Transistor Placement for Noncomplementary Digital VLSI Cell Synthesis

MICHAEL A. RIEPE
Magma Design Automation
and
KARET A. SAKALLAH
University of Michign

There is an increasing need in modern VLSI designs for circuits implemented in high-performance logic families such as Cascode Voltage Switch Logic (CVSL), Pass Transistor Logic (PTL), and domino CMOS. Circuits designed in these noncomplementary ratioed logic families can be highly irregular, with complex diffusion sharing and nontrivial routing. Traditional digital cell layout synthesis tools derived from the highly stylized "functional cell" style break down when confronted with such circuit topologies. These cells require a full-custom, two-dimensional layout style which currently requires skilled manual design. In this work we propose a methodology for the synthesis of such complex noncomplementary digital cell layouts. We describe a new algorithm which permits the concurrent optimization of transistor chain placement and the ordering of the transistors within these diffusion-sharing chains. The primary mechanism for supporting this concurrent optimization is the placement of transistor subchains, diffusion-break-free components of the full transistor chains. When a chain is reordered, transistors may move from one subchain (and therefore one placement component) to another. We will demonstrate how this permits the chain ordering to be optimized for both intra-chain and inter-chain routing. We combine our placement algorithms with third-party routing and compaction tools, and present the results of a series of experiments which compare our technique with a commercial cell synthesis tool. These experiments make use of a new set of benchmark circuits which provide a rich sample of representative examples in several noncomplementary digital logic families.

Categories and Subject Descriptors: B.6.3 [**Logic Design**]: Design Aids—*Automatic synthesis*; B.7.2 [**Integrated Circuits**]: Design Aids—*Layout*; *placement and routing*; J.2 [**Physical Sciences and Engineering**]: Electronics; J.6 [**Computer Aided Engineering**]: Computer Aided Design (CAD)

General Terms: Algorithms, Design, Experimentation, Performance

Additional Key Words and Phrases: Cell Synthesis, digital circuits, noncomplementary circuits, transistor placement, transistor chaining, Euler graphs, sequence pair optimization, benchmark circuits

Work supported by the Defense Advanced Research Projects Agency (DARPA) under ARPA/ARO contract DAAH04-94-G0327.

Authors' addresses: M. A. Riepe, Magma Design Automation, Inc., 2 Results Way, Cuppertino, CA 95014; email: riepe@magma-da.com; K. A. Sakallah, Department of Electrical Engineering and Computer Science, University of Michigan, 1301 Beal Ave., Ann Arbor, MI 48109-2122; email: karem@eecs.umich.edu.

1. INTRODUCTION

Library cells are the lowest level of the digital VLSI design hierarchy. Clearly, the quality of cells in a given library has a direct impact on the quality of the final design. The cells must be designed to be compact and fast, with minimized power and parasitics, and with careful attention paid to requirements on the physical architecture of the cells as viewed by the higher-level placement and routing tools. Automated synthesis techniques have found limited application at the cell level because existing tools are unable to match the quality of human designed cells. For this reason cells are often designed by hand, requiring a significant investment in manpower.

An additional difficulty lies in the fact that the lifetime of a typical cell library may be as short as 1 or 2 years. Compaction techniques may be used to migrate a cell library to a new process technology if little more than a linear shrink is required, but this is unlikely to extend the lifetime for more than one or two process generations before the loss in performance necessitates a complete redesign of the library. These problems are only becoming worse as device geometries shrink into the deep submicron regime.

In order to account for deep submicron effects when designing large chips, ever closer interaction is required between front-end logic synthesis tools and back-end placement and routing tools, power and delay optimization tools, and parasitic extraction tools. In order to enable this interaction, cell libraries must become more flexible. Multiple versions of each cell with different drive strengths are required. It may even be necessary to support versions of cells in different logic families with different power/delay tradeoffs. In leading-edge integrated circuit designs, static CMOS and dynamic domino CMOS tend to be dominant, but there is a growing need for circuits implemented in high-performance logic families such as Cascode Voltage Switch Logic (CVSL) and Pass Transistor Logic (PTL). In addition, synchronizing elements such as latches and flip-flops tend to be designed using custom analog sense amplifier techniques.

In addition to the need for families of cells which are parameterized in terms of their electrical behavior, it has been demonstrated that standard-cell placement and routing tools are able to obtain significantly higher routing quality if they have the ability to choose between multiple instances of cells with a wide variety of pin orderings. In one experiment [Lefebvre et al. 1997], an average reduction in the number of routing tracks of 10.8% was demonstrated over five benchmark circuits.

It seems clear that as the number of cells in a typical cell library grows from the hundreds into the thousands, a dramatic increase in designer productivity will be required, necessitating a move toward more automated cell synthesis techniques. Several authors, in fact, advocate a move completely away from static cell libraries as we know them, toward a system which permits the automated synthesis of cells on demand [Lefebvre et al. 1997; Burns and Feldman 1998]. This would permit logic synthesis tools to request specific logic decompositions, doing away with the traditional technology mapping step; standard-cell and datapath placement and routing tools to request cells with an exact pin

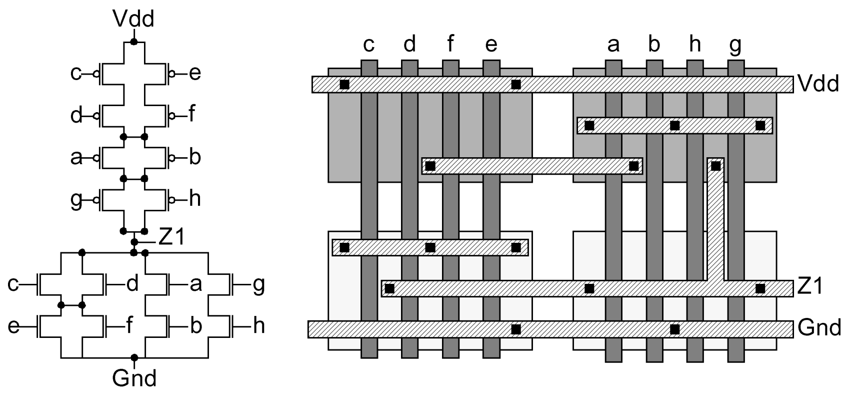


Fig. 1. An example of a complex gate designed in the "functional cell" style of Uehara and Van-Cleemput [1981].

ordering; interconnect optimization tools to request cells with specific input and output impedance values; and power optimization tools to request cells, perhaps from one of several different logic families, with specific power/delay tradeoffs.

Such an on-demand cell synthesis system will require effort on many fronts:

(1) Automated transistor schematic generation (constraint-driven logic family selection, netlist creation, and transistor sizing);
(2) automated cell geometry synthesis;
(3) automated cell testing and characterization; and
(4) development of enabling logic synthesis, placement and routing, and power/delay optimization technology.

In our work we address the second item in the above list: the fully automatic synthesis of library cell mask geometry. The input specification consists of a sized transistor-level schematic, a process technology description (design rules, parasitics, etc.), and a description of the constraints imposed by the higher-level placement and routing environment. We refer to this last item as the *cell template*. A list of common cell template constraints are enumerated in Lefebvre et al. [1997].

1.1 Motivation

The CMOS cell synthesis problem has a rich history going back approximately 20 years. Most of this research has centered on a formulation of the problem which was referred to as the "functional cell" in a seminal paper by Uehara and VanCleemput [1981]. In this style, an example of which is given in Figure 1, the transistors take on a very regular row-based structure. They are arranged in a linear fashion so as to minimize the number of gaps in the diffusion islands (so called "diffusion breaks"). We will refer to layouts in this style as one-dimensional, or 1D, layouts.

The synthesis of 1D layouts can be formulated as a straightforward graph optimization problem: to find a minimal dual Euler-trail covering for a pair of

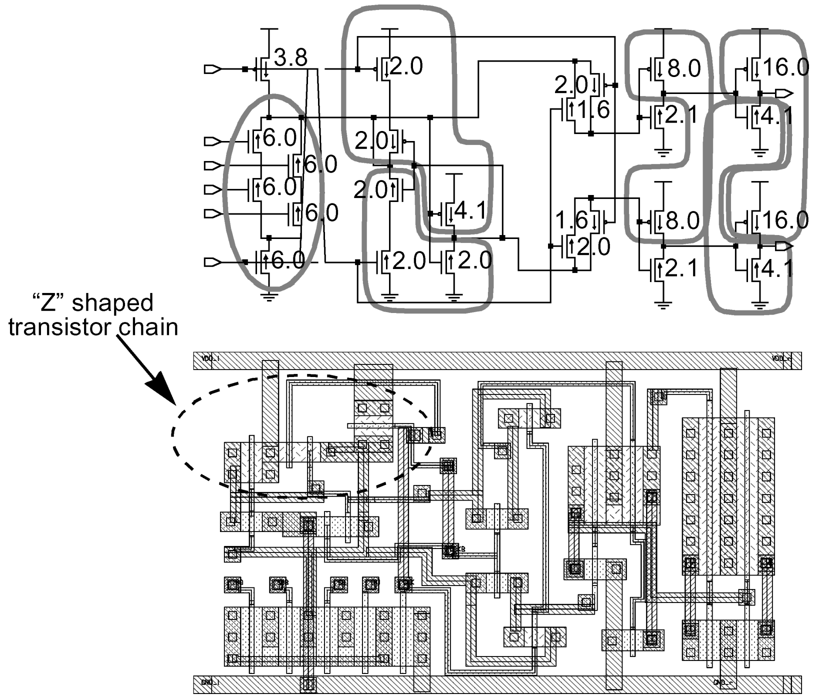


Fig. 2. A manually designed cell showing complex two-dimensional layout structure.

dual series-parallel multigraphs. Uehara and VanCleemput [1981] developed an approximate solution technique for this problem, while Maziasz and Hayes [1992] presented the first provably optimal algorithms.

A major drawback of the 1D layout style is that it applies directly only to fully complementary nonratioed series-parallel CMOS circuits. Several significant systems have extended this style to cover circuits with limited degrees of irregularity. Among these are *C5M* [Burns and Feldman 1998], *Excellerator* [Poirier 1989], *LiB* [Hsieh et al. 1991], and *Picasso II* [Lefebvre and Skoll 1992]. Dynamic CMOS circuits, if the p-channel pull-up transistors are ignored, can be optimized using a simpler single-row 1D formulation—a good summary is presented by Basaran [1997].

