An Integrated Flow for Technology Remapping and Placement of Sub-half-micron Circuits

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

ABSTRACT - This paper presents a new design flow, FPD-SiMPA, and a set of techniques for synthesizing high-performance sub-half micron logic circuits. FPD-SiMPA consists of logic partitioning, floorplanning, global routing, and timing analysis/budgeting steps, followed by technology remapping and detailed placement of the selected logic clusters. The strength of the approach lies in the dynamic programming-based algorithm, SiMPA-D, used for performing simultaneous technology mapping and linear placement of logic clusters. This algorithm generates a set of solutions for each cluster, all of which are non-inferior (in terms of gate area, cutwidth and delay), and hence permits trade-offs between total area (gate plus wiring) and total delay (gate plus wiring). Experimental results from a large number of MCNC benchmarks have proved the effectiveness of the proposed flow.

I. INTRODUCTION

As IC fabrication capabilities reach down to the sub-half-micron, the significance of interconnect delay and area can no longer be ignored. The existing enhancements to both synthesis and physical design tools (such as non-linear delay modeling, custom wire load models, back annotation of calculated delays, statistical wire length estimation) have not been capable of solving the problem. Therefore, trade-offs between logical and physical domains must be addressed in an integrated fashion. Huge business opportunities will be lost unless more revolutionary changes to the design flow are made.

When designing high performance VLSI circuits, designers often find that their designs do not meet the timing and/or area constraints after layout. This predicament is mainly a consequence of the weak interaction between logic synthesis and physical design tools. Logic synthesis, capable of significantly altering the timing and area of the circuits as it proceeds from Boolean minimization to technology mapping, uses a relatively simple model of wires. In contrast, physical design which has accurate wire information from back-end extraction tools is incapable of altering the gate implementation, and hence cannot change the timing/area profile of the circuit drastically.

Design technologies for methodology, design flow, tool, and standard must thereby be advanced so as to produce chip designs incorporating an excess of a 100 million transistors in the same time and using the same number of designers currently consumed in the production of 5 million-transistor chips. Without these advances, the semiconductor and electronics industries will suffer an economic death as they fall off the productivity curve [Be97].

This paper proposes a solution to these design difficulties by introducing a new algorithm, SiMPA-D, in addition to a novel design flow, FPD-SiMPA, which properly takes advantage of SiMPA-D's strength; SiMPA-D (Simultaneous Technology Mapping and Linear Placement Algorithm based on Disjoint Combination) is a polynomial complexity, dynamic programming based algorithm which simultaneously performs technology mapping and linear placement for tree-like circuits by generating an area-delay trade-off curve containing a set of non-inferior area-delay solutions for the tree. Throughout this algorithm, which acts as a bridge between the synthesis and physical design worlds, the exact delay and area values for each solution (considering both gates and wires) are known. SiMPA-D may not find the minimum area implementation as a part of its proposed solution set but, the total area of each solution is proved to be, at worst, a constant factor away from the corresponding optimal area implementation. In that sense, it is an approximation algorithm and not simply a heuristic one with unbounded error. The set of solutions generated by SiMPA-D reveals a near optimal way of trading area for delay and vice versa. This feature has been exploited by FPD-SiMPA (Floorplan Driven Simultaneous Technology Remapping and Linear Placement Algorithm); FPD-SiMPA partitions a given mapped circuit into a set of non-overlapping trees (clusters) each of which is linearly placed. Subsequently, the clusters are floorplanned using the exact area and delay information for each cluster. FPD-SiMPA then identifies those clusters which determine the size and delay of the whole chip. Each of the critical clusters in turn gets remapped and placed using SiMPA-D or SiMPA-E described in [LSP97]. The exact knowledge of delay and area trade-off provided by SiMPA-D makes the outcome of FPD-SiMPA predictable at each step.

This paper is organized as follows: In section II, background and prior works about placement, synthesis and the methodologies combining these two are introduced. Section III describes our proposed methodology in combining placement and technology resynthesis. It also introduces SiMPA-D and other techniques used in this methodology. In section IV, the experimental results are given and analyzed. Concluding remarks and the references used in this work are stated in sections V and VI, respectively.

II. BACKGROUND AND PRIOR WORKS

II.1. Technology Mapping

The problem of technology mapping for general circuit structures is NP-hard [HS96]. In 1987 Keutzer [Ke87] pointed out the similarity between the library binding problem and optimal code generation in a compiler. In his algorithm, the circuit is partitioned into tree subgraphs and each tree sub-graph is mapped using a dynamic programming algorithm which finds the minimum gate area mapping of the tree in polynomial time. This work was later extended by Rudell [Ru89] to minimum delay technology mapping and by Touati et al. [TMBW90] who minimized area mapping under delay constraints. In [CP92], Chaudhary and Pedram presented a dynamic programming algorithm in order to construct the set of all possible mappings of a tree with different area-delay trade-offs. However, neither of the these works considers wiring area or delay during technology mapping.

