	*1.05*	*0.44*
rd53	*226*	*3.16*	*155*	*2.00*	*0.69*	*0.63*
rd73	*335*	*3.78*	*346*	*2.73*	*1.03*	*0.72*
rd84	*857*	*7.01*	*767*	*4.61*	*0.89*	*0.66*
sao2	*815*	*6.20*	*778*	*3.49*	*0.95*	*0.56*
squar5	*256*	*3.65*	*221*	*2.36*	*0.87*	*0.65*
table3	*5014*	*55.17*	*5356*	*53.51*	*1.07*	*0.97*
vg2	*543*	*3.66*	*462*	*2.67*	*0.85*	*0.73*
xor5	*96*	*1.56*	*92*	*1.10*	*0.96*	*0.71*
C17	*40*	*0.78*	*41*	*0.37*	*1.02*	*0.47*
C432	2810	11.91	1329	12.28	0.47	1.03
C1908	4288	17.46	2538	14.17	0.59	0.81
C880	2673	9.53	2377	10.74	0.89	1.13
C1355	2452	8.84	2684	7.84	1.09	0.89
C499	2304	8.64	2702	6.55	1.17	0.76
C2670	4678	8.40	4597	7.17	0.98	0.85
C3540	10101	27.63	6985	22.20	0.69	0.80
C5315	11232	16.03	10201	16.22	0.91	1.01
C7552	12738	24.53	14158	11.77	1.11	0.48
				Average Ratios:	***0.96***	***0.66***

Table 2: *SiMPA versus the conventional flow

Circuit	***Runtime (sec) Conventional***	***Runtime (sec) *SiMPA***	***Ratio***
alu2	*146*	*178*	*1.22*
apex7	*63*	*67*	*1.06*
cm150a	*15*	*22*	*1.47*
cm151a	*11*	*15*	*1.36*
cm162a	*16*	*17*	*1.06*
duke2	*146*	*152*	*1.04*
k2a	*519*	*753*	*1.45*
rot	*215*	*312*	*1.45*
table3	*243*	*265*	*1.09*
C1908	*255*	*291*	*1.14*
C880	*155*	*199*	*1.28*
C1355	*191*	*243*	*1.27*
C3540	*663*	*711*	*1.07*

Table 3: Runtimes

6. References

[CP92]K. Chaudhary, and M. Pedram "A near-optimal algorithm for technology mapping minimizing area under delay constraints," In *Proceeding 29th Design Automation Conference,* pages 492-498, June 1992.

[CLB94]J. Cong, Z. Li, and R. Bagrodia, "Acyclic multi-way partitioning of boolean networks," In *Proceedings of Design Automation Conference,* pages 670-675, June 1994.

[GJ79]M. R. Garey and D. S. Johnson, *Computers and Intractability,* Bell Telephone Lab Inc., 1979.

[HS96] G. D. Hachtel, and F. Somenzi, *Logic synthesis and verification algorithms.*Kluwer Academic Publishers, Norwell, MA, 1996.

[IPFC93]S. Iman, M. Pedram, C. Fabian, and J. Cong, "Finding uni-directional cuts based on physical partitioning and logic restructing," *4th ACM/SIGDA Physical Design Workshop,* pages 187-198, April 1993.

[Ke87]K. Keutzer "DAGON: Technology mapping and local optimization," In *Proceedings of Design Automation Conference,* pages 341-347, June 1987.

[Le82]T. Lengauer, "Upper and lower bounds on the complexity of the min-cut linear arrangement problem on trees," In *SIAM J. Alg. Disc. Meth.,* vol. 3, no. 1, pages 99-113, March 1982.

[LSP97]J. Lou, A. H. Salek, and M. Pedram,"An exact solution to simultaneous technology mapping and linear placement problem," In *Proceedings International Conference on Computer-Aided Design,* pages 671-675, November 1997.

[LSP98]J. Lou, A. H. Salek, and M. Pedram,"An integrated flow for technology remapping and placement of sub-half-micron circuits," In *Proceedings of Asia and South Pacific Design Automation Conference,* pages 295-300, February 1998.

[MFNK96]H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani,"VLSI module placement based on rectangle packing by the sequence-pair," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 15(12), pages 1518-1524, December 1996.

[NFMK96]S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani,"Module placement on BSG-structure and IC layout applications," In *Proceedings of International Conference on Computer-Aided Design,* pages 484-491, November 1996.

[PK92]M. Pedram and E. Kuh, "Bear-FP: A Robust Framework For Floorplanning," In *International Journal of High Speed Electronics,* Vol. 3, No. 1, pages 137-170, 1992.

[Ru89]R. Rudell "Logic synthesis for VLSI design," Memorandum UCB/ERL M89/49, Ph.D. Dissertation, University of California at Berkeley, April 1989.

[Sh93]N. Sherwani, *Algorithms for VLSI Physical Design Automation,* Kulwer Academic Publishers, 1993.

[SLP98]A. H. Salek, J. Lou, and M. Pedram, "Simultaneous floorplanning, technology mapping, and gate-placement for deep submicron designs," *University of Southern California Technical Report,* 1997.

[SSLM92] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli," SIS: A system for sequential circuit synthesis", *Memorandum No. UCB/ERL M92/41,* Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, May 1992.

[St83] L. Stockmeyer, "Optimal orientation of cells in slicing floorplan designs," *Information and Control,* volume 57, pages 91-101, 1983.

[TMBW90]H. J. Touati, C. W. Moon, R. K. Brayton, and A. Wang "Performance oriented technology mapping," In *Proc. Sixth MIT Conf. Advanced Res. VLSI,* pages 79-97, April 1990.

[Ya85]M. Yannakakis, "A polynomial algorithm for the min-cut linear arrangement of trees," In *Journal of the Association for Computing Machinery,* vol. 32, no. 4, pages 950-988, October 1985.