Despite the elegant formulation presented by the one-dimensional abstraction, it must break down at some point. When designing aggressive, high-performance circuits, the designer may call upon logic families such as domino-CMOS, pseudo-NMOS, CVSL, and PTL. These provide the designer with different size/power/delay tradeoffs than are available with a static CMOS implementation. However, these noncomplementary ratioed logic families often result in physical layouts which are distinctly two-dimensional (2D) in appearance.

An example of such a circuit is shown in Figure 2. The example is a hand-designed mux-flip flop standard cell implemented in a complementary GaAs process [Bernhardt et al. 1995]. This example demonstrates a number of properties which deviate from the standard one-dimensional style:

(1) It is highly irregular. Some regularity is present in the rows, or "chains," of merged transistors, but these chains are of nonuniform size and are not arranged in two simple rows.
(2) There are instances of complex diffusion sharing, such as the "Z"-shaped structure in the upper-left corner.
(3) The transistors are given a wide variety of channel widths.
(4) The routing is nontrivial.
(5) The port structure required by the back end placement and routing tools must be taken into account.

1.2 Prior Work

A variety of approaches have been taken to address the 2D cell synthesis problem. Tani et al. [1991] and Gupta and Hayes [1997] discussed a style in which two-dimensional layouts are formed from multiple one-dimensional rows. The former presented a heuristic technique based on min-cut partitioning while the latter presented an exact formulation, called *CLIP*, based on integer linear programming. To distinguish this style from non-row-based two-dimensional styles, we refer to it as being *1-1/2 dimensional*.

Xia et al. [1994] developed a method for BiCMOS cell generation. The method groups MOS transistors into locally optimal chains which behave as fixed blocks in the design. Bipolar transistors are treated individually and are given a fixed area with a flexible aspect ratio. A branch and bound algorithm is used to explore a slicing tree-based floorplanning model of the circuit to find a floorplan of minimal area.

It is also relevant to discuss work in analog circuit placement and routing in this context. One such system is *Koan/Anagram* by Cohn et al. [1994]. This system uses simulated annealing to find a placement of analog components, simultaneously seeking to optimize device connectivity through arbitrary geometry sharing, while satisfying design rule constraints as well as analog constraints such as device symmetry and matching. It then uses a custom area router with aggressive rip-up and reroute capability to make the remaining required connections.

Fukui et al. [1995] developed the first system for true two-dimensional digital transistor placement and routing. It uses a simulated annealing algorithm to find good groupings of transistors into diffusion-shared chains and performs a greedy exploration of a slicing structure to find a two-dimensional virtual grid floorplan. It uses a symbolic router to perform detailed routing, and a final compaction step to permit transistors within chains to slide into locally optimal positions.

In a more recent work by the same group, Saika et al. [1997] presented a second tool which operates by statically grouping the transistors into maximally sized series chains and then locating a high-quality one-dimensional solution in order to form more complex chains. Then a simulated annealing algorithm is used to modify this linear ordering by placing the diffusion-connected groups onto a two-dimensional virtual grid. Routing is done by hand.

Two other efforts which are contemporary with our own [Riepe 1999; Riepe and Sakallah 1999] have recently published results. Vahia, Askar, and Ciesielski [Vahia and Ciesielski 1999; Oskar and Ciesielski 1999] described a fast greedy constructive algorithm for the placement of transistor chains, while Serdar and Sechen [1999] presented an efficient annealing-based solution. Both groups showed very compact layouts of several difficult circuits. However, both limited their search to the placement of statically ordered transistor chains, and both made use of manual routing. In addition both works assumed multiple layers of metal interconnect, allowing them to work with diffusion-limited placements and thus effectively decoupling the placement and routing problems.

1.3 Proposed Methodology

In this section we present the architecture for a 2D cell synthesis system which targets complex noncomplementary digital MOS circuits. We begin with a precise definition of the problem and then outline the top-level flow in our proposed optimization methodology.

One can view our problem definition, which is elaborated below, as a relaxation of the highly stylized rules which were adopted by the 1D “functional cell” style for the optimization of dual CMOS circuits. Our goal is to attempt to emulate the creativity used by an expert designer by permitting selected new degrees of freedom in the layout style while keeping the problem of manageable complexity.

The following four points summarize our assumptions when defining the cell-level transistor placement and routing problem:

(1) The input to the system is a sized transistor netlist, the process design rules and technology parameters, and a description of the cell layout style (“template constraints”).
(2) Transistor source/drain geometry sharing is encouraged, but is not the primary optimization objective. Obtaining a routed cell of minimum area is the primary objective.
(3) Individual transistors, or chains of source/drain connected transistors, may be placed in any position or orientation to optimize the objective as long as the design rules and template constrains are satisfied.
(4) Routing may be performed in two layers: polysilicon and first-level metal. There is no preferred direction for routing in either layer.

As our top level optimization framework, we adopt the flow diagrammed in Figure 3. In the first step clusters of transistors are formed, each representing a set to be composed into a single merged structure, or *transistor chain*. Clusters of size one represent degenerate chains of individual transistors. These clusters are passed to the placement step (step 2), where they are assigned a chain ordering and broken into diffusion-break free components called *subchains*, and each resulting subchain is assigned a position and an orientation. Finally, the

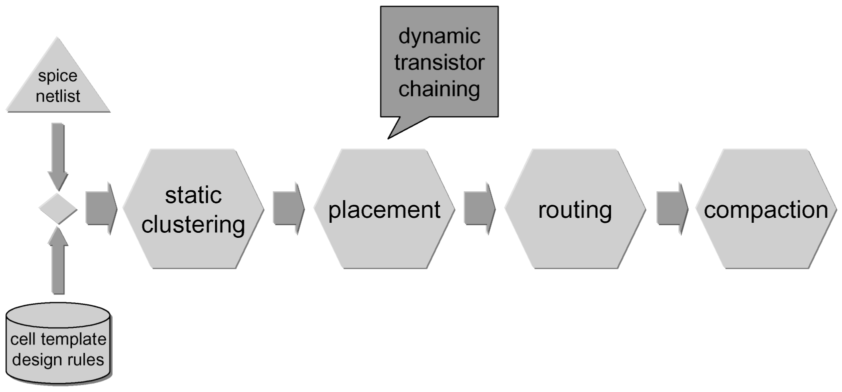


Fig. 3. A flowchart demonstrating the proposed methodology.

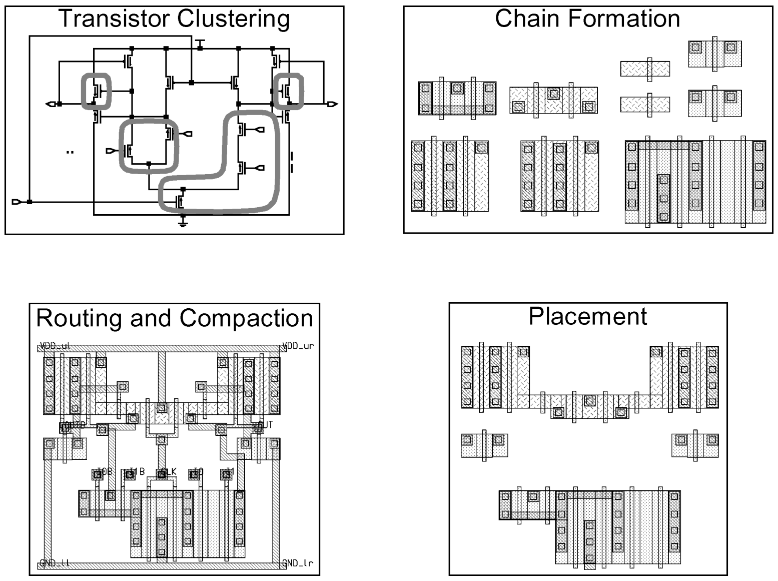


Fig. 4. Major stages of the proposed cell synthesis flow.

completed placement is routed (step 3) and then compacted (step 4) to remove unutilized space. An illustration of each of the major steps of this process is shown in Figure 4.

In this paper we present our algorithms for steps 1 and 2, which we combine with a third-party router and compactor into a complete end-to-end tool flow in order to conduct our experiments. Thus, our primary concern is to obtain from the initial netlist *a routable placement of minimum area*.

Central to our methodology are techniques for the modeling and optimization of transistor diffusion sharing. When electrical connections are made through geometry sharing, area is saved both because of the overlapping geometry and because of the elimination of wiring. In other reported 2D cell synthesis systems

[Xia et al. 1994; Fukui et al. 1995; Saika et al. 1997; Vahia and Ciesielski 1999; Oskar and Ciesielski 1999; Serdar Sechen 1999] this optimization is performed as a static chaining step before placement. *Koan* [Cohn et al. 1994], which is targeted at analog synthesis, performs no explicit chain formation, but instead allows arbitrary merged structures to form dynamically during placement.

The unique aspect of our methodology is that it adopts the strengths of both the 2D digital cell styles and the analog styles—we make use of a late binding process in which explicit digital-style transistor chains are optimized *dynamically* during placement, and we also allow arbitrary analog-style shared structures to form through dynamic geometry merging. During placement the chain optimization process thus has access to global placement and routing information which is not available in a static preprocessing step.

1.4 Outline

In the remainder of this paper we elaborate the details of each step in our proposed methodology. In Section 2 we review the relevant aspects of transistor chaining theory. In Section 3 we introduce the details of our placement algorithms. In Section 4 we discuss the implementation of our experimental system, which is named *TEMPO*, and present some experimental results. In Section 5 we summarize the discussion and provide conclusions.

2. TRANSISTOR CHAINING THEORY

It should be clear from the layout shown in Figure 2 that most geometry sharing in digital circuits, even the complex digital circuits with which we are concerned, takes place within transistor chains. However, in contrast to the functional cell style chains of Figure 1, note that the chains in noncomplementary circuits do not normally appear in dual pairs. In this section we review the theory behind nondual transistor chain optimization. We begin by establishing some basic terminology.

Definition 1 (*Transistor Cluster*). A transistor cluster is an unordered set of same-polarity transistors, all of which are reachable through a sequence of source or drain terminal connections. These are also often referred to as *channel-connected components (CCCs).*

Definition 2 (*Transistor Chaining*). A transistor chaining is a mapping which assigns a unique linear ordering to the source and drain terminals of the transistors in a cluster.