II.2. Linear Placement

The linear placement problem of a graph has been extensively

This research was supported in part by DARPA under contract number F33615-95-C1627, SRC under contract number 97-DJ-559.

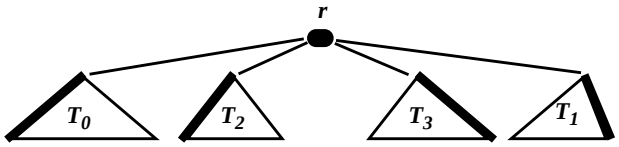

FIG. 1: LA AT EACH STEP OF DYNAMIC PROGRAMMING

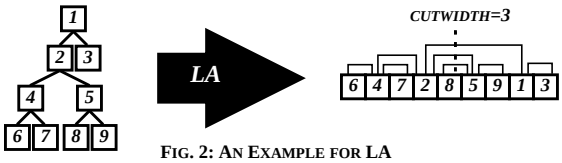

FIG. 2: AN EXAMPLE FOR LA

studied. The MINCUT problem is to find a linear placement that minimizes the maximum cutwidth. This problem is NP-hard for the general graphs [Ya85].

II.2.a Lengauer's Algorithm

Lengauer's algorithm (LA) gives an approximate solution for MINCUT placement of a tree. At worst, the cutwidth of its output placement can be a factor of two away from the minimum cutwidth. LA, which is a bottom-up, dynamic programing based algorithm solves the problem for sub-trees first and then merges these solutions together recursively. At each step of dynamic programming, LA preserves the already constructed placement solution for each subtree, sorts these solutions decreasingly with respect to their cutwidth (ties are broken by giving priority to the balance placements), and places them in that order around the root node as shown in Fig.1. Therefore, the farthest sub-trees (on the right and left) are those with the highest cutwidth. Furthermore, LA places the sub-solutions such that their heavier side (in terms of cutwidth around the root of each sub-solution) faces away from the root [Le82]. Fig.1 shows the heavier sides of each sub-tree with thicker lines.

Example: Fig. 1 shows a simple example for LA.

Lemma 1: For any given tree, the cutwidth of the placement made by LA is at most 2X away from the optimal value. [Le82]

II.2.b Yannakakis's Algorithm

Yannakakis's algorithm (YA), a dynamic programming based bottom-up algorithm, is an exact solution for the tree MINCUT problem. At each level of dynamic programming, YA does not preserve the already constructed placement solution for the subtrees; instead, it merges them together in a very complicated manner in order to achieve the optimal solution. The reader is referred to [Ya85] for more details and to [LSP97] for a brief outline of YA.

Lemma 2: For any given tree, YA finds the placement with the lowest possible cutwidth. [Ya85]

II.3. Combining Synthesis and Physical Design

The problem of eliminating the gap between synthesis and physical design phases can be attacked from two directions. The first approach starts with synthesis techniques and extends down to the world of physical design. The other approach, on the other hand, begins with physical design techniques and extends upwards to the synthesis domain. The methodology proposed in this paper is based on a hybrid approach which enjoys the benefits of both approaches. It is hybrid because it simultaneously works in the synthesis and physical domains.

II.3.a From Synthesis to Physical Design

Pedram and Bhat's work in [PB91] is the first attempt to combine physical design with logic synthesis. They introduced the notion of coupled mapping and placement in order to consider the effect of wires during mapping. The key idea was to generate a "companion" placement during the mapping phase. The placement information is used to evaluate the cost of a gate match during the mapping process. The placement is dynamically updated in order to maintain the correspondence between the logic and layout representations. In the end, a mapped network along with a placement solution is generated. This algorithm assumes that during the bottom-up process of concurrent mapping and placement, the dynamic programming principle holds, which is only an approximation.

SiMPA-E (Simultaneous Technology Mapping and Linear Placement Algorithm for Exact-Area) proposed in [LSP97] is a major extension in Pedram and Bhat's work. SiMPA-E, in contrast to the other technique, performs optimal technology mapping and placement simultaneously for tree-structure circuits. In that work YA and KA were merged together and the technology mapping and linear placement of each tree is done simultaneously and optimally with respect to the total area. SiMPA-E can also be used by FPD-SiMPA, a short description of which comes later in this paper.

