Optimal 2-D Cell Layout with Integrated Transistor Folding†

Avaneendra Gupta
Cadence Design Systems, Inc.
555 River Oaks Parkway, #1A1
San Jose, CA 95134
1-408-895-2129
avigupta@cadence.com

John P. Hayes
Dept. of EECS, University of Michigan
1301 Beal Avenue
Ann Arbor, MI 48109
1-734-763-0386
jhayes@eecs.umich.edu

1. ABSTRACT

Folding, a key requirement in high-performance cell layout, implies breaking a large transistor into smaller, equal-sized transistors (legs) that are connected in parallel and placed contiguously with diffusion sharing. We present a novel technique *FCLIP* that integrates folding into the generation of optimal layouts of CMOS cells in the two-dimensional (2-D) style. *FCLIP* is based on integer linear programming (ILP) and precisely formulates cell width minimization as a 0-1 optimization problem. Folding is incorporated into the 0-1 ILP model by variables that represent the degrees of freedom that folding introduces into cell layout. *FCLIP* yields optimal results for three reasons: (1) it implicitly explores all possible transistor placements; (2) it considers all diffusion sharing possibilities among folded transistors; and (3) when paired P and N transistors have unequal numbers of legs, it considers all their relative positions.

***FCLIP* is shown to be practical for relatively large circuits with up to 30 transistors. We then extend *FCLIP* to accommodate and-stack clustering, a requirement in most practical designs due to its benefits on circuit performance. This reduces run times dramatically, making *FCLIP* viable for much larger circuits. It also demonstrates the versatility of *FCLIP*'s ILP-based approach in easily accommodating additional design constraints.**

2. INTRODUCTION

Cell layout synthesis falls in the category of constrained optimization whose goal is to find a solution that optimizes some cost function under a set of constraints. The cost function can be the cell area, its delay, or a combination of these. The constraints include bounds on width or height, aspect ratio, number of diffusion rows, or the maximum size of transistors. Since cell layout optimization is NP-hard [3], any exact algorithm can, in the worst case, have an exponential run time. Therefore, most prior techniques for cell synthesis have avoided optimal algorithms in favor of faster, but less exact heuristic methods.

Maziasz and Hayes [13] have shown that for one-dimensional (1-D) cell layout, exact algorithms can be both computationally feasible and generate significantly better solutions than heuristic methods. Recently, the authors successfully employed an exact optimization method, integer linear programming (ILP), in the *CLIP* technique to generate 2-D layouts of minimum width and height [7]. The layout problem is formulated as a 0-1 ILP problem, which is then solved using an off-the-shelf integer solver. *CLIP* provides optimum layouts for practical-sized cells. However, it assumes equal transistor sizes. Often, transistors must be individually sized to meet a circuit's performance goals. This leads to unequal transistor sizes that can waste cell area. To meet the performance goals and minimize area, large transistors must be folded or split into smaller transistors of uniform size. In this paper, we present the first exact technique that integrates transistor folding into 2-D cell synthesis. This technique, called *FCLIP* (*Folding in Cell Layout via Integer Programming*), extends *CLIP* to generate optimal 2-D cell layouts with folding.

FCLIP minimizes cell area in the following stages: First, transistors are folded based on user-specified limits on the maximum size of the P and N transistors. The input circuit is preprocessed to generate P/N pairs and identify and-stacks, that is, transistors that are connected in series. And-stack clustering is not only necessary in practical designs, but also reduces the complexity of the problem and, in turn, *FCLIP*'s run times. Then an ILP model is formulated and solved to determine a 2-D layout of minimum width W_{min}; this model maximizes diffusion sharing among folded transistors and minimizes vertical inter-row connections. A second ILP model is then constructed to generate a layout that has width W_{min} and minimum height, measured by the number of horizontal routing tracks. This paper only discusses 2-D cell width minimization with folding; however, *FCLIP* can be extended to minimize cell height also.

FCLIP yields optimal results with folding for two reasons: (1) It implicitly explores all diffusion sharing possibilities among folded transistors; and (2) when paired P/N transistors have unequal numbers of legs, it considers all their relative positions. Not only does *FCLIP* support 2-D layout, it is superior to prior folding techniques proposed for 1-D layout [9, 11, 14] which consider folding only after a transistor placement has been determined, and, as we show later, can produce suboptimal layouts. *FCLIP*'s optimal method is particularly targeted towards the layout of standard-cells and datapath bit-cells for high-volume, high-performance microprocessor designs. These cells, which typically have 20-40 transistors, are used multiple times and

† This research was sponsored by a grant from Intel Corporation.

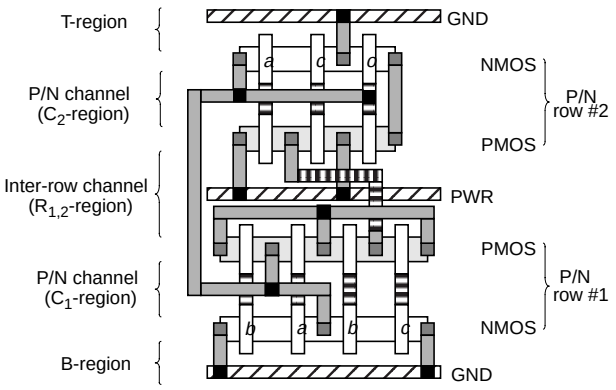


Fig. 1: The two-dimensional (2-D) cell layout style

1. The circuit to be laid out is a static dual CMOS circuit of fixed structure (no transistor reordering is allowed).
2. Alternate P/N rows are flipped to allow the power rails to be shared among adjacent diffusion rows.
3. P and N transistors of a pair are vertically aligned so that their terminals on common nets can be connected using vertical wires.
4. Intra-cell routing is restricted to polysilicon and Metal1, each of which can be used in the vertical or horizontal direction.
5. When a net spans multiple P/N rows, connections are made to join terminals that are on transistors placed closest together.
6. Terminals that are on transistors placed in adjacent diffusion rows are connected using routes in the channel between the rows. Hence, such connections do not affect the width of the cell.
7. Diffusion areas do not allow Metal1 wires to be routed over them. All routes that span different rows are routed along the sides of the cell, and contribute to the widths of the rows they pass through.

Table 1: Assumptions underlying 2-D width minimization

must have their layouts optimized as much as possible.

We begin by describing transistor folding and its advantages in Section 3. Section 4 reviews relevant aspects of the *CLIP* method [7] for minimizing 2-D cell width. Section 5 extends *CLIP* to *FCLIP* by integrating folding into ILP model formulation. Finally, Section 6 demonstrates the versatility of *FCLIP*'s ILP-based approach by incorporating and-stack clustering, a requirement in most practical designs.