Definition 3 (*Transistor Chain*). A transistor chain is a transistor cluster which has been assigned a chaining. This is a well-defined physical structure for which a design-rule correct layout can be generated.

Definition 4 (*Diffusion Break*). A diffusion break is a position within a transistor chain at which two neighboring transistors do not share a common electrical terminal. Therefore these two terminals will not be able to share geometry and must be separated by the proper design rule distance.

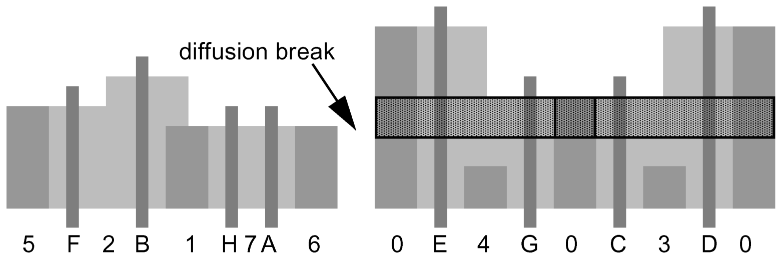


Fig. 5. An example demonstrating our transistor chaining terminology.

Definition 5 (*Transistor Subchain*). A transistor subchain is a subset of a transistor chain which is free of diffusion breaks. We will make use of this definition in our dynamic chaining approach discussed in Section 3.2.

An example which illustrates the above definitions is provided in Figure 5. Shown is the physical geometry for a single transistor *chain* with the transistor gates labeled with letters and the source/drain electrical nodes labeled with numbers. The example is one possible *chaining* of the transistor cluster $\{A, B, C, D, E, F, G, H\}$. This particular chaining can be described with the following ordered list of node numbers and gate names: $(5, F, 2, B, 1, H, 7, A, 6, 0, E, 4, G, 0, C, 3, D, 0)$. This chaining consists of two *subchains* $(5, F, 2, B, 1, H, 7, A, 6)$ and $(0, E, 4, G, 0, C, 3, D, 0)$, and one *diffusion break* (which can be identified in the ordered chaining list by the presence of two adjacent node numbers $6, 0$).

The process of nondual transistor chain optimization involves two steps. First, the transistors in the design must be partitioned into clusters. Second, a chaining must be found for each cluster. The first problem, transistor clustering, has received little attention in the literature and is usually performed heuristically. We will discuss our approach in Section 3.1. The second problem, chain optimization, is traditionally broken up into two components: chain *width* minimization and chain *height* minimization.

Chain *width* minimization is the process of finding a chaining which minimizes the number of diffusion breaks, and hence the width of the chain. *Height* minimization is normally the process of locating, from among the set of minimum width solutions, a chaining which requires the fewest number of routing tracks to complete the intra-chain source-drain routing.[1] However we would like to point out that, while a minimum "height" solution is locally minimal with respect to the intra-chain routing, it may not be globally minimal with respect to the complete inter-chain cell routing cost. We will explore this observation more completely in Section 3.2.

Transistor chain width optimization is traditionally posed as a graph problem. An undirected graph called the *diffusion graph* [Uehara and Van Cleemput 1981] is constructed for each cluster by associating a vertex with each electrical node. Edges represent transistors and span the corresponding source-drain terminal vertices. A *transistor chaining* is represented in this graph as a trail

[1]In the channel routing literature, the required number of routing tracks is often reffered to as the *maximum routing density.*

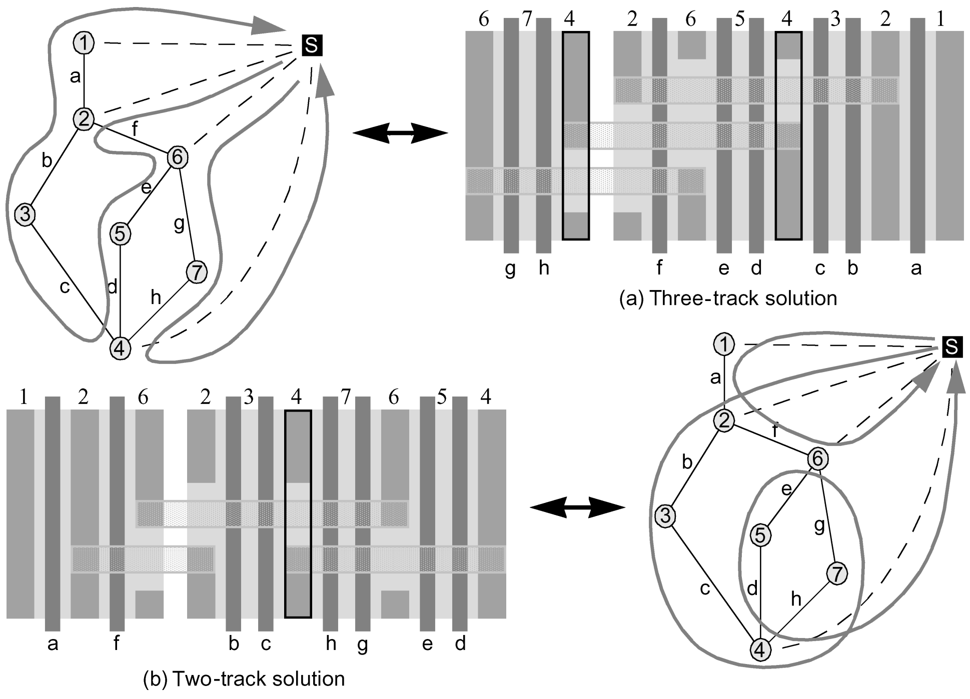


Fig. 6. Two different chainings with different maximum routing density.

which traverses every edge exactly once, with a *diffusion break* occurring between every pair of edges at which the trail is not connected. For example, in Figure 6 we show two different chainings for the same transistor cluster. Both graphs have one diffusion break, but note that chain a requires one fewer horizontal routing track than chain b and hence has a smaller chain "height."

An undirected graph which can be covered by a single continuous trail, touching every vertex at least once and every edge exactly once, is called an *Eulerian graph*, and such a trail is called an *Euler trail*. It is well known that a graph is Eulerian if and only if it has either zero or two odd-degree vertices. In order to find a minimum width chaining on a diffusion graph, we use a technique introduced by Basaran [1997]. An artificial vertex (the *super-vertex*) is added and edges (called *super-edges*) are drawn connecting it to all odd degree vertices, creating an Eulerian graph, which we will call the *augmented diffusion graph*. The algorithm in Figure 7 is used to find a covering Euler trail on this graph beginning at the vertex v_α and ending at the vertex v_β. The Euler trail must begin and end at the super-vertex, though any internal vertex may be used if the input graph was already Eulerian. The order in which the graph edges are visited corresponds to the ordering of the transistors in the chain, with the super-edges corresponding to diffusion breaks.[2] In Figure 6 we show Euler paths associated

[2]Note that there may be an exponential number of different solutions generated by algorithm EULER, but they will all be optimal in the sense that they all have the same minimal number of diffusion breaks

```
EULER($v_\alpha$, $v_\beta$)
    if $v_\alpha$ has no edges
        return $v_\alpha$
    starting from $v_\alpha$ create a random walk of $G$, never
    visiting the same edge twice, until $v_\beta$ is reached;
    let [$v_\alpha$, $v_1$], ..., [$v_n$, $v_\beta$] be this walk;
    delete [$v_\alpha$, $v_1$], ..., [$v_n$, $v_\beta$] from $G$;
    return (EULER($v_\alpha$, $v_\alpha$), EULER($v_1$, $v_1$), ..., EULER($v_n$, $v_n$))
```

Fig. 7. Algorithm to locate an Euler path from v_α to v_β in an Eulerian graph [Papadimitriou and Steiglitz, 1982].

with each chaining—the augmented diffusion graph super-vertex is labeled "S" and the super-edges are shown with dashed lines.

3. CELL LEVEL TRANSISTOR PLACEMENT

In order to attack the general two-dimensional cell synthesis problem, our methodology adopts a very general placement and routing approach. However, the need to optimize diffusion sharing distinguishes this problem from traditional macro-block placement in a number of ways. In the remainder of this section we discuss our approach to transistor placement. Section 3.1 presents our algorithm for creation of the static transistor clusters from which we will construct our transistor chains. Section 3.2 discusses the algorithms which we use for the dynamic exploration of the different chainings of each transistor cluster. Section 3.3 discusses a simple method by which we permit more complex forms of arbitrary geometry sharing to occur in the placement. Section 3.4 shows how these techniques can be combined with a simulated annealing algorithm to achieve the concurrent optimization of transistor chain ordering and placement. Section 3.5 discusses the methods which we use to model the routing cost function during placement.

3.1 Static Transistor Clustering

In the first step, transistor clustering, the goal is to determine the optimal number of transistor clusters and which transistors belong in each cluster. However, at this stage very little information is available, only the transistor connectivity and the sizes of each transistor, so the choices can only be heuristic in nature. On the schematic of the CGaAs mux flip-flop shown in Figure 2, we outline the clusters chosen by the designer in the corresponding manual layout.

Authors in previous works have used a wide variety of heuristics at this stage. In Fukui et al. [1995], the authors used simulated annealing. In Saika et al. [1997], only simple chains made up of maximum length series chains were created and larger chains are formed during the 1D optimization step. It is also fairly common, especially in one-dimensional tools, to approach clustering by performing logic gate recognition, as in Lefebvre and Skoll [1992].

We begin with an initial set of clusters formed from the set of all maximal channel-connected components (MCCCs).[3] We then make use of a Fidducia-Mattheyses (FM) bipartitioning algorithm [Fiduccia and Mattheyses 1982] to recursively partition these clusters until a user-supplied upper size bound is met. Because we are primarily interested in minimizing the net cutset, and not in obtaining balanced partitions, a loose balance criterion is used. An optional feature allows the user to specify an upper limit on channel width variation among the transistors in a particular cluster.[4] This feature is an alternative to transistor folding, and is enforced when the initial MCCCs are formed.