II.3.b From Physical Design to Synthesis

Physical design consists of a myriad of techniques designed specifically for the different problems which arise in this field ranging from physical partitioning and floorplanning to placement and routing. Most of the connections already created between this field and synthesis are established through placement. Placement algorithms that apply local netlist transformations, after an initial placement, have the advantage of using accurate information about delay and area for performing a good synthesis. The algorithm proposed in [KSF94] starts from an initial placement followed by timing optimization using fanout buffering and gate resizing transformations. Estimations of the net delays based on the initial placement are used for selecting the most useful transformations. In [LPPD93], the authors proposed to resynthesize the logic in the most congested regions of the chip so as to reduce the routing area. Stenz et al. in [SRRJ97] proposed a technique, which performs iterative timing driven netlist transformations on a companion placement. All of these techniques perform resynthesis on an already mapped and placed circuit while using different types of resynthesis techniques. None of these techniques integrate placement and resynthesis.

III. THE PROPOSED METHODOLOGY

III.1. Introduction

We focus on technology mapping and placement. This paper presents a new technique which combines these two steps and provides the designer with a set of solutions each with a different area and delay trade-off. This technique does not place any restrictions on the rest of the design flow.

FPD-SiMPA consists of two main components, a delay model and a simultaneous placement and mapping technique. This flow is not sensitive to the selection of the delay model and therefore many different models can be used. For the simultaneous placement and remapping part, there are two candidates, SiMPA-D (Simultaneous Technology Mapping and Linear Placement Algorithm based on Disjoint Combination) and SiMPA-E (Simultaneous Technology Mapping and Linear Placement for Exact-Area). SiMPA-D, proposed for the first time in this paper, because of its capability to generate a set of area-delay trade-off solutions, fits very well in this flow. Also, SiMPA-E is a very powerful tool for finding the minimum area implementation of a decomposed tree-like circuit [LSP97]. These algorithms along with our delay model are introduced in the forthcoming sections in more detail.

III.2. The Outline of the Methodology

The steps constituting FPD-SiMPA are shown in Fig.3. A mapped circuit is the input to FPD-SiMPA. That mapping is treated as a starting point and it may later be changed by SiMPA-D/E. Therefore, the method by which the mapping has been generated is not really relevant to our discussion. Any conventional minimum-area or minimum-delay technology mapping method such as [Ru89] and [CP92] is acceptable in this step.

Partitioning the given circuit (called the *primary graph* henceforth) into a set of maximally non-overlapping trees (called *cluster*s

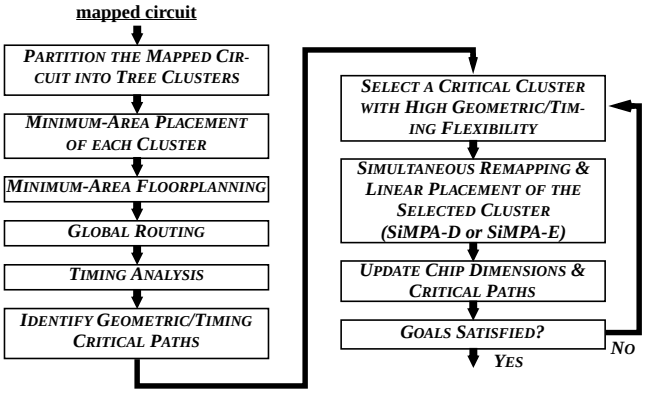

FIG. 3: THE DESIGN STEPS IN FPD-SiMPA

in this paper) is the first task done in FPD-SiMPA. Subsequently, the generated clusters are treated as macro-cells. The interconnections among these macro-cells generate a directed-acyclic graph which is named the *secondary graph*. The tree partitioning can be achieved in a straight-forward manner during a depth-first traversal of the primary graph.

Having finished the partitioning step, FPD-SiMPA finds the minimum-area linear placement for each cluster. We should point out here that the clusters are already mapped and hence their gate-areas are known; therefore, finding a minimum area placement of a cluster means finding the minimum cutwidth placement of that cluster which is simply a MINCUT placement problem (c.f. Section III.4). YA solves this problem exactly and thus finds the absolute minimum total area implementation for each cluster. Beginning with the minimum area placement of clusters is an appropriate starting point for FPD-SiMPA, because it later corrects the timing violations by carefully trading area for lower delay.

Since the gates inside each cluster have been placed, the area and delay for all the clusters is known. This information may now be fed into a floorplanner. The floorplanner is also a minimum-area floorplanner, because a minimum area implementation is desired as the starting point. The outcome of the floorplanning phase on the secondary graph is an actually placed circuit. The next step is the global routing of the floorplanned circuit to construct routing trees for each net present in the secondary graph.

Timing analysis follows global routing. In this step, using the delay model (c.f. in Section III.5) and the global routes, the timing information for each node in the placed circuit is calculated. This information is later used to identify the timing-critical clusters which need to be resynthesized for further optimization.

To proceed to the next step in FPD-SiMPA, we first need to define the following terms; delay-critical, height-critical, and width-critical clusters:

Definition 1: Using the delay model, the critical path(s) of the placed circuit is (are) found. All the clusters located on the critical timing path(s) are called *delay-critical clusters*. Obviously, each cluster can occur only once on the critical path due to the acyclic nature of the initial circuit.