3. TRANSISTOR FOLDING

The assumed 2-D cell layout style is illustrated in Fig. 1, and generalizes the well studied 1-D style [4, 6, 9, 14]. 2-D cells contain multiple rows of P and N diffusions called *P/N rows*. The P and N transistors of a P/N row are grouped into *P/N pairs* using standard techniques [6]. In dual CMOS circuits, the P and N transistors of each P/N pair share a common gate net. The basic assumptions of our 2-D style are summarized in Table 1 [7]; they are typical of those used in the layout of standard-cells and datapath bit-cells in current microprocessor designs.

In practical cell designs, the size of each transistor is determined individually to meet the circuit's performance goals such as rise and fall delays [15]. The *size* of a transistor, illustrated in the cross-sectional view of Fig. 2*a*, is defined as the width of its P or N channel. Since transistors are placed horizontally, their sizes affect both cell width and height. Hence, non-uniformity in transistor size often leads to wasted cell area. *Transistor folding* is the process of splitting a transistor into two or more smaller, equal-sized transistors called *legs*. The legs of a folded

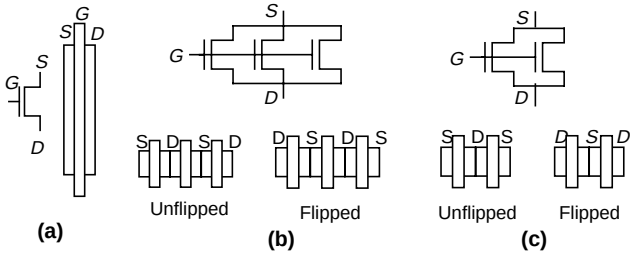


Fig. 2: (a) A transistor and its folding into (b) an odd (three) and (c) an even (two) number of legs

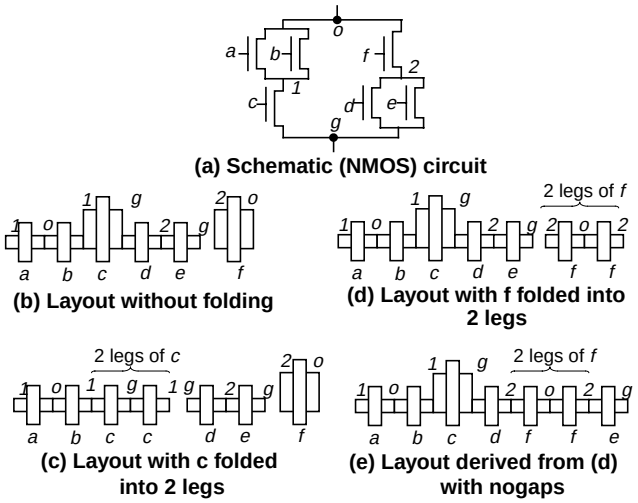


Fig. 3: Effects of folding on diffusion abutment

transistor are connected in parallel and are typically placed contiguously with diffusion sharing in the cell layout.

Figure 2 illustrates folding into an odd or even number of legs. The nets at the ends of the folded transistor depend on the parity of the number of legs. When a transistor with source and drain nets (*S*, *D*) has an odd number of legs, the nets at its ends remain (*S*, *D*) irrespective of the its orientation (Fig. 2*b*). However, with an even number of legs (Fig. 2*c*), the end nets depend on the transistor's orientation: (*S*, *S*) if the transistor is placed unflipped, and (*D*, *D*) if flipped. The placement affects diffusion sharing with adjacent transistors and hence, both orientations of a folded transistor should be considered for possible abutment.

Figure 3 demonstrates the impact of folding on diffusion sharing and the cell width. A minimum-width 1-D placement without folding for the NMOS circuit of Fig. 3*a* is shown in Fig. 3*b*; it corresponds to the chain cover [6] {*abcde*, *f*}. If transistor *c* is folded into two legs, a diffusion gap is introduced in the chain *abcde* between *c* and *d* (Fig. 3*c*). However, if a different cover {*abfde*, *c*} is chosen, folding *c* does not introduce any new gaps. On the other hand, folding *f* into two legs transforms the placement of Fig. 3*b* to that of Fig. 3*d*, which yields a placement without gaps (Fig. 3*e*). Hence, folding can often be exploited to increase diffusion sharing and eliminate diffusion gaps.

Finally, if the transistors of a P/N pair have different numbers of legs, the diffusion sharing possibilities are further increased. Consider the P/N pair in Fig. 4 with its transistors folded into 3 and 5 legs, respectively. Since the P transistor has fewer legs, it can be placed in three possible

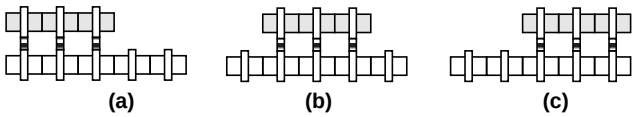


Fig. 4: Placement of pairs with unequal legs and the P transistors (a) left-justified, (b) centered, (c) right-justified.

ways relative to the N transistor: (*a)* left-justified, (*b*) centered, and (*c*) right-justified. Each placement affects the nets at the ends of the P/N pair. Moreover, since no net appears on one side of the P diffusion, there is no net to be matched (shared) with the pair placed on that side. The centered position is the most flexible since it relaxes diffusion sharing on both sides of the P diffusion. Thus, when the transistors of a P/N pair have unequal numbers of legs, their placements can increase the possibility of diffusion sharing with other pairs.

The above effects of folding on diffusion sharing and the cell width are further magnified when we consider pair-wise diffusion sharing and allow an arbitrary number of legs for each transistor. In addition, folding affects the cell height since it reduces the height of each leg of the folded transistor and affects the routing within the cell. For 1-D layout, folding significantly reduces cell area and control of the cell's aspect ratio [6]. In summary, folding affects both the cell width and height for the following reasons: (a) It decreases the transistor's height, which affects the cell height; (b) it increases the number of transistors, which affects the cell width and intra-cell routing; (c) the orientation of a even-legged transistor affects the transistor's diffusion sharing with other transistors; and (d) when the P/N transistors of a pair have unequal numbers of legs, their relative placement—centered, left-justified, or right-justified—also affects diffusion sharing.