3.2 Dynamic Transistor Chaining

As we mentioned in Section 3.1, it is very difficult to determine an optimal static clustering of the transistors into diffusion-connected chains. This is because the clustering step has no information about the relative positions of the chains in the final placement. For the same reason, traditional chain optimization algorithms cannot be expected to locate globally optimal chaining solutions if performed as a preprocessing step prior to placement. Such an approach can find a chaining solution with minimum width and minimum internal routing cost (i.e., "height") but, as mentioned previously, such chainings may not be globally optimal when external wiring costs are taken into account. It is thus not clear *a priori* to which chain a particular transistor should be assigned, and which chain ordering should be chosen. The following technique, which we refer to as *dynamic chain optimization*, allows the placement step to dynamically alter the chainings and thus perform optimization with regard to the global area and wiring costs.

Dynamic chain optimization is enabled by the static clustering step, which is permitted to find relatively large transistor clusters. Using the algorithm described below, the placement step dynamically explores the universe of all possible minimum-width chaining solutions for each cluster. For each candidate chaining solution, a geometry generator is called to obtain the design-rule correct chain geometry and perform local intra-chain routing.[5] The transistor chains are then split at the diffusion breaks, and the resulting *subchains* are passed to the placement engine as the atomic placeable objects. These subchains are free to be placed in any two-dimensional arrangement that optimizes the area and the external routing.

It is easy to see that there will be many Euler trail covers for a given cluster's diffusion graph, and that all of them will have the same number of diffusion breaks. Thus the set of all trail covers corresponds to the set of all minimum-width chains, and the number of subchains being placed will remain constant.

[3]The MCCCs are trivially found in linear time as they are simply the connected components [Cormen et al. 1990] of the diffusion graph.

[4]In the experiments reported in Section 4.3, we use an FM balance criterion tolerance of $\pm 20\%$, we enforce a maximum cluster size limit of 20 transistors, and we limit the maximum variation among channel widths to a factor of 2.

[5]Intra-chain routing is a simple instance of one-dimensional routing without vertical constraints and can be performed optimally in linear time with a greedy left-edge algorithm. This routing is performed only if full contract strapping is not specified in the cell template.

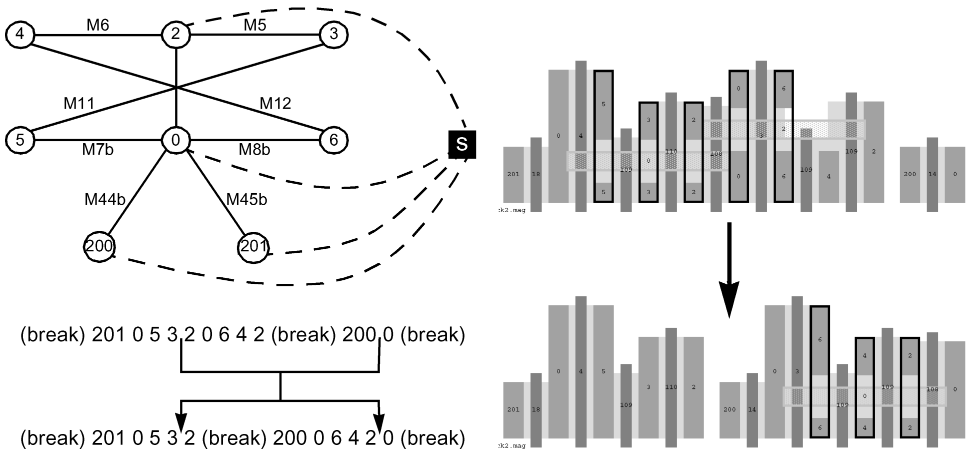

4
M6
2
M5
3
M11
M12
5
M7b
0
M8b
6
S
M44b
M45b
200
201
(break) 201 0 5 3 2 0 6 4 2 (break) 200 0 (break)
(break) 201 0 5 3 2 (break) 200 0 6 4 2 0 (break)

Fig. 8. A chain modification performed with Basaran's [1997] Iterative method.

However, when a new chaining is found for a cluster, individual transistors may move from one subchain to another, and the individual subchains may grow or shrink in size, providing a significant degree of added flexibility during placement.

Note again that we are not attempting to optimize the internal routing cost (chain "height") of the transistor chains, as is traditional in the literature. In fact, the physical height of the chain is fixed by the transistor channel widths in the input schematic. However, the chain geometry generator will report the number of internal chain nets which cannot be routed on top of the chains, and these nets are added to the global inter-chain routing cost (see Section 3.5). Thus the subchains are optimized such that their internal local routing cost is taken into account as well as the resulting global routing cost.

In our dynamic approach, it is the task of the placement engine to explore the universe of all possible minimum-width chainings for each cluster. We make use of the following iterative technique due to Basaran [1997] for generating one valid chaining from another. As shown in Figure 8, we select a random subtrail from the current trail cover and reset the edges of this subtrail in the diffusion graph. We then call algorithm `Euler()` to generate a different subtrail with the same two endpoints. Notice how the relative sizes of the two subchains in our example changed as a result of this transformation. By controlling the size of the selected subtrail, one can gain some indirect control over the degree of change to the chain (a feature which proved useful in implementing the automatic range limiting capability of our simulated annealing based placement engine). It can be shown that Basaran's method is complete, and can thus be used to construct every possible minimum-width Euler trail which can be embedded in the selected graph.

3.3 Support for Arbitrary Geometry Sharing

While most geometry sharing in complex cell layouts occurs in explicitly formed transistor chains, we observed that this is not always the case. An example of

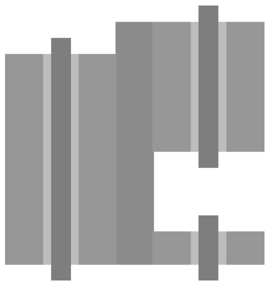


Fig. 9. An example of a second-order shared structure.

more complex sharing can be see in the "Z"-shaped structure in the upper-left corner of the layout in Figure 2. In addition to explicitly merged chains, we also allow geometry to merge during the placement step, thus permitting larger merged structures to form and taking advantage of less obvious patterns of connection. This is a trick often used by skilled human designers, and also is the primary method for geometry merging in the *Koan* [Cohn et al. 1994] analog placement tool. We call the resulting merged objects *second-order shared structures*. An example in Figure 9 shows two small transistors merging with a single larger transistor.

In order to support arbitrary geometry merging at the transistor level, we use a fairly simple mechanism. If two objects are in the proper configuration such that they have electrically compatible ports facing each other, the design rule constraint δ_1 can be relaxed to a smaller value, δ_2, to allow those ports to overlap. This may result in design rule violations in the final placement if the ports do not line up precisely, but this can be repaired in a postprocessing step.

3.4 Placement Modeling and Optimization

In our implementation, we have chosen to model the placement search space using the symbolic Sequence Pair representation developed by Murata et al. [1995]. This choice was made because we conjecture that symbolic methods provide a more efficient and rigorous framework with which to traverse the search space than direct representations of the object coordinates as used in *Timberwolf* [Sechen and Sangiovanni 1985] and *Koan* [Cohn et al. 1994]. In particular we chose the Sequence Pair over symbolic Slicing Trees because some valid placements are not representable in the latter.

In order to systematically explore the Sequence Pair solution space, as in Murata et al. [1995], we use simulated annealing [Rutenbar 1989]. Our solution space is defined by the set of all minimum-width chainings for each transistor cluster, the set of all rotations and mirrorings of each placeable object, and the set of all permutations of two sequences, Γ^- and Γ^+. This pair of sequences represent the relative x-axis and y-axis positions of the placeable objects, respectively. The following set of moves are used by the simulated annealing engine:

(1) Swap a pair of objects in either Γ^+, or Γ^- or both;
(2) translate one object to a different position in either Γ^+, or Γ^- or both;

(3) rotate and/or mirror one object into a different orientation; and
(4) reorder a selected chain as explained in Section 3.2.

Our simulated annealing engine makes use of a standard adaptive cooling schedule, automated initial temperature selection, range limiting, and statistical move selection [Cohn 1992]. The cost function is a weighted combination of the placement cost and an estimate for the routing cost:

$$\text{cost} = w_1 \cdot \text{placement} + w_2 \cdot \text{routing} \tag{1}$$

Several alternative placement cost metrics can be selected by the user: optimization of the cell area or perimeter are both possible with the ability to fix the height, width, or aspect ratio. Our routing cost metric is discussed in Section 3.5.

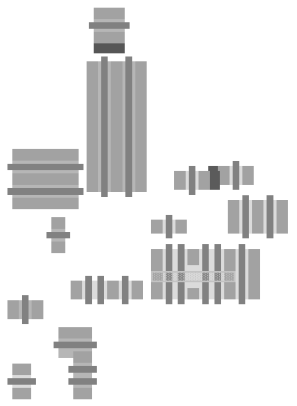
3.5 Routing Model

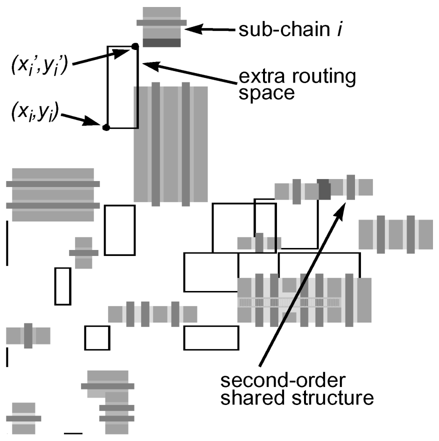
Because of the serialized split between placement and routing, during the placement step we are forced to make use of computationally efficient *estimates* of the actual routing cost. The routing model which we use during placement accounts for routing effects in two ways. The routing component of the placement cost function in Equation (1) is included to encourage the simulated annealing algorithm to find a placement, among all near-minimum area placements, which has low routing complexity. This cost is estimated with a manhattan net length estimate. Multiterminal signal nets are approximated as rectilinear minimum spanning trees (RMSTs), which provide a fairly tight lower bound on the actual routing cost and can be computed in linear time.[6]

In addition to the routing complexity component of the cost function, we must also account for the extra space that this routing will require. Without inserting extra space between the transistors prior to routing, we will most likely present the router with an infeasible problem, especially when few layers of interconnect are available. However, to model this cost precisely would require a complete global routing solution. As an alternative, we make use of the following approximate technique which is a variant of the technique used in Murata et al. [1995].