Definition 2: For each row, we can sum up the width of all the clusters located on each row to find the row widths. The clusters located on the longest row(s) are *width-critical clusters*. For example, *{T1, T2, T3}* is the set of all width-critical cells in the example given in Fig. 4.

Definition 3: *Height-critical clusters* are all the clusters building up the height of the chip. For example in Fig.4, *{B2, M2, T3}* is the set of all height-critical clusters in that placement.

FPD-SiMPA next finds the sets of clusters which are located on geometrical and timing critical paths. The "cluster selection" procedure picks one critical cluster at a time. The critical cluster is chosen

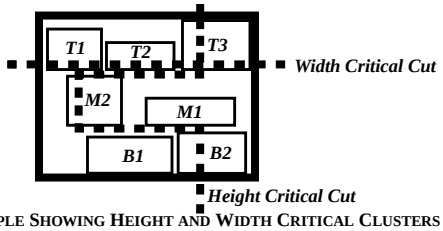

FIG. 4: AN EXAMPLE SHOWING HEIGHT AND WIDTH CRITICAL CLUSTERS

by a heuristic which is dependent on the objective function, allowing different strategies to be used for different problems. We will not discuss possible heuristics for different cases, in Section III.3, we will discuss the behavior and characteristics of the general problem which can be used as helpful hints to engineers responsible for devising a heuristic for an application.

The selected cluster is then decomposed into 2-input NAND and inverter gates, and then SiMPA-D/E is run on it. The output of SiMPA-D/E is a set of non-inferior solutions represented in the form of an area-delay trade-off curve. This curve is visited by a "solution selection" procedure which examines all available solutions and selects the best, according to the objective function. Fig.5 shows the internal flow of this step.

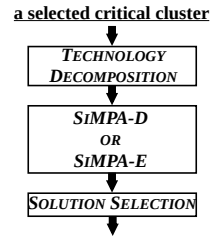

Fig. 5

After choosing a satisfactory solution for the chosen critical cluster, the size and timing information for the whole chip are updated to reflect the changes in the cluster.

Finally, FPD-SiMPA decides whether the convergence has occurred and/or the goals have been achieved. If the circuit needs more optimization, it is sent back to the cluster selection step, where another critical cluster is selected and the same resynthesis process is repeated.

The next sections will present further detail and analysis on some steps of FPD-SiMPA.

III.3. Critical Cluster Selection

Few simple observations are helpful in devising and employing a good cluster selection procedure. A good performance in this step allows FPD-SiMPA to be more effective in performing the area/delay trade-off.

Let's call the sets of delay, height, and width critical clusters, *D*, *H*, and *W*, respectively.

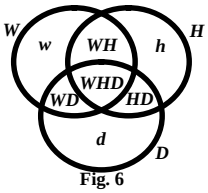

Fig. 6

Observation 1: There is at least one cluster in common between *H* and *W*.

Observation 2: Each member of *D, H* and *W* which is not a member of the other sets (*d, h,* and *w* sections in Fig. 6) can be optimized independently to a first order approximation. For example, reducing the height (cutwidth) of a member of *h* will not change the set of width critical clusters *W,* unless the cluster whose height was reduced happens to get too wide and becomes a member of *W.* However, this situation can be avoided by considering the available space on the left and right sides of the candidate cluster.

Observation 3: For the clusters located in *WH, HD,* and *WD* sections, optimization may still be possible without sacrificing the other parameter. The only cases in which we have to sacrifice area/delay for meeting the delay/area design constraints are the clusters located in the *WHD* section of Fig.6.

Observation 4: The critical clusters which are more relaxed with respect to the other parameters are more likely to improve the design parameters without any penalty. For example, a delay-critical cluster which has lots of free space in its neighborhood is more likely to be sped up by enlarging its area to occupy the adjacent free regions

without increasing the chip area.

Many different cluster selection strategies can be conceived based on the above observations. For example, we use a simple method which starts by selecting clusters from *w, h,* and *d* in the decreasing order of their flexibility to the other parameters. If the design constraints are not satisfied as such, we process the other clusters and try to improve the design parameters for the chip with as small penalty as possible.

III.4. Area Calculation

The total area for one-dimensional standard-cell layout is given as following:

$A = W . (h + \beta . c)$ where,

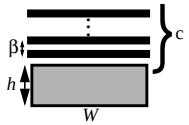

Fig. 7

W is the summation of the widths of all the cells used in the design: $W = \sum_{cell_i} width(cell_i)$ where,

h is the cell height, a constant determined by the ASIC library,

β is the minimum distance between the centers of two adjacent wires:

$\beta = (minWireWidth + minWireSpacing)$

and c is the maximum cutwidth (also known as cut-density) of the design.