Thus, folding should be considered during the process of transistor placement, and not later, as is usually the case [9, 11, 14]. Four different folding problems can be defined:

1. *Static placement and folding*: Given a pre-specified 2-D transistor placement and limits on transistor size (*folding limits*), fold transistors in place and determine their orientation to preserve the placement and minimize area.

2. *Static placement with dynamic folding*: Given a 2-D transistor placement, determine the number of legs and orientation for each transistor so that cell area is minimized

3. *Dynamic placement with static folding*: Given folding limits, fold the transistors and determine their 2-D placement, and their orientation to minimize the cell area.

4. *Dynamic placement and folding*: For each transistor, determine the number of its legs, their 2-D placement, and their orientation, so that the overall area is minimized.

Problem 1 is the simplest since its only goal is to find an orientation for each transistor that maximizes diffusion sharing and minimizes cell height. Figure 5 illustrates how flipping transistors and changing the relative positions of the transistors of P/N pairs can affect diffusion sharing. The layout of Fig. 5*b* is obtained by folding the transistors of Fig. 5*a*. Both layouts have one diffusion gap. However, by flipping P_1, making N_2 right-justified with respect to P_2, and flipping N_2, we obtain a layout with no gaps (Fig. 5*c*).

Problem 2 also pre-specifies a transistor placement, but determines the number of legs for each transistor during the

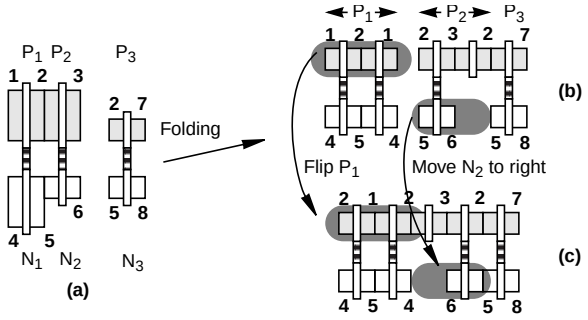


Fig. 5: Illustration of folding with a pre-specified placement: Layout (a) before folding, (b) after folding with transistors in place; (c) after re-orienting and re-positioning transistors

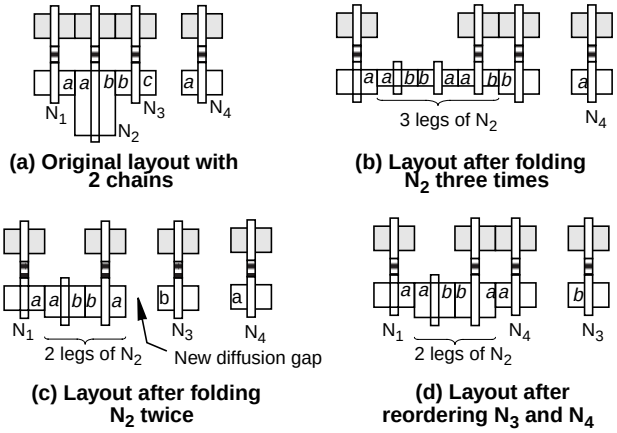


Fig. 6: Effect of even and odd folding on diffusion sharing

solution phase. The dynamic-programming technique of Her and Wong [8] solves this problem exactly for 1-D layouts and heuristically for 2-D layouts. Other heuristic techniques for this problem are included in the *GENAC* [14], *LiB* [10], and *THEDA.P* [11] 1-D cell synthesis tools. However, since problem 2 fixes the transistor ordering before folding, it can restrict the number of folds—to an odd number of legs to prevent chain splitting—and produce suboptimal abutments. Figure 6*a* shows a layout with two chains placed with one diffusion gap. When N_2 has an odd number of legs, no new gap is introduced (Fig. 6*b*). However, even folding of N_2 (Fig. 6*c*) requires a new gap between N_2 and N_3. In this case, if transistor ordering is not fixed, the positions of N_3 and N_4 can be swapped, allowing N_2 to share diffusion with N_3. This yields the layout of Fig. 6*d*, which still requires only one gap.

The third problem—dynamic placement with static folding—is a generalization of the first. It fixes the number of legs for each transistor, and then determines their position and orientation to minimize cell area. For 1-D layout, this problem has been addressed by Gupta, et al. in the *XPRESS* [6] tool and by Malavasi and Pandini [12].

Finally, problem 4 is the most general in that it allows both placement and folding to be dynamic. Hence, any technique for this problem must simultaneously select the amount of folding for each transistor and determine its best placement and orientation.

In this paper, we address the third problem—we assume static folding and aim at dynamically determining a 2-D

Parameters	Interpretation
1. *numPairs, numRows, maxSlots*	The number of pairs, P/N diffusion rows, and slots respectively
2. *pairs, rows, slots, nets*	The set of pairs, rows, slots, and nets
3. *PpairNets, NpairNets*	*PpairNets*[*p*] = {gate, source, and drain nets of the P transistor of pair *p*} (*NpairNets* is similarly defined)
4. *nDifffAtBottomOfCell*	A decision variable that is 1 (0) if the N diffusion is placed at the bottom (top)
5. *Psrc*[*pairs, nets*], *Pgate*[*pairs, nets*], *Pdrn*[*pairs, nets*]	*Psrc*[*p, n*] = 1 if pair *p* has net *n* on the source diffusion of its P transistor (*Pgate*[*p, n*] and *Pdrn*[*p, n*] are similarly defined for gate / drain terminals)
6. *Nsrc*[*pairs, nets*], *Ngate*[*pairs, nets*], *Ndrn*[*pairs, nets*]	*Nsrc*[*p, n*] = 1 if pair *p* has net *n* on the source diffusion of its N transistor (*Ngate*[*p, n*] and *Ndrn*[*p, n*] are similarly defined for gate / drain terminals)
7. *share*[*pairs, orients, pairs, orients*]	*share*[p_i, o_i, p_j, o_j] = 1 if pair p_i in orient o_i can share diffusion with pair p_j in orient o_j

Table 2: Input (1–4) and derived (5–7) parameters for *CLIP*

placement that minimizes cell width. This is most widely applicable since practical designs typically specify folding limits for P and N transistors. Moreover, as discussed earlier, determining the 2-D placement after folding has several area advantages over static placement.