Figure 10(a) shows an intermediate placement without the addition of routing space. It is clear that some nets will be unroutable. Figure 10(b) shows the same placement with the addition of extra routing space. Here each object is translated from its original coordinate (x_i, y_i) to a new coordinate (x_i', y_i') based on an estimate of the total routing resources which will be required below it and to its left. The added boxes demonstrate the movement of each object—the lower-left corner of box i marks (x_i, y_i) and the upper-right corner marks (x_i', y_i'), for object i appearing to its upper-right.

[6]Both Prim's and Kruskal's algorithms are $O(V + E)$ where V is number of vertices and E is the number of edges.


(a) Before routing space insertion



(b) After routing space insertion

Fig. 10. A demonstration of routing space insertion.

This cell expansion is performed as follows. We first expand the cell's bounding box using the equation

$$x_{\max}' = x_{\max} + \beta T\left(\frac{\sum_{i\in N} H_i}{y_{\max}}\right), \quad y_{\max}' = y_{\max} + \beta T\left(\frac{\sum_{i\in N} W_i}{x_{\max}}\right) \tag{2}$$

where $(x_{\max}, y_{\max})$ and $(x_{\max}', y_{\max}')$ are the old and new coordinates of the upper-right bounding box corner, respectively, T is the routing pitch, N is the set of all nets, H_i and W_i are the height and width of net i's bounding box, respectively, and β is a user-defined scaling factor.[7] We then assign new coordinates to each block i based on the following equations:

$$x_i' = x_i + (x_{\max}' - x_{\max})\frac{x_i}{x_{\max}}, \quad y_i' = y_i + (y_{\max}' - y_{\max})\frac{y_i}{y_{\max}} \tag{3}$$

This method approximates the total number of horizontal and vertical routing tracks which will be required, assuming that each net requires one horizontal and one vertical track, and distributes these evenly throughout the design. We therefore call this the *uniform expansion method*.[8]

As a final observation, note that our method does not break apart second-order shared structures (one is marked with an arrow) which are present in the original unexpanded placement. This is explicitly supported in the expansion algorithm by identifying instances of geometry sharing and assigning all shared structures the same expansion factor as the most lower-left shared object.

[7]In our experiments β was determined experimentally for each circuit by observing the routability of several trial placements over a range of different β values.

[8]The method presented in Murata et al. [1995] moves each block based on the set of all routes which begin to its lower left. We have found that this places an artificial bias in the routing, concentrating the routing resources toward the lower left and creating congestion in the upper right. In contrast, our *uniform expansion* method looks at the total number of required horizontal and vertical routing tracks and obtains the lower-left routing estimate by distributing the required resources evenly throughout the cell.

4. EXPERIMENTAL EVALUATION

To validate our approach, we have conducted an extensive series of experiments using a new set of high-performance benchmark circuits. In the following we describe our prototype implementation and benchmark circuits and report the results of our experiments.

4.1 Implementation

A prototype tool called *TEMPO* (Transistor Enabled Micro Placement Optimization) has been designed as a framework to explore the ideas discussed in this paper. *TEMPO* is implemented in approximately 48,000 lines of C++ code and has been tested under the Sun Solaris and Linux operating systems. To complete our flow, we make use of the *Anagram-II* [Cohn et al. 1994] router for postplacement detailed routing and the *Masterport* leaf cell compactor from Cascade Design Automation Inc. We have adopted a datapath-style cell template for the current set of experiments: Internal routing is assumed to be two-layer and limited to poly and metal-1. Control inputs are assumed to arrive vertically in metal-2 and data/power/ground inputs and outputs arrive horizontally in metal-3. Prior to routing, we insert these terminals as overcell routing tracks in a position which is as close as possible to the center of the associated net's bounding box. Once the detailed routing step has placed contacts to these overcell tracks, the metal geometry is removed, leaving behind a set of gridded pins for connection by the chip-level router.

4.2 Benchmarks

The lack of a standard set of 2D cell synthesis benchmark circuits has made comparison with competing approaches difficult. In the case of the static CMOS logic family, it is a simple matter to generate artificial benchmark circuits—Maziasz and Hayes [1992], in fact, showed that there are exactly 3505 dual series-parallel circuits of "practical size" (i.e., with a maximum series chain length of 4). However, no standard set of circuit benchmarks exist for the broader category of circuits with which we are concerned. For the following set of experiments, we make use of a new set of 22 benchmark circuits which we have assembled from the solid-state circuits literature. These circuits, described in Table I, are representative of the class of aggressive, high-performance, and low-power library cells for which this system is targeted. Included are circuits designed in domino and zipper CMOS, static and dynamic CVSL, as well as single- and double-ended PTL. As can be seen, these circuits represent a broad cross section of design styles and applications; however, some general observations can be made. Most of the circuits are derived from datapath applications and are either arithmetic circuits, advanced latches and flip-flops, or latches/flip-flops with embedded logic.

Data which can be used to characterize the complexity of these circuits is graphed in Figure 11. The circuits range in size from an 18-transistor sense-amp flip-flop to an 84-transistor dynamic eight-way multiplexor taken from an experimental 1-GHz PowerPC condition code unit. We also show the average net fanout for each circuit as an indication of cell routing complexity

Table I. Benchmark Circuit Descriptions

#	Name[a]	Description
1	aoi-lff	True Single Phase Clocked Logic (TSPCL) and-or-invert logic flip-flop
2	blair-ff	High-speed differential double-edge-triggered CMOS flip-flop
3	dcsl3-42comp	Differential Current Switch Logic 4-2 compressor (DCSL-3 style load)
4	dcvsl-xor4	Dynamic Cascode Voltage Switch Logic 4-way XOR gate
5	dec-csa-mux	High-speed carry-save adder with muxed latching inputs
6	diff-fa	Differential full adder
7	dpl-fa	Double-Pass Transistor Logic full adder
8	dptl-42comp	Dynamic Pass Transistor Logic 4-2 compressor sum logic
9	emodl-cla	Enhanced Multiple Output Domino Logic carry-lookahead adder stage
10	ghz-ccu-merge	4-Way dynamic merge circuit for 1-GHz PowerPC condition code unit
11	ghz-ccu-prop	4-Way dynamic propagate circuit for 1-GHz PowerPC condition code unit
12	ghz-mux8-la	8-Input latching mux with hold and scan for 1-GHz PowerPC
13	mux2-sdff	Semi-Dynamic dual-rail logic flip-flop with embedded 2-input mux
14	muxff	Dynamic Complementary GaAs mux flip-flop
15	ptl-42comp	Pass Transistor Logic 4-2 compressor
16	ptl-rba	Pass Transistor Logic redundant binary adder
17	qnp-fa	Quasi NP-domino pipelined full adder
18	sa-ff	Differential edge-triggered sense-amplifier flip-flop
19	sa-mux-ff	Differential edge-triggered sense-amp flip-flop with embedded 4:1 mux
20	t17-fa	Seventeen-transistor low-power full-adder
21	xor-ff	Sense-amplifier flip-flop with dynamic differential embedded XOR gate
22	zip-fa2	Dual-bit adder in CMOS zipper logic

[a]Sources: # 1—Rogenmoser and Huang [1996]; # 2—Blair [1997]; 3—Somasekhar and Roy [1996]; # 4—Heller et al. [1984]; # 5—Kowaleski et al. [1996]; # 6—Shimazu et al. [1989]; # 7—Suzuki et al. [1993]; # 8—Hanawa et al. [1996]; # 9—Wang et al. [1997]; # 10—Burns and Nowka [1998]; # 11—Burns and Nowka [1998]; # 12—Silberman et al. [1998]; # 13—Klass [1998]; # 14—Stetson [1997]; # 15—Yano et al. [1996]; # 16—Makino et al. [1996]; # 17—Lu and Samueli [1993]; # 18—Montanero et al. [1996]; # 19—Stephany et al. [1998]; # 20—Shams and Boyoumi [1998]; # 21—Matsui et al. [1994]; # 22—Friedman and Liu [1984].

(fanout data is shown for all nets, and with the power and ground nets excluded).

In most cases, the benchmark circuits were specified in the literature without transistor sizing information. Due to this fact, and our desire to implement the circuits in a common technology, we have tuned the transistors in each circuit ourselves. We made use of a simple heuristic nonlinear gradient descent algorithm, described in Riepe [1999], using electrical models from a commercial 0.5-μm CMOS process.

In addition to problems in transistor-level layout synthesis, we expect these benchmarks to prove useful on other problems in high-performance cell-level integrated circuit design automation. Other possible applications include transistor-level circuit synthesis, transistor tuning, cell testing and characterization, and performance-driven technology mapping. The characteristics of these circuits are described more completely in Riepe [1999], and the sized netlists can be obtained by contacting the authors.

4.3 Results

We report the results of a series of experiments which compare *TEMPO* to the *LAS* synthesis tool [Chow et al. 1999] from Cadence Design Systems Inc. *TEMPO* was run on a 300-MHz Intel Pentium-II-based workstation while *LAS* was run on a 170-MHz Sun Ultrasparc system; both had 128 megabytes of RAM.