Lemma 3: The total width of any linear placement for a mapped circuit is the summation of the widths of its gates.

Theorem 1: For a mapped tree circuit, YA gives the minimum total area linear placement implementation.

Proof: Regardless of the placement order, the total width W is constant, Lemma 3. In a given standard-cell design style, h and β are constants too, therefore, for a given mapped tree circuit, a linear arrangement has the minimum total area iff it has the minimum cutwidth. This minimum cutwidth linear placement is found by YA. ■

Theorem 2: The placement given by LA for a given mapped tree circuit T is at most 2X away from the corresponding minimum area implementation.

Proof: Similar to the previous proof, W, h, and β are constants. Call the minimum possible cutwidth of a tree c_{min}. According to Theorem 1, the total area of the minimum area implementation of the circuit is $A_{min} = W . (h + \beta . c_{min})$. The cutwidth of the placement built by LA (c_{LA}) is at most 2X larger than c_{min}, Lemma 1. Thus, $A_{LA} = W . (h + \beta . c_{LA})$, is at most 2X away from the optimum value (A_{min}). This upper bound is reached if $h=0$ and T is a balanced tree. ■

Theorem 3: Two trees T and T' are similar except in certain minimal subtrees; call them $s_1, s_2, ..., s_n$ in T and $s_1', s_2', ..., s_n'$ in T'. If LA finds placements with lower or equal cutwidth for all $s_1, s_2, ..., s_n$ compared to their counterparts in T', it also finds a placement with lower or equal cutwidth for T.

Proof: The proof is omitted due to the space limitations.

III.5. Delay Calculation

The delay model used in this work is similar to the one proposed in [LSP97]. Interested readers are referred to that work for the details.

III.6. SiMPA-D

SiMPA-D uses KA and LA to do simultaneous technology mapping and linear placement for delay minimization. The dynamic programming (DP) nature of both KA and LA facilitates coupling these two design steps. At every stage of the bottom-up decision making in DP, the solution is built from the solutions to the sub-problems which are already known. All the solutions to the sub-problems are disregarded except for the one (assuming a single parameter cost function) which minimizes the cost function. This process of pruning the inferior solutions at every stage of DP provides the polynomial complexity of the algorithm.

Although the objective is to minimize the total area, total area is not a sufficient cost function in this DP algorithm by which to measure the cost of the solutions and eliminate them accordingly. It is easy to see that when we place two sub-solutions, S_1 and S_2, next to each other, the cutwidth of the resulting placement is a function of *MAX* operation over the cutwidths of the two sub-solutions. This property causes the violation of the dynamic programming principle for total area. In other words, with respect to that cost function, the final optimal solutions does not consist of the optimal solutions of the sub-problems. Fortunately, the principle of dynamic programming holds when we define the cost function as a two-tuple of (cutWidth, gateArea). In SiMPA-D, since LA is used, cutwidth is the total number of horizontal wires crossing any vertical line in a placement. However, in SiMPA-E where YA is used, cutwidth is the cost function used in YA which is a generalized form of cut capacity, CCF.

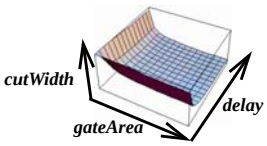

Fig. 9
3-D SOLUTION CURVE

Since SiMPA-D uses LA, the final placement of a tree is a simple combination of the placement of each sub-tree (sub-solution) which is different from what happens in YA and SiMPA-E. That means while constructing the final placement, the inside structure of the sub-problems are not changed. Therefore, the calculated timing information for each sub-solution remains valid through SiMPA-D (more details are given later). This guarantees that the dynamic programming principle in SiMPA-D for delay is valid (subject to the unknown-load problem). This opportunity lets us use a three parameter cost function (cutwidth, gate area, delay) and perform optimization with respect to both geometry and timing at the same time. At every node during this DP-based algorithm, a three-dimensional curve (Fig.9) of the solutions is calculated and the inferior points (defined below based on the definition of the cost function) are dropped out to maintain the polynomial complexity of the algorithm.

Definition 4: In SiMPA-D, we define the solution S_1 to be inferior with respect to S_2 iff the following condition holds:

$gateArea(S_1) >= gateArea(S_2)$ **AND** $cutWidth(S_1) >= cutWidth(S_2)$ **AND** $delay(S_1) >= delay(S_2)$

where, *cutWidth* is the maximum number of horizontal wires of the placement crossing any vertical line.

SiMPA-D starts from the primary inputs (PI's) and using a postorder traversal tries all the possible library matches at every node of the decomposed tree circuit. At each node n, a set of non-inferior solutions are stored for use in the subsequent steps. For a match, say m, SiMPA-D retrieves all the non-inferior solutions for the subtrees connected to the inputs of m, where every solution for a subtree contains the corresponding mapping and linear placement (using LA). The set of placements for any combination of sub-solutions along with m when passed to LA generates a placement solution for n, where for every generated solution the corresponding cutwidth, gate area, and delay can be easily extracted and stored in n. In general, for every combination of the input subtree solutions, LA should be called once. However, the number of trial combinations may be significantly reduced (it becomes polynomial) with the use of a suitable ordering scheme similar to the one given in [CP92]. Having calculated all the necessary solutions for n, SiMPA-D will then drop out the inferior solutions in the solution set of node n and the resulting solution set is stored for future use by SiMPA-D.

Due to the availability of all the physical information in SiMPA-D, the total delay of a circuit (gate plus wire delays) can be calculated accurately at every step and the wire load and delay can be captured exactly using the placement generated by LA. The following example will confirm the validity of this claim (Fig.11). During the bottom-up manipulation process, the placement information is known for match m and all of its input gates, i.e., $input_1$, $input_2$ and $input_3$. Therefore,

```
INPUT: a tree network N, a library L
OUTPUT: a mapped and linearly placed netlist of N
1. Decompose N using a technology decomposition algorithm
2. Perform a Depth-First-Search from PO to PI in N
3. For each node n in the reverse DFS order
4.      For every match m of n
5.        gateArea = sum of accumulated gate area of all inputs of m
          + gate area of m
6.          cutWidth= cutwidth of the linear placement generated by LA
            for m and all the connected subtrees.
7.          Update the delay values of all inputs because of the load
          change by the match and the placement
8.          delay=ouput signal arrival time of m in the current impl.
9.        Store the solution <gateArea, cutWidth, delay> in 3-D curve of n
10.     Prune the inferior solutions in 3-D curve of n
11. Select the best solution from the PI, and recursively select the
    solutions of all the inputs which lead to this best solution.
```

Fig. 10: PSEUDO CODE FOR SIMULTANEOUS TECHNOLOGY MAPPING AND LINEAR PLACEMENT ALGORITHM BASED ON DISJOINT COMBINATION (SIMPA-D)

the topology and lengths of $wire_1$, $wire_2$ and $wire_3$ are known. The capacitance and resistance values for these wires can be calculated exactly, and the loads on $input_1$, $input_2$ and $input_3$, which consist of the input load of match *m* and the corresponding wire load, are precisely known. Therefore, the gate delays of these inputs can be calculated exactly. Moreover, the delays of $wire_1$, $wire_2$ and $wire_3$ can be precisely calculated using the Elmore Delay Model. So the arrival times at the inputs of current match are known exactly.

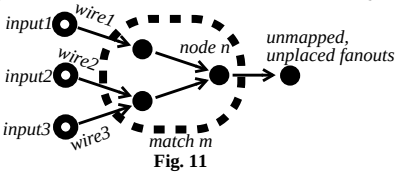

Fig. 11
Total delay calculation

The exact calculation of the total delay (due to the availability of exact wire load and wire delay) is however subject to the "unknown load" problem; that is, the actual loading at the node is unknown since its fanout nodes have not been mapped yet. Consequently in this paper, all discussions regarding delay are also subject to the unknown load problem. This problem is summarized as follows; during each step of dynamic programming, the mapping and placement information of the fanouts of the current node are unknown. Therefore, the fanout load of the current node *n* and the gate delay of match *m* cannot be determined accurately (Fig. 11). Thus, we use a heuristic solution similar to that presented in [Ru89] and [TMBW90] for performing delay re-calculation as a means for solving this problem. During each step of DP, we assume a constant load ahead which is the input capacitance of a 2-input *NAND* gate and the capacitance of a wire segment whose length is estimated by the fanout count of the node [VP95]. We can calculate the load change at the fanins of a match since they are already mapped and placed. We update the delay of all the fanins of this match ($input_1$, $input_2$ and $input_3$ in Fig.11) in order to use the correct load information. Therefore, the error generated by the unknown load effect in the previous step will not propagate to the next step.