4. WIDTH MINIMIZATION

If W_r is the width of the *r*-th P/N row, the **2-D cell-width minimization problem** can be stated as follows [7]: Minimize the cell width W_{cell} by placing the P/N pairs in a given number of rows such that the maximum width among all rows is minimized. That is, minimize W_{cell}

$$W_{cell} = \max \{W_r: \text{for each P/N row } r = 1, 2, ...\} \quad (1)$$

As discussed in [7], W_{cell} for a 2-D layout depends on several factors: diffusion sharing, inter-row connections that run vertically between P/N rows and so add to their width, and the diffusion type (P or N) placed at the bottom of the cell. The width W_r of row *r* can be expressed as follows, where t_r, c_r, and v_r are the numbers of pairs, chains, and inter-row wires, respectively, in row *r*:

$$W_r = t_r + c_r - 1 + v_r \quad (2)$$

Hence, the following layout characteristics need to be determined: the row, position, and orientation of each pair, the inter-pair diffusion sharing; and the diffusion type, P or N, to be placed at the cell bottom.

We now review the *CLIP* technique of [7] for minimizing 2-D cell width. *CLIP* uses integer linear programming (ILP) as its core optimization engine. The layout parameters are modeled as 0-1 variables; the width minimization objective is converted to a linear cost function; and linear constraints over the 0-1 variables are used to ensure a valid 2-D layout.

Inputs and outputs. Table 2 lists the input and derived parameters for *CLIP*. To represent the position of each pair in the 2-D plane, we introduce place-holders called *slots* in each row in which pairs will be placed. Slots are numbered in increasing order from the left. Figure 7 illustrates row and slot numbering. We also define the following sets of integers: *slots* = {1,2,...*maxSlots*}, *rows* = {1, 2, ..., *numRows*}, and *orients* = {1, 2, 3, 4} representing the four possible orientations for each P/N pair. The boolean array *share* represents diffusion sharing between two P/N pairs. The element *share*[p_i, o_i, p_j, o_j] is set to 1 if pairs p_i and p_j can be placed adjacent to each other in orientations, o_i and o_j, respectively, with diffusion abutment.

The basic decision variables for each pair are represented by 0-1 arrays *X* and *Xor*, where *X*[*p*, s, *r*] = 1 implies pair *p* is placed in slot *s* of row *r*, and *Xor*[*p*, *o*] = 1 implies *p* is placed in orientation *o*. To model diffusion sharing, we introduce array *nogap* where *nogap*[*s*, *r*] = 1 if adjacent slots *s* and *s* + 1 have no gap between them. Then the width of each P/N row W_r can be expressed as follows.

$$\begin{aligned} W_r &= \#\text{pairs in row } r + \#\text{gaps in row } r + \#\text{vertical wires} \quad (3)\\ &= \#\text{pairs in row } r + (\#\text{pairs in row } r - 1 - \#\text{abutments}) + v_r \\ &= 2 \times \Sigma\, X[p, r] - \Sigma\, nogap[s, r] + v_r - 1 \end{aligned}$$

Constraints. We now describe the constraints of *CLIP* that enforce a valid 2-D layout.

1. *Pair inclusion*: Each pair must be placed in exactly one slot with one orientation.

$$\sum_{s \in slots} \sum_{r \in rows} X[p, s, r] = 1 \qquad \forall\, p \in pairs \quad (4)$$

$$\sum_{o \in orients} Xo[p, o] = 1 \qquad \forall\, p \in pairs \quad (5)$$

2. *Slot occupancy*: We force the first slot in each P/N row to be filled with exactly one pair, and slots to be filled in a left-justified order, that is, in each row *r*, the slot *s* should be occupied before the slot *s* + 1.

$$\sum_{p \in pairs} X[p, 1, r]=1 \qquad \forall\, r \in rows \quad (6)$$

$$\sum_{p \in pairs} X[p, s-1, r] \;\geq\; \sum_{p \in pairs} X[p, s, r] \qquad \forall\, r \in rows,\ s \in slots \quad (7)$$

3. *Diffusion sharing*: The variable *nogap*[*s*, *r*] can be defined by the following logic equation:

nogap[*s*, *r*] (8)
= **for** every pair p_i, p_j of pairs that can share diffusion
  for each orientation o_i of p_i, o_j of p_j, *share*[p_i, o_i, p_j, o_j] = 1
    or (p_i is placed in slot *s* in row *r*
      and p_i is placed in orientation o_i
      and p_j is placed in slot *s* + 1 in row *r*
      and p_j is placed in orientation o_j)

= **or** {*X*[p_i, *s*, *r*] **and or** {*X*[p_j, *s*+1, *r*] **and** *merged*[p_i, p_j]:
  $\forall\, p_j \in pairs$}: $\forall\, p_i \in pairs$}

Here *merged*[p_i, p_j] is 1 if pair p_i can share diffusion with pair p_j placed to its immediate right.

merged[p_i, p_j] (9)
= **or** {*Xor*[p_i, o_i] **and** *Xor*[p_j, o_j]:
  $\forall\, o_i, o_j \in orients$ such that *share*[p_i, o_i, p_j, o_j]}

= *Xor*[p_i, 1] **and or** {*Xor*[p_j, o_j]:
  $\forall\, o_j \in orients$ such that *share*[p_i, 1, p_j, o_j]}
 or *Xor*[p_i, 2] **and or** {*Xor*[p_j, o_j]:
  $\forall\, o_j \in orients$ such that *share*[p_i, 2, p_j, o_j]}
 or *Xor*[p_i, 3] **and or** {*Xor*[p_j, o_j]:
  $\forall\, o_j \in orients$ such that *share*[p_i, 3, p_j, o_j]}
 or *Xor*[p_i, 4] **and or** {*Xor*[p_j, o_j]:
  $\forall\, o_j \in orients$ such that *share*[p_i, 4, p_j, o_j]}

Finally, we ensure that for any given P/N pair, there can be at most one other pair placed on its immediate left or right side with diffusion abutment.

$$\sum_{p_j \in pairs} merged[p_i, p_j] \leq 1 \qquad \forall\, p_i \in pairs \quad (10)$$

$$\sum_{p_i \in pairs} merged[p_i, p_j] \leq 1 \qquad \forall\, p_j \in pairs \quad (11)$$