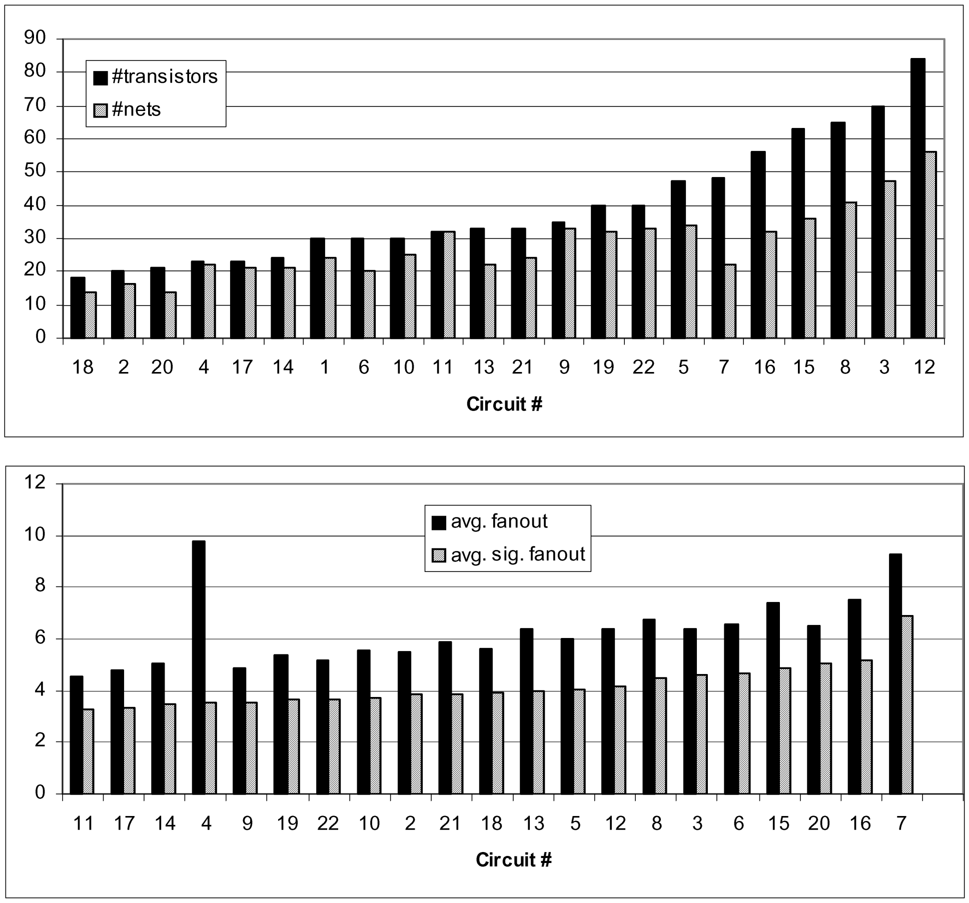


Fig. 11. Benchmark circuit characteristics.

We assume a standard 0.5-mm CMOS process and make use of the MOSIS scalable submicron design rules (rev. 7.2) with a lambda value of 0.3 μm. *LAS* was run in full optimization mode to explore all possible transistor foldings and cell aspect ratios. It is a 1-1/2-dimensional row-based system which assumes a standard-cell style template—to minimize the overhead of the metal-1 power rails, we required these to have minimum width. *TEMPO* cells were optimized for minimum area under our datapath style cell template. *LAS*, which uses deterministic algorithms, was run once per circuit. *TEMPO*, because of its stochastic nature, produces results with a statistical spread in quality. The results we present for each circuit represent the best solution from a series of 100 trials. We discuss runtime, and the statistical distribution of our results, later in this section.

The improvement in benchmark cell area obtained with *TEMPO* is displayed in Figure 12. The quality of the results is encouraging: with two notable exceptions, the *TEMPO* layouts are consistently smaller than the *LAS* layouts. In 15 of the 22 circuits, this improvement exceeded 10%, and in 8 of the 22,

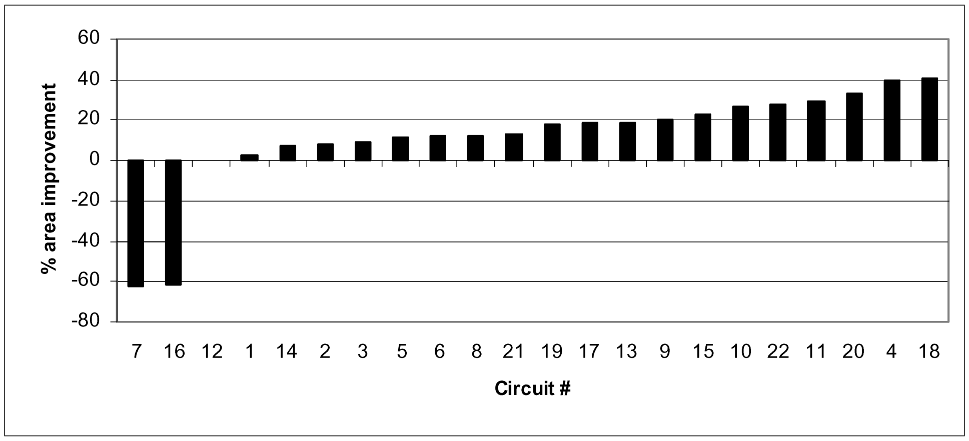


Fig. 12. The cell area improvement seen when comparing *TEMPO* to *LAS*.

the improvement exceeded 20%. Three representative layouts are shown in Figure 13. The upper cell is the dcvsl-xor4 benchmark which provides a visually appealing example of a 2D transistor arrangement with fairly sparse routing. This is in contrast to the row-based transistor arrangement and channel routing used by *LAS*. An area improvement of 39.69% was obtained for this circuit. Our second example, the mux2-sdff circuit, provides another demonstration of dense placement and routing with an area reduction of 18.93%. Our third example, the ptl-rba circuit, is one of the two cases for which *LAS* produced a superior layout. In this case, *LAS* was able to obtain an extremely efficient linear transistor arrangement with little wasted space, while *TEMPO* could not achieve a competitive placement. In this case, the *LAS* area is 61.45% smaller than *TEMPO*'s.

It is interesting to observe the effectiveness of *TEMPO*'s dynamic geometry sharing optimization. In Figure 14 we show three different placements obtained for the muxff benchmark circuit. In Figure 14(a) we show an optimized placement obtained with diffusion sharing disabled—it is obviously extremely noncompact. In Figure 14(b) we show the pattern of sharing which is obtained with our arbitrary geometry sharing mechanism enabled. No explicit transistor chaining is performed in this mode; nevertheless *TEMPO* is able to discover several small chains of merged transistors as well as several complex second-order shared structures. However, the layout is irregular and chain lengths are short, rarely exceeding three or four transistors. This example clearly demonstrates that digital cells benefit tremendously from an approach based on explicit transistor chain optimization, as seen in the final example in Figure 14(c).

Because *TEMPO* makes use of a stochastic optimization algorithm, it is likely that it will achieve a different result every time that it is run. We were interested in observing the statistical spread in the final layout area obtained in this way. Figure 15 shows a histogram of the area spread observed for the sa-ff benchmark. Figure 16 shows the number of standard deviations below the mean at which the minimum area layout was located. This number was consistently

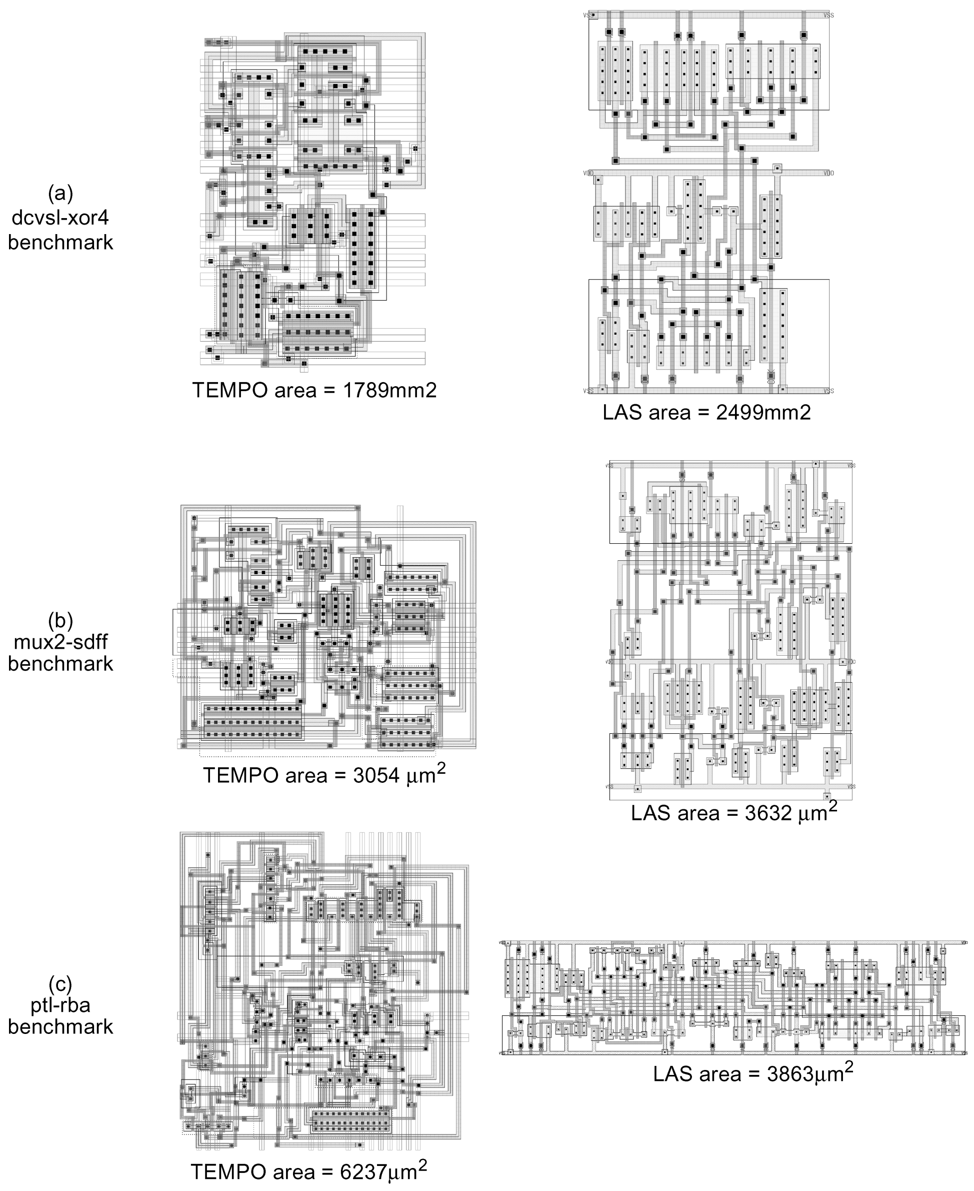


Fig. 13. Three representative cell layouts. Results of *TEMPO* are displayed on the left while results of *LAS* are displayed on the right (pairs are shown approximately to scale).

between 1 and 2. Assuming a Gaussian distribution, 95% of the solutions will fall within 2 standard deviations of the mean, indicating that at least 20 trials will be necessary to achieve good results.

In Figure 17, we show some data indicating the per-run execution time of *TEMPO*. Clearly, *TEMPO*'s simulated-annealing-based 2D placement algorithm, and the *Anagram-II* router's aggressive rip-up and reroute strategy, come

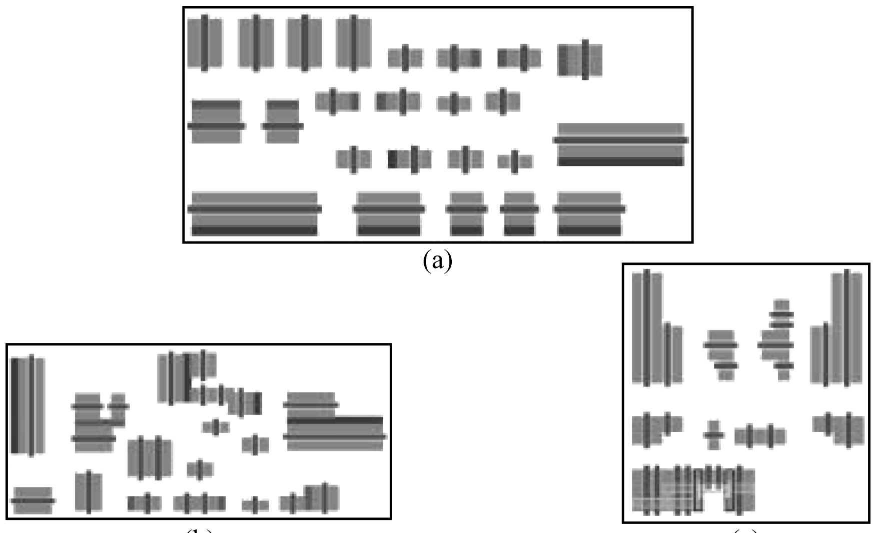


Fig. 14. The muxff benchmark (a) without chaining or merging, (b) with chaining disabled, and (c) with both chaining and merging enabled.

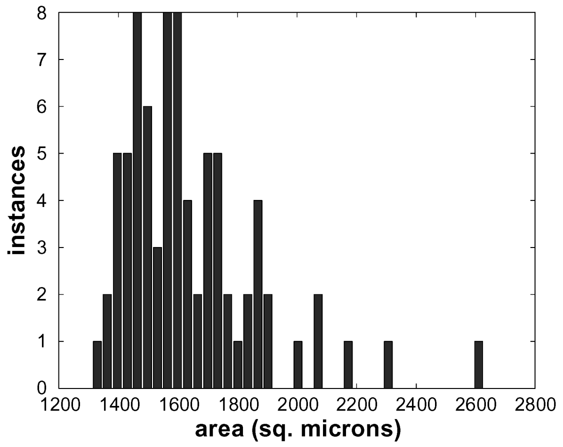


Fig. 15. Layout area distribution observed over 100 experimental trials for one representative benchmark circuit.

at a substantial cost. This cost must also be multiplied by the number of trials conducted in the effort to obtain the best layout (100 in our experiments). Improving the convergence behavior and the predictability of the placement results to eliminate this high cost will be a significant topic of future effort.

5. CONCLUSIONS AND FUTURE WORK

In this work, we have discussed the problem of cell synthesis for high-performance, nondual, digital VLSI cells. Our proposed solution is based on an unconstrained two-dimensional transistor placement and routing framework,

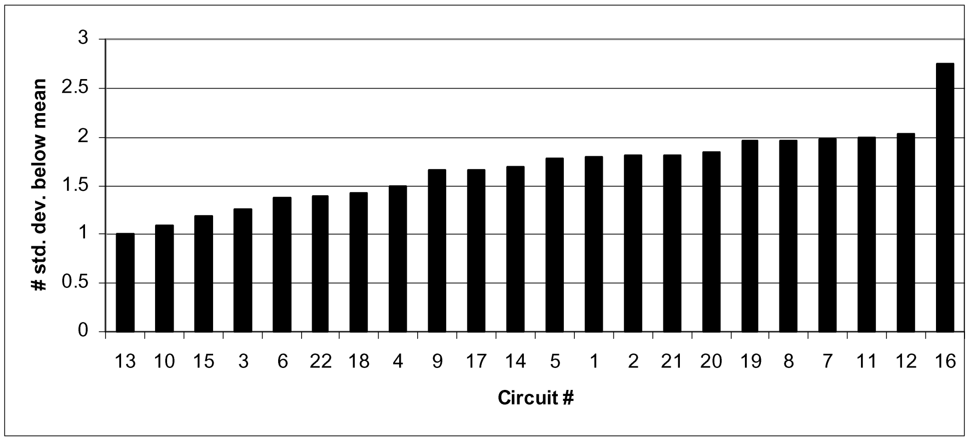
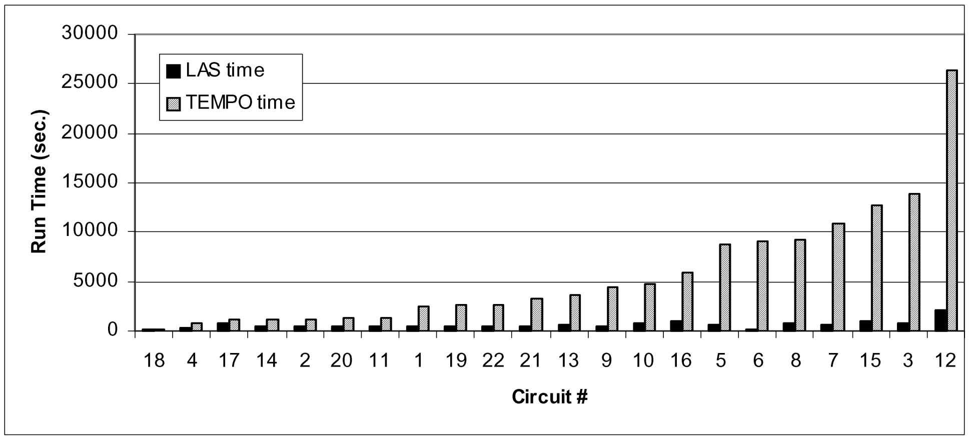


Fig. 16. Number of standard deviations below the mean at which the minimum area solution for each benchmark circuit was found.

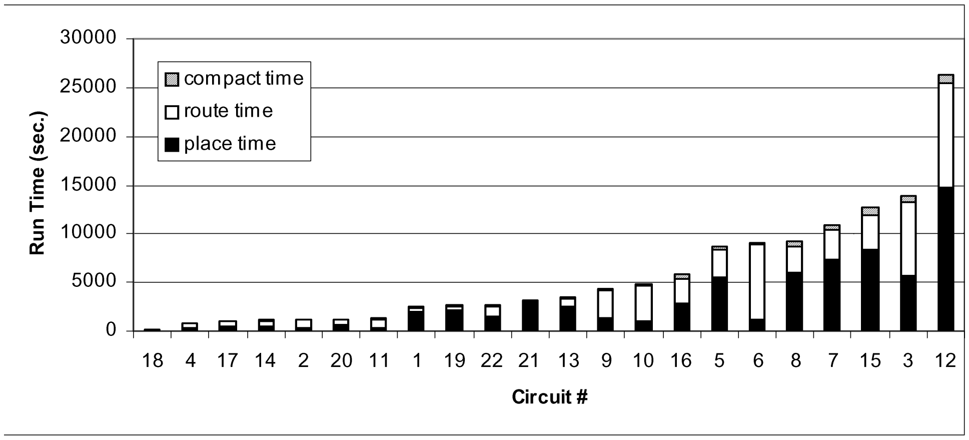
with an emphasis on methods for the modeling and optimization of geometry sharing among the transistors. Our specific contributions include the following:

(1) A static transistor clustering technique, consisting of an FM-style bipartitioning algorithm with a loose balance criterion and an optional upper bound on transistor channel width variation.
(2) A simulated annealing placement optimization engine with a new degree of freedom: dynamic transistor chain optimization. This late-binding technique allows chain optimization to proceed with detailed knowledge of the global placement and routing cost function.
(3) A unique placement model in which the atomic placeable objects consist of transistor *subchains*. As new chainings are found for each cluster during placement, transistors may move from one subchain in the cluster to another, and the relative sizes of the subchains may change, but the number of subchains (and thus placeable objects) remains constant.
(4) Support for the sharing of geometry between arbitrary neighboring objects in the placement.
(5) A routing component in the placement cost function which includes transistor subchain "spill" routing, as well as global inter-chain routing cost estimates, permitting global routing costs to be optimized.
(6) A new variant of an algorithm for the insertion of routing space in the placement cost function.
(7) A new set of 22 benchmark circuits representing leafcells in a wide variety of noncomplementary digital logic families.

We have developed a prototype implementation of a transistor-level placement tool based on these ideas, and assembled a complete end-to-end synthesis flow which makes use of third-party tools for detailed routing and postrouting compaction. An initial set of experiments, conducted on a new set of benchmark circuits, demonstrated encouraging results. However, the wide statistical



(a) Per-run execution time comparison



(b) *TEMPO* runtime components

Fig. 17. *TEMPO* and *LAS* run time statistics for all 22 benchmark circuits. Times shown are for a single run of each tool. To account for statistical spread, *TEMPO* was run 100 times on each circuit and the best result reported.

spread in the quality of results produced by our stochastic placement optimization approach resulted in a high computational cost which will need to be remedied in the future.

Our experience has demonstrated a number of promising areas for improvement and further work. Currently we make no effort to maintain symmetry between transistor chains, since we are not targeting series-parallel CMOS logic families. However, series-parallel subcircuits are often embedded within larger circuits, so it would be beneficial to include some form of dual-chain optimization capability as well. In addition, in order to simplify poly gate routing, it will often be advantageous to maintain partial symmetry between chains, dual or not, which share many common gate inputs. Some logic families, most notably

CVSL, contain sensitive analog circuitry which requires symmetric placement and routing capability, and techniques for this could be borrowed from the analog placement and routing literature. We also see the need for parasitic-aware performance-driven optimization techniques, transistor folding support, optimized well and well contact insertion, and more accurate techniques for routing cost estimation and congestion management.

ACKNOWLEDGMENTS

The authors would like to thank Professor Rob Rutenbar and Dr. Phiroze Parakh for many interesting discussions, and the anonymous reviewers for their helpful suggestions.

REFERENCES

Askar, S. and Ciesielski, M. 1999. Analytical approach to custom datapath design. In *Proceedings of the 1999 International Conference on Computer-Aided Design*. 98–101.

Basaran, B. 1997. Optimal diffusion sharing in digital and analog CMOS layout. Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA; also, CMU Report No. CMUCAD-97-21, Carnegie Mellon University, Pittsburgh, PA.