YA cannot be used in SiMPA-D, because as mentioned in Section II.2.b, while merging the placements of the subproblems YA may break a subtree and place the rest inside the generated gap in order to minimize the cutwidth. In this case, the wire load and the wire delay of the broken subtree along with the total delay will be drastically changed. Consider the following example in Fig.12, where we have 3 subtrees T_0, T_1 and T_2. We assume the mapping and the placement for each subtree have already been done, and a 3-D curve including a

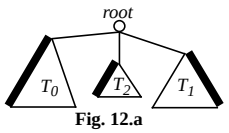

Fig. 12.a

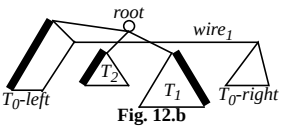

Fig. 12.b

set of non-inferior solutions for each subtree has been generated. YA may break the tree T_0 and put T_1 and T_2 in its gap to achieve the minimum cutwidth. As we can see, $wire_1$ which connects the left and right parts of T_0, will then be elongated. As a result, the delay calculation based on the previous length of $wire_1$ is no longer valid. This violates the principle of dynamic programming since the delay value for subtree T_0 may change depending on the structure of other subtrees, T_1 and T_2. Throughout its calculation of the final result, YA does not guarantee to preserve the inferiority of some solutions with respect to delay. Therefore, delay cannot be used as an optimization parameter in YA. In contrast, as previously mentioned, while combining the sub-solutions LA does not modify their internal placement. This property makes LA a suitable placement algorithm when delay parameter is used to distinguish between inferior and non-inferior solutions. Another important characteristic of LA is that it is an approximation algorithm and is guaranteed to find a placement which has at most twice the cutwidth of the optimal cutwidth placement for a specific mapping. This implies that for every solution the area penalty due to delay consideration is bounded. Indeed, the difference between YA and LA is less than or equal to one track in most of our experiments making the area penalty negligible.

During SiMPA-D, inferior points can be safely discarded without losing non-inferior solutions. Therefore, given the upper bound(s) for any arbitrary subset of {tracks, gate area, delay} all the points in the 3-dimensional curve with parameters less than the imposed upper bound(s) constitute the non-inferior implementations of the circuit satisfying the given constraint.

Theorem 4: Every point in the 3-D curve is at most 2X away, in terms of total area, from the total area of its corresponding optimal placement.

Proof: The proof follows from Theorem 2 and the method by which SiMPA-D calls LA and prunes the inferior points. ■

Theorem 5: For any area-optimal solution *S* (which is a solution found by SiMPA-E), there exists a solution *S'* on the 3-D curve with a lower or equal gate area and with a total area that is at most 2X larger than that of *S*.

Proof: If these two solutions have similar mappings, the proof directly follows from Theorem 2. Assume *S"* to be a solution built by running LA on the mapping of *S*. The total area of *S"* is at most 2X larger than *S* according to Theorem 2. The mapping of *S"* differs from the mapping of *S'* in certain minimal sub-trees. These mappings and their corresponding LA placements have been dropped during SiMPA-D by their corresponding solutions in *S'* which possess lower gate area and cutwidth. Therefore, according to Theorem 3, the total area of *S'* is smaller than or equal to the total area of *S"*, (and hence within 2X of *S*). ■

Theorem 6: SiMPA-D is a polynomial complexity algorithm.

Proof: At every recursive step of SiMPA-D, the maximum number of non-inferior solutions is a polynomial function of *n* (the number of vertices in the decomposed tree as follows). Assuming the width of the gates in the library are multiples of a constant unit and the widest gate has a width of *W*, the maximum number of distinct values for cutwidth is *n-1*, and the gate area is *nW*. For every combination of gate area and cutwidth values, SiMPA-D keeps one solution at most (the one with the least delay). Therefore, the total number of the points in any 3-D curve is bounded by *n.(n-1).W*. Since the runtime of SiMPA-D is a polynomial function of the number of solution points, the total runtime complexity is polynomial as well. ■

III.7. SiMPA-E

The simultaneous technology mapping and linear placement algorithm for exact area minimization, SiMPA-E, first proposed in [LSP97], is a superset of KA and YA. The exactness of KA and YA is preserved in SiMPA-E making it an exact solution for minimum total area. For a detailed discussion the reader is referred to [LSP97].

Theorem 7: The final curve built by SiMPA-E includes all the cutwidth and gate area non-inferior solutions to the problem. [LSP97]

IV. EXPERIMENTAL RESULTS

FPD-SiMPA was implemented in the SIS environment [SRRJ97]. We compare the experimental results from conventional flow and FPD-SiMPA using SiMPA-D for delay minimization. Both flows use the 4-parameter delay equation for calculating gate delays [LSP97] and use the Elmore delay model for wire delay calculation. The library we use is a CASCADE standard cell library (0.5u HP CMOS process). The area and delay reported here are the total chip area, and the total chip delay after detailed routing.

The input to both flows is a technology mapped netlist in which the delay is optimized by the SIS mapper. The conventional flow uses GordianL and Domino for placement, TimberWolf for global routing and YACR for detailed routing. In the FPD-SiMPA flow, we cluster the netlist and linearly place each cluster using Yannakakis' mincut linear placement algorithm. BearFP is called to floorplan the clusters and find the minimum area floorplan solution. After identifying the timing critical paths, we choose the candidate clusters for re-mapping and re-placement using SiMPA-D to minimize the delay. The run time of FPD-SiMPA is comparable to that of conventional flow.

Our experimental results show that FPD-SiMPA successfully improves the delay and area by as much as 54% and 53%, respectively. On average, we improved the delay by 19% while reducing the area by 13%. This improvement comes from both our design flow and the strength of our simultaneous technology mapping and linear placement algorithm. For some circuits, such as C7552, we were able to reduce the delay by more than 50% while keeping the area increase to within 10%, whereas for many other circuits, we were able to reduce the delay and area simultaneously. However, there are some circuits, such as frg2, for which we could improve neither delay nor area. We believe the reason to be the loss of our ability to perform optimization across tree boundaries. For some circuits, however, optimization across the tree boundaries may bring a significant improvement in the circuit quality. We are currently working on a new logic-replicating partition technique, which will enable us to perform optimization across the tree boundaries.

	Conventional		FPD-SiMPA		Ratios	
CIRCUIT	AREA	DELAY	AREA	DELAY	AREA	DELAY
alu4	*4267136*	*20.93*	*4064043*	*17.32*	*0.95*	*0.83*
apex6	*4220940*	*11.63*	*3280464*	*9.18*	*0.78*	*0.79*
dalu	*8793378*	*27.81*	*9007605*	*26.30*	*1.02*	*0.95*
Z5xp1	*656750*	*11.59*	*513590*	*9.18*	*0.78*	*0.79*
Z9sym	*995196*	*8.41*	*980045*	*3.93*	*0.98*	*0.47*
frg2	*4967865*	*10.98*	*5294103*	*11.46*	*1.07*	*1.04*
sao2	*908283*	*9.41*	*620806*	*5.80*	*0.68*	*0.62*
C432	*2511441*	*13.98*	*1172676*	*14.29*	*0.47*	*1.02*
C880	*2740674*	*12.72*	*2283125*	*12.19*	*0.83*	*0.96*
C499	*3099603*	*9.56*	*3035760*	*8.48*	*0.98*	*0.89*
C1908	*4159199*	*17.36*	*2912016*	*12.78*	*0.70*	*0.74*
C1355	*2267650*	*9.81*	*2739392*	*8.80*	*1.21*	*0.90*
C3540	*10057735*	*27.36*	*7378446*	*20.67*	*0.73*	*0.76*
C5315	*11579390*	*15.99*	*9660618*	*13.66*	*0.83*	*0.85*
C7552	*12995712*	*24.53*	*14246148*	*11.29*	*1.10*	*0.46*
				Average:	*0.87*	*0.81*

V. CONCLUSION

This paper presents a novel algorithm, SiMPA-D, which performs simultaneous technology remapping and linear placement for tree-structured circuits while targeting minimum total chip delay and/or area. The proposed algorithm which uses a dynamic programming technique generates a three-dimensional area-delay trade-off curve of gate area, cutwidth, and delay. This paper also describes a new methodology, FPD-SiMPA, which exploits the above algorithm to synthesize high-performance sub-half micron logic circuits. This methodology is capable of controlling the trade-off between area and delay, and produces circuit implementations with highly predictable performance characteristics. Experimental results proved the effectiveness of the proposed flow and SiMPA-D.

VI. REFERENCES

[Be97] R. Bushroe "Panel: physical design and synthesis: merge or die," In *Proceeding 34th Design Automation Conference*, pages 238-239, June 1997.

[CCh84] F. R. K. Chung, "On optimal linear arrangements of trees," In *Comput. Math. Applic. 10*, pages 43-60, 1984.

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

[GKTP95] R. Gupta, B. Krauter, B. Tutuianu, and T. Pileggi," The Elmore delay as a bound for RC trees with generalized input signals," in *Proceedings of 32nd Design Automation Conference*, pages 364-369, 1995.

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

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

[KSF94] L. N. Kannan, P. R. Suaris, and H. Fang, "A methodology and algorithms for post-placement delay optimization, "In *31st ACM/IEEE Design Automation Conference*, 1994.

[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.

[LPPD93] S. Liu, K. Pan, M. Pedram, and A. M. Despain, "Aleviating routing congestion by combining logic resynthesis and linear placement," In *European Conference on Design Automation*, pages 578-582, 1993.

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

[PB91] M. Pedram, and N. Bhat "Layout driven technology mapping," In *Proceedings of the 28th Design Automation Conference,* pages 99-105, June 1991.

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

[SRRJ97] G. Stenz, B. M. Riess, B. Rohfleisch, and F. M. Johannes, "Timing driven placement in interaction with netlist transformations," In *International Symposium on Physical Design*, pages 36-41, April, 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.

[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.

[VP95] H. Vaishnav, and M. Pedram, "Logic extraction based on normalized netlengths," In Proceedings *of the International Conference on Computer Design*, October 1995.

[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.