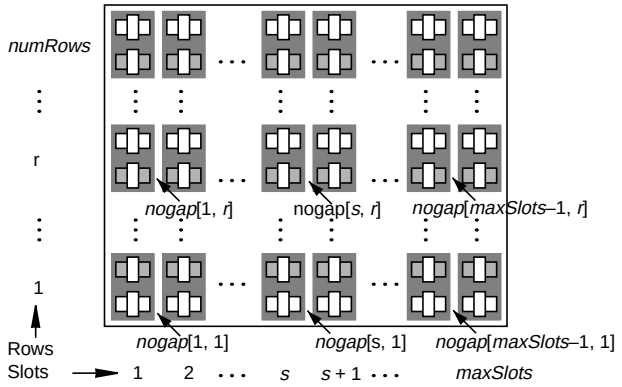


Fig. 7: Rows, slots, and diffusion gaps in a 2-D cell layout

4. *Inter-row connectivity*: These constraints determine the nets that must be routed from one P/N row to another. We introduce the 0-1 variables *V*, where *V*[*r, n*] = 1 if net *n* is routed vertically along row *r*. To represent *V*, we introduce four simpler, auxiliary 0-1 variables, *A*[*r, n*], Y[*r, n*], *T*[*r, n*], and *B*[*r, n*], which take the value 1 if net *n* appears above row *r*, below row *r*, in the top diffusion of row *r*, and in the bottom diffusion of row *r*, respectively. By enumerating all sixteen assignments of 0-1 values to the four variables *A, Y, T*, and *B*, the following reduced sum-of-products expression is obtained for *V*:

$$V = (\text{A } \textbf{and } Y \textbf{ and not } T) \textbf{ or } (A \textbf{ and } B \textbf{ and not } T) \textbf{ or } (Y \textbf{ and not } B \textbf{ and } T) \quad (12)$$

Constraints (8, 9, 12) are nonlinear since they involve logical **and** and **or** operators, so they must be converted to linear inequalities to be included in an ILP model. As shown in [7], these constraints can be linearized without introducing any new variables.

5. WIDTH MINIMIZATION WITH FOLDING

We now describe how the above width minimization model can be extended to incorporate transistor folding in *FCLIP*.

Inputs and outputs. Given folding limits on maximum transistor size, we first compute the number of legs for each transistor. Let *legs* be an integer array, where *legs*[*p*] contains the larger of the number of legs of the P and N transistor of P/N pair *p*.

As discussed in Section 3, transistor folding introduces a new degree of freedom to placement: the relative positions—centered, or left/right justified—of the transistors of each P/N pair. Hence, we change the array *share* of *CLIP* to a 6-dimensional 0-1 array, where *share*[p_i, o_i, x_i, p_j, o_j, x_j] is 1 if pairs p_i and p_j can share diffusions in orientations o_i and o_j, and in relative positions x_i and x_j, respectively. Here x_i and x_j take values 1, 2, or 3 for centered, left-justified, and right-justified, respectively. The value of *share* is again determined from the circuit's netlist. Fig. 8 illustrates the various diffusion sharing possibilities between two pairs P_1 and P_2, where P transistor of P_1 has 3 legs while its N transistor has 5 legs. Observe that in left-justified and centered positions, pair P_1 has no P-diffusion net on its right side. Therefore, since the P-diffusion net to the left of P_2 can be arbitrary, the P transistors of both pairs can be placed either unflipped or flipped.

To model the three different relative positions of each P/N pair, we change the *Xor* variables of the *CLIP model* to *Xor*[*pairs, orients, positions*], where *Xor*[*p, o, x*] = 1 when pair *p* is placed in orientation *o* and in relative position *x*. If *p*'s transistors have the same leg count, we set Σ *Xor*[*p, o*, 2] = Σ *Xor*[*p, o*, 3] = 0 (over all $o \in$ *orients*) for the left (x = 2) and right-justified positions (x = 3). If the difference in the number of legs is 1, then we set Σ *Xor*[*p, o*, 1] = 0 (over all $o \in$ *orients*) for the centered position x = 1.

The width W_r of each row *r*, defined by Eq. (3), is modified in *FCLIP* to consider the leg count *legs*[*p*] of each pair *p*.

$$\begin{aligned} W_r &= \#\text{pairs in row } r \\ &\quad + (\#\text{pairs in row } r - 1 - \#\text{abutments}) + v_r \\ &= 2 \times \Sigma\, legs[p] \times Xrow[p, r] - \Sigma\, nogap[s, r] + v_r - 1 \end{aligned} \quad (13)$$

Constraints. First, we introduce a new constraint to ensure that each P/N pair *p* is placed in exactly one orientation and relative position.

$$\sum_{o \in orients} \; \sum_{x \in \{1, 2, 3\}} Xor[p, o, x] = 1 \qquad \forall\, p \in pairs \quad (14)$$

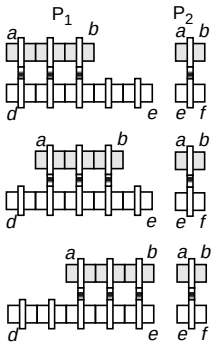
The pair inclusion, slot occupancy, and inter-row connectivity constraints of *CLIP* remain the same in *FCLIP*. The diffusion sharing constraint (8) is modified to handle the relative positions of P/N pairs. The variable *nogap*[*s, r*], which must be 1 if there is no gap between slots *s* and *s* + 1 in row *r*, is now defined as

$$\begin{aligned} &nogap[s, r] \\ &= \textbf{for every pair } p_i \text{ and } p_j \text{ of P/N pairs} \\ &\quad \textbf{for every relative position } x_i \text{ of } p_i \text{ and } x_j \text{ of } p_j \\ &\quad\quad \textbf{for each orientation } o_i \text{ of } p_i \text{ and } o_j \text{ of } p_j \\ &\quad\quad\quad \textbf{or } (p_i \text{ is placed in slot } s \text{ in row } r \text{ in orientation } o_i \\ &\quad\quad\quad\quad \textbf{and } p_j \text{ is placed in slot } s + 1 \text{ in row } r \text{ in orientation } o_j \\ &\quad\quad\quad\quad \textbf{and } p_i \text{ is placed in position } x_i \\ &\quad\quad\quad\quad \textbf{and } p_j \text{ is placed in position } x_j) \\ &= \textbf{or } (X[p_i, s, r] \textbf{ and } (\textbf{or } (X[p_j, s{+}1, r] \textbf{ and } merged[p_i, p_j]: \\ &\qquad\qquad \forall\, p_j \in pairs)): \forall\, p_i \in pairs) \end{aligned} \quad (15)$$