Bernhardt, B. et al. 1995. Complementary GaAs (CGaAs^{TM}): A high performance BiCMOS alternative. In *Proceedings of the 1995 GaAs IC Symposium*. 18–21.

Blair, G. 1997. Comments on "New single-clock CMOS latches and flip-flops with improved speed and power savings." *IEEE J. Solid State Circ. 32*, 10 (Oct.), 1611.

Burns, J. and Feldman, J. 1988. C5M—a control logic layout synthesis system for high-performance microprocessors. *IEEE Trans. CAD 17*, 1 (Jan.), 14–23.

Burns, J. and Nowka, K. 1998. Parallel condition-code generation for high-frequency powerPC microprocessors. In *Proceedings of the 1998 Symposiums on VLSI Circuits*. 115.

Chow, S., Chang, H., Lam, J., and Liao, Y. 1992. The layout synthesizer: An automatic block generation system. In *Proceedings of the 1992 CICC*. 11.1.1–11.1.4.

Cohn, J. 1992. Automatic device placement for analog cells in KOAN. Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA; also, CMU Report No. CMUCAD-92-07, Carnegie Mellon University, Pittsburgh, PA.

Cohn, J., Garrod, D., Rutenbar, R., and Carley, L. R. 1994. *Analog Device-Level Layout Automation*. Kluwer Academic Publishers, Boston, MA.

Cormen, T., Leiserson, C., and Rivest, R. 1990. *Introduction to Algorithms*. McGraw-Hill Book Company, New York, NY. 88.

Fiduccia, C. and Mattheyses, R. 1982. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference.* 241–247.

Friedman, V. and Liu, S. 1984. Dynamic logic CMOS circuits. *IEEE J. Solid State Circ. 19*, 2 (Apr.), 265.

Fukui, M., Shinomiya, N., and Akino, T. 1995. A new layout synthesis for leaf cell design. In *Proceedings of the 1995 ASP Design Automation Conference*. 259–263.

Gupta, A. and Hayes, J. P. 1997. CLIP: An optimizing layout generator for two-dimensional CMOS cells. In *Proceedings of the 34th Design Automation Conference*. 452–455.

Hanawa, M., Kaneko, K., Kawashimo, T., and Maruyama, H. 1996. A 4.3ns 0.3mm CMOS 54x54b multiplier using precharged pass-transistor logic. In *Proceedings of the 1996 International Solid-State Circuits Conference*. 265.

Heller, L., Griffin, W., Davis J., and Thomas, N. 1984. Cascode voltage switch logic: A differential CMOS logic family. In *Proceedings of the 1984 International Solid-State Circuits Conference*. 17.

Hsieh, Y., Huang, C., Lin, Y., and Hsu, Y. 1991. LiB: A CMOS cell compiler. *IEEE Trans. Comput.-Aided Des. 10*, 8 (Aug.), 994–1005.

Klass, F. 1998. "Semi-dynamic and dynamic flip-flops with embedded logic. In *Proceedings of the 1998 Symposium VLSI Ciruits*. 109.

Kowaleski, J., Wolrich, G., Fischer, T., Dupcak, R., Kroesen, P., Pham, T., and Olesin, A. 1996. A dual-execution pipelined floating-point CMOS processor. In *Proceedings of the 1996 International Solid-State Circuits Conference*. 359.

Lefebvre, M. and Skoll, D. 1992. PicassoII: A CMOS leaf cell synthesis system. In *Proceedings of the 1992 MCNC International Workshop on Layout Synthesis*, vol. 2. 207–219.

Lefebvre, M., Marple, D., and Sechen, C. 1997. The future of custom cell generation in physical synthesis. In *Proceedings of the 1997 Design Automation Conference*. 446–451.

Lu, F. and Samueli, H. 1993. A 200-MHz CMOS pipelined multiplier-accumulator using a quasi-domino dynamic full-adder cell design. *IEEE J. Solid State Circ. 28*, 2 (Feb.), 125.

Makino, H., Suzuki, H., Morinaka, H., Nakase, Y., Mashiko, K., and Sumi, T. 1996. A 286 MHz 64-b floating point multiplier with enhanced CG operation. *IEEE J. Solid State Circ. 31*, 4 (April), 510.

Matsui, M., Hara, H., Uetani, Y., Kim, L.-S., Nagamatsu, T., Watanabi, Y., Chiba, A., Matsuda, K., and Sakurai, T. 1994. A 200 MHz 13 mm^2 2-D DCT macrocell using sense-amplifying pipeline flip-flop scheme. *IEEE J. Solid State Circ. 29*, 12 (Dec.), 1484.

Maziasz, R. L. and Hayes, J. P. 1992. *Layout Minimization of CMOS Cells*. Kluwer Academic Publishers, Boston, MA.

Montanero, J., Witek, R., Anne, K., Black, A., Cooper, E., Dobberpuhl, D., Donahue, P., Eno, J., Farell, A., Hoeppner, G., Kruckemyer, D., Lee, T., Lin, P., Madden, L., Murray, D., Pearce, M., Santhanam, S., Snyder, K., Stephany, R., and Thierauf, S. 1996. A 160 MHz 32b 0.5 W CMOS RISC microprocessor. In *Proceedings of the 1996 International Solid-State Circuits Conference*. 215.

Murata, H., Fujiyoshi, K., Nakatake, S., and Kajitani, Y. 1995. Rectangle packing based module placement. In *Proceedings of the 1995 International Conference on Computer-Aided Design*. 472–479.

Papadimitriou, C. and Steiglitz, K. 1982. *Combinatorial optimization algorithms and complexity*. Prentice-Hall, Englewood Cliffs, NJ. 413.

Poirier, C. 1989. Excellerator: Custom CMOS leaf cell layout generator. *IEEE Trans. Comput.-Aided Des. 8* (July), 744–755.

Riepe, M. and Sakallah, K. 1999. Transistor-level micro-placement and routing for two-dimensional digital VLSI cell synthesis. In *Proceedings of the 1999 International Symposium on Physical Design*. 74–81.

Riepe, M. 1999. Transistor-level micro-placement and routing for two-dimensional digital VLSI cell synthesis. Ph.D dissertation, The University of Michigan, Ann Arbor, MI.

Rogenmoser, R. and Huang, Q. 1996. An 800-MHz 1-μm CMOS pipelined 8-b adder using true single-phase clocked logic-flip-flops. *IEEE J. Solid State Circ. 31*, 3 (Mar.), 405.

Rutenbar, R. 1989. Simulated annealing algorithms: An overview. IEEE Circ. Dev. 5, 1 (Jan.) 19–26.

Saika, S., Fukui, M., Shinomiya, N., and Akino, T. 1997. A two-dimensional transistor placement algorithm for cell synthesis and its application to standard cells. *IEICE Trans. Fund., E80–A*, 10 (Oct.), 1883–1891.

Sechen, C. and Sangiovanni-Vincentelli, A. 1985. The TimberWolf placement and routing package. *IEEE J. Solid State Circ. SC-20*, 2 (Apr.), 510–522.

Serdar, T. and Sechen, C. 1999. AKORD: Transistor level and mixed transistor/gate level placement tool for digital datapaths. In *Proceedings of the 1999 International Conference on Computer-Aided Design*. 91–97.

Shams, A. and Bayoumi, M. 1998. A new full adder cell for low-power applications. In *Proceedings of the 1998 Great Lakes Symposium on VLSI*. 47.

Shimazu, Y., Kengaku, T., Fujiyama, T., Teraoka, E., Ohno, T., Tokuda, T., Tomisawa, O., and Tsujimichi, S. 1989. A 50MHz 24b floating-point DSP. In *Proceedings of the 1996 International Solid-State Circuits Conference*. 45.

Silberman, J., Aoki, N., Boerstler, D., Burns, J., Dhong, S., Essbaum, A., Ghosal, U., Heidel, D., Hofstee, P., Lee, K., Meltzer, D., Ngo, H., Nowka, K., Posluszny, S., Takahashi, O., Vo, I., and Zoric, B. 1998. A 1.0GHz single-issue 64b PowerPC integer processor. In *Proceedings of the 1998 International Solid-State Circuits Conference*. p. 231.

Somasekhar, D. and Roy, K. 1996. Differential current switch logic: A low power DCVS logic family. *IEEE J. Solid State Circ. 31*, 7 (July), 987.

Stephany, R., Anne, K., Bell, J., Cheney, G., Eno, J., Hoeppner, G., Joe, G., Kaye, R., Lear, J., Litch, T., Meyer, J., Montanaro, J., Patton, K., Pham, T., Reis, R., Silla, M., Slaton, J., Snyder, K., and Witek, R. 1998. A 200 MHz 32b 0.5 W CMOS RISC microprocessor. In *Proceedings of the 1998 International Solid-State Circuits Conference*. 239.

Stetson, S. 1997. University of Michigan, Ann Arbor, MI. Private communication.

Suzuki, M., Ohkubo, N., Yamanaka, T., Shimizu, A., and Sasaki, K. 1993. A 1.5ns 32b CMOS ALU in double pass-transistor logic. In *Proceedings of the 1993 International Solid-State Circuits Conference*. 91.

Tani, K. et al. 1991. Two-dimensional layout synthesis for large-scale CMOS circuits. In *Proceedings of the 1991 International Conference on Computer-Aided Design.* 490–493.

Uehara, T. and VanCleemput, W. M. 1981. Optimal layout of CMOS functional arrays. *IEEE Trans. Comput. C-30, 5* (May), 305–312.

Vahia, D. and Ciesielski, M. 1999. Transistor level placement for full custom datapath cell design. In *Proceedings of the 1999 International Symposium on Physical Design*. 158–163.

Wang, Z., Jullien, G., Miller, W., Wang, J., and Bizzan, S. 1997. Fast Adders Using Enhanced Multiple-Out-put Domino Logic. *IEEE J. Solid State Circ. 32*, 2 (Feb.), 209.

Xia, H., Lefebvre, M., and Vinke, D. 1994. Optimization-based placement algorithm for BiCMOS leaf cell generation. *IEEE J. Solid State Circ. 29*, 10 (Oct.), 1227–1237.

Yano, K., Sasaki, Y., Rikino, K., and Seki, K. 1996. Top-down pass-transistor logic design. *IEEE J. Solid State Circ. 31*, 6 (June), 797.

Received December 2000; accepted March 2002