Here, as in *CLIP*, *merged*[p_i, p_j] = 1 if pairs p_i and p_j are in orientations that allow p_i to share diffusion with pair p_j to its immediate right. Hence,

$$\begin{aligned} &merged[p_i, p_j] \\ &= \textbf{or } (Xor[p_i, o_i, x_i] \textbf{ and } Xor[p_j, o_j, x_j] \\ &\qquad : \forall\, o_i, o_j, x_i, x_j \text{ such that } share[p_i, o_i, x_i, p_j, o_j, x_j]) \\ &= Xor[p_i, 1, 1] \\ &\quad \textbf{and } (\textbf{or } (Xor[p_j, o_j, x_j]: \forall\, o_j, x_j, share[p_i, 1, 1, p_j, o_j, x_j])) \\ &\textbf{or } Xor[p_i, 1, 2] \\ &\quad \textbf{and } (\textbf{or } (Xor[p_j, o_j, x_j]: \forall\, o_j, x_j, share[p_i, 1, 2, p_j, o_j, x_j])) \\ &\textbf{or } Xor[p_i, 1, 3] \\ &\quad \textbf{and } (\textbf{or } (Xor[p_j, o_j, x_j]: \forall\, o_j, x_j, share[p_i, 1, 3, p_j, o_j, x_j])) \\ &\textbf{or } Xor[p_i, 2, 1] \\ &\quad \textbf{and } (\textbf{or } (Xor[p_j, o_j, x_j]: \forall\, o_j, x_j, share[p_i, 2, 1, p_j, o_j, x_j])) \\ &\ldots \\ &\textbf{or } Xor[p_i, 4, 3] \\ &\quad \textbf{and } (\textbf{or } (Xor[p_j, o_j, x_j]: \forall\, o_j, x_j, share[p_i, 4, 3, p_j, o_j, x_j])) \end{aligned} \quad (16)$$

Equation (16) is a sum-of-products expression with product terms of the form *Xor*[p_i, o_i, x_i] **and** (*Xor*[p_j, 1, 1] **or** *Xor*[p_j, 1, 2] **or** *Xor*[p_j, 1, 3] **or** *Xor*[p_j, 2, 1] **or** ...) for every pair p_i and p_j of P/N pairs, and for every orientation o_i of p_i. Now, *merged*[p_i, p_j] appears as a positive term in expression (15) for *nogap*[*s, r*] and hence, as a negative term in the minimization cost function (13). It is therefore sufficient to have constraints that set *merged*[p_i, p_j] to 0 when p_i and p_j cannot share their adjacent diffusions; *merged*[p_i, p_j] will automatically be set to 1, if possible, when the model is



P1 orient	P2 orient
1	1
1	3
3	1
3	3
1	1
1	3
3	1
3	3
3	3

Fig. 8: Relative positions of transistors of a P/N pair P_1 and their orientations (1, 2, 3, or 4) that allow for diffusion sharing with another pair P_2 placed to its immediate right

solved. Since, for any pair p_j, at most one of $Xor[p_j, o_j, x_j]$ can be set to 1 in a given solution, each product term of (16) is equivalent to the linear inequality below:

$$2 \times merged[p_i, p_j] \leq Xor[p_i, o_i, x_i] + \sum_{o_j \in orients} \sum_{x_j \in \{1,2,3\}} share[p_i, o_i, x_i, p_j, o_j, x_j] \times Xor[p_j, o_j, x_j] + 2 \times \sum_{o_k \in orients,\, o_k \neq o_i} \sum_{x_k \in \{1,2,3\},\, x_k \neq x_i} Xor[p_i, o_k, x_k] \quad (17)$$

In words, for every orientation o_i and relative position x_i of p_i, if $Xor[p_i, o_i, x_i] = 1$, then $merged[p_i, p_j] = 1$ if and only if pair p_j is placed in some orientation and relative position in which it can share diffusions with p_i. However, if $Xor[p_i, o_i, x_i] = 0$, then $merged[p_i, p_j]$ is made independent of o_i and x_i by the last term in the above inequality.

Experimental results. We now present the results of experiments that apply *FCLIP* to five representative CMOS circuits taken from various sources. All our ILP models were specified in *AMPL* (*A Mathematical Programming Language*) [5], a high-level language that allows the models to be described in parameterized form, that is, independently of the input data used for a specific instance. We evaluated several general-purpose ILP solver programs but found the specialized 0-1 solver *OPBDP* [1] to be best suited to our application. All *FCLIP* run times presented here have been obtained with *OPBDP.*

Table 3 gives the minimum-width 2-D layouts obtained with *FCLIP* for the test circuits. The layouts in one, two, and three P/N rows were generated by enforcing different folding limits on the P and N transistors. *OPBDP*'s -h103 heuristic for selection of the branching variable was used for layout in a single row; for two or more rows, *OPBDP*'s h101 heuristic proved to have the shortest run times.

For circuits as large as the full adder with 28 transistors, *FCLIP* run times are in seconds in most cases. Compared to *CLIP*, the only additional variables that folding introduces in *FCLIP* are due to the relative positions—centered, left-, or right-justified—of the P and N transistors of pairs. These new variables depend on the difference in the numbers of legs of the P and N transistor of a pair. If this difference is either zero or more than one, then only the centered position must be considered—no additional variables are required. However, for a difference in leg count of one, both the left and right-justified positions must be considered, which requires the variables $Xor[p, o, 2]$ and $Xor[p, o, 3]$, and increases the total number of variables for pair p by four

Circuit	# trans	# nets	# rows	Wcell P/N folding limits				*OPBDP* time (secs)[a] P/N folding limits			
				None	10/5	5/5	5/3	None	10/5	5/5	5/3
D-latch [4]	18	12	1	10	12	21	21	0.4	10	1	5
			2	5	6	10	11	1	2	543	5
			3	4	4	8	8	1	1	2	3
				None	10/10	10/5	5/5	None	10/10	10/5	5/5
Series-parallel cct.[b]	20	20	1	11	17	20	26	0.1	95	0.3	1
			2	8	11	12	15	1	14	12	9
			3	5	7	8	10	7	41	68	96
				None	10/5	10/4	8/2	None	10/5	10/4	8/2
4-to-1 multi-plexer [9]	22	14	1	14	17	24	38	50	7	137	2,261
			2	7	9	12	19	77	40	92	73
			3	5	6	8	12	31	13	41	36
				None	20/5	10/5	5/5	None	20/5	10/5	5/5
8-input NAND	24	20	1	14	17	25	42	21	3	1	49
			2	7	10	13	22	6	35	172	13
			3	5	7	9	15	27	61	137	120
				None	10/4	8 / 3	5 / 2	None	10/4	8/3	5/2
Full adder [9]	28	17	1	16	17	*	*	12	2	*	*
			2	8	10	17	18	6	20	1,335	2,122
			3	6	7	12	12	90	290	1,857	3,137

a.A * indicates that OPBDP did not terminate after one hour.
b.Circuit for $z = (a.b.c + d.e.f + (g + h).(i + j))'$

Table 3: Minimum width layouts and run times obtained by *FCLIP* for various P and N folding limits

(one for every orientation $o \in orients$). As is evident from Table 3, the run times with transistor folding equal those without folding in most cases. In a few of the largest circuits, the run times with folding are over an order of magnitude more than those without folding, mainly due to the additional *Xor* variables.

FCLIP considers each pair separately. However, in most practical designs, transistors that are connected in series in the circuit, called *and-stacks*, must be placed contiguously in the layout for area and performance enhancement. The next section extends *FCLIP* to incorporate and-stack clustering, and shows that this can dramatically reduce run times and extend *FCLIP* to larger circuits.

6. AND-STACK CLUSTERING

An *and-stack* [7] of size n is a group of $n \geq 2$ transistors connected in series. For example, the N transistors N_i, N_{i+1}, ..., N_j in Fig. 9*a* form an and-stack. Since the nets that connect two series-connected transistors, called *internal nets*, do not connect to any other terminal, they do not require diffusion-to-metal contacts, called *straps*, when these transistors are placed contiguously via diffusion sharing. The absence of straps allows the transistors to be placed closer together. Not only does this save cell area, but the resulting smaller diffusion area has better electrical properties as well. Hence, most practical designs place and-stacks contiguously.

We first extend the ILP model (without folding) of Section 4 to implement and-stacks of arbitrary size in the circuit. For each and-stack, we introduce constraints on the relative

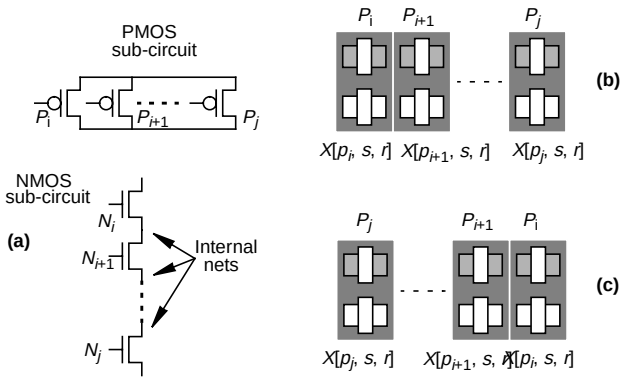


Fig. 9: (a) Stack p_i–p_{i+1}–...–p_j, and its placement in (b) unflipped and (c) flipped orientations

placement of its constituent P/N pairs, and on the diffusion sharing between them. We then show how and-stack incorporation into the transistor folding model *FCLIP* is just a special case of the model without folding.

Inputs and outputs. Let *stacks* be the set of and-stacks in the circuit and let *numStacks* be the total number of stacks. Let *stkSize*[*stk*] specify the size of stack *stk* in terms of the number of its P/N pairs. Let *stkPairs*[*stk*, *stkSize*[*stk*]] contain the list of pairs of stack *stk*, ordered by their connectivity in the stack. Since internal nets will be connected via diffusion sharing, they cannot contribute to cell width or height and hence are dropped from the model.

We introduce new variables to model the placement and orientation of each stack. Let *XrowStk*[*stacks*, *rows*] be binary variables where *XrowStk*[*stk*, *r*] = 1 if stack *stk* is placed in row *r*. Each stack comprising the ordered pairs (p_i, p_{i+1}, ..., p_j) can be placed in two orientations: unflipped (p_i, p_{i+1}, ..., p_j) or flipped (p_j, p_{j-1}, ..., p_i) (Fig. 9). We introduce the binary variable *stkDir*[*stacks*] where *stkDir*[*stk*] = 0 if stack *stk* is placed unflipped, and 1 otherwise.

Constraints. We must ensure that each and-stack is placed in exactly one row, and that its pairs are placed in contiguous slots with diffusion sharing depending on its orientation—unflipped or flipped.

1. *Stack placement*: Each stack must be placed in exactly one row of the 2-D layout.

$$\sum_{r \in rows} XrowStk[stk, r] = 1 \qquad \forall\ stk \in stacks$$

In addition, all pairs of a stack must be placed in the same row as the stack itself.

$$stkSize[stk] \times XrowStk[stk, r] = \sum_{i \in 1..stkSize[stk]} Xrow[stkPairs[stk, i], r]$$

2. *Stack pair placement*: The adjacent pairs in stack *stk* must be placed in contiguous slots where their order depends on the value of *stkDir*. For example, if *stkDir*[*stk*] = 0 (Fig. 9*b*), then the difference in the slot values of pairs p_{i+1} and p_i is 1; if *stkDir*[*stk*] = 1 (Fig. 9*c*), then difference is –1.

$$\sum_{s \in slots, r \in rows} s \times X[stkPairs[stk, i+1], s, r] \; - \sum_{s \in slots, r \in rows} s \times X[stkPairs[stk, i], s, r]$$
$$= 1 - 2 \times stkDir[stk]$$

3. *Stack diffusion sharing*: Adjacent pairs of each stack *stk* must share their diffusions. Again, the order of diffusion sharing depends on the value of *stkDir*[*stk*]. While

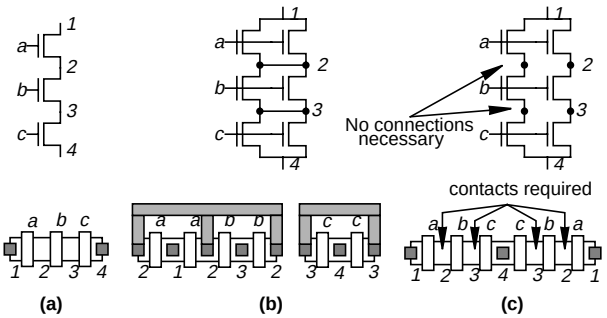


Fig. 10: Effect of folding on the layout of and-stacks: (a) before folding; (b) after folding into 2 legs; and (c) after folding with leg-interlaced placement

merged[p_i, p_{i+1}] = *merged*[p_{i+1}, p_{i+2}] = ... = *merged*[p_{j-1}, p_j] = 1 in the unflipped orientation, the flipped orientation must have *merged*[p_j, p_{j-1}] = *merged*[p_{j-1}, p_{j-2}] = ... = *merged*[p_{i+1}, p_i] = 1. This is modeled by the following constraints, one for each orientation, which imply that the number of consecutive pairs merged in *stk* equal *stkSize*[*stk*] – 1.

$$\sum_{i \in 1..stkSize[stk]-1} merged[stkPairs[stk, i], stkPairs[stk, i+1]]$$
$$= (stkSize[stk] - 1) \times (1 - stkDir[stk])$$

$$\sum_{i \in stkSize[stk]..2} merged[stkPairs[stk, i], stkPairs[stk, i-1]]$$
$$= (stkSize[stk] - 1) \times stkDir[stk]$$

Constraints (10, 11) for *merged*[p_i, p_j], that permit a given pair to be merged with at most one pair on its left and right sides, implicitly ensure that all pairs of an and-stack are merged in the same direction. In order to incorporate folding in the foregoing model, we note that when transistors are folded, it is not always possible to lay out and-stacks contiguously with diffusion sharing. Figure 10*a* shows an and-stack with three transistors and its layout. Now, let each transistor be folded into two legs. The resulting circuit and one possible layout are shown in Fig. 10*b*. Any layout for the folded and-stack requires at least one diffusion gap since net *2* appears on the right side of transistor *b*, and cannot be merged with either the source or drain net of transistor *c*. Therefore, for transistor folding, the stack placement and stack pair placement constraints remain the same, while the stack diffusion sharing constraints are eliminated.

In some designs however, when transistors of and-stacks are folded, the desired placement is as shown in Fig. 10*c*. Here, the legs of the folded transistors are *interlaced*, that is, the order of transistor legs is *a*-*b*-*c*-*c*-*b*-*a* (instead of *a*-*a*-*b*-*b*-*c*-*c*). This placement is possible because the circuit of Fig. 10*b* is electrically equivalent to that of Fig. 10*c* in which the internal nets *2* and *3* do not require explicit connections. Interlaced layouts, like Fig. 10*c*, have several area and performance advantages over non-interlaced layouts:

- Interlacing allows transistor legs to be placed without any diffusion gaps which reduces the cell area.
- Since internal nets do not have to be connected, their diffusion terminals do not need diffusion-to-metal contacts. This allows the legs to be placed closer, reducing diffusion area and routing complexity and, in turn, reducing overall area and enhancing performance.

To incorporate leg-interlacing into *FCLIP*, all three sets of constraints—stack placement, stack pair placement, and

Cct.	# trans	# nets	# rows	*OPBDP* run times (secs.)[a] P/N folding limits							
				None		20 / 5		10 / 5		5 / 5	
8-input NAND	24	17	1	21	**1**	3	**2**	1	**0.4**	49	**29**
			2	6	**4**	35	**2**	172	**0.2**	13	**5**
			3	27	**0.1**	61	**2**	137	**0.4**	120	**0.6**
				None		10 / 4		8 / 3		5 / 2	
Full adder [9]	28	17	1	12	**4**	2	**0.5**	*	**871**	*	**2,300**
			2	6	**1**	20	**37**	1,335	**107**	2,122	**74**
			3	90	**1**	290	**1**	1,857	**25**	3,137	**32**
				None		10 / 10		5 / 5		4 / 4	
Series-parallel cct.[b]	32	32	1	48	**2**	*	**280**	*	**1,735**	6	**1**
			2	49	**1**	*	**16**	*	**57**	1,643	**52**
			3	*	**1**	*	**5**	*	**29**	*	**65**

a.A * indicates that OPBDP did not terminate after one hour.
b.Circuit for $z = (a.b.c.d + e.f.g.h + (i + j)(k + l)(m + n)(o + p))'$

Table 4: *FCLIP* run times without (unshaded columns) and with (shaded columns) and-stack clustering

stack diffusion sharing—are necessary. Only the value of the share variables must be re-computed so as to accurately reflect diffusion sharing between interlaced transistors.

This clearly demonstrates the versatility of our ILP-based optimization technique in handling new design requirements such as folding and leg-interlacing. Other performance-oriented features such as minimizing the total net length, reducing the diffusion area on critical nodes, or maximizing the diffusion area on PWR/GND nodes can also be easily accommodated by suitably modifying the cost function and adding new constraints. In addition, the 2-D width minimization problem solved by *FCLIP* can be extended to address both width and height minimization based on the technique of *CLIP* described in [7].

Experimental Results. Table 4 presents results of *FCLIP* when and-stacking is used. As these results show, run times drop significantly with and-stacking. On the average, and-stacking reduces run times by one or two orders of magnitude. For example, for the largest circuit with 32 transistors, run times in most cases are only a few seconds. The cell widths with and-stacking deviate from those without stacking by only 5% on the average for the circuits tested. Thus, besides being a common performance-oriented requirement in practical designs, and-stacking is very effective in extending *FCLIP* to larger circuits.

7. CONCLUSIONS

FCLIP is the first algorithmic technique that integrates transistor folding into optimal 2-D cell layout generation. Unlike most prior folding methods, *FCLIP* folds transistors before synthesis and is not restricted to 1-D layouts. Via its ILP-based algorithm, *FCLIP* is able to implicitly explore all possible 2-D transistor placements and diffusion sharing possibilities, and so generates a 2-D layout that has the minimum cell width under the modeling assumptions. Its efficient ILP model formulation and off-the-shelf ILP solver make *FCLIP* practical for relatively large circuits with up to 30 transistors. *FCLIP* can also be extended along the lines discussed in [7] to minimize cell height as well.

The ILP-based approach of *FCLIP* is quite versatile, in that new design requirements such as and-stack clustering and leg-interlacing can be easily and efficiently accommodated. In addition, and-stack clustering, while heuristic in nature, is shown to significantly reduce run times, and still yield near-optimal layouts for larger cells.

FCLIP's optimal or near-optimal approach is particularly targeted towards the layout of standard-cells or datapath bit-cells in the design of high-volume, high-performance chips such as microprocessors. These cells, which typically have 20-40 transistors, have three primary layout requirements: (a) they must be optimized as much as possible since they are used multiple times; (b) they must strictly adhere to folding limits set by the technology; and (c) they must not exceed a specified cell width which, in datapaths, is dictated by the datapath-pitch., and therefore may require a 2-D layout. *FCLIP* addresses all these requirements and is an attractive technique for such application domains.

8. REFERENCES

[1] P. Barth, *Logic Based 0-1 Constraint Programming*, Kluwer, Boston, 1995.

[2] Cadence Design Systems, Inc., *Virtuoso Layout Synthesizer Tutorial and Reference*, 1992-94.

[3] S. Chakravarty, X. He, nd S. S. Ravi, "Minimum area layout of series-parallel transistor networks is NP-hard," *IEEE Trans. on CAD*, vol. 10, no. 6, pp. 770-782, June 1991.

[4] C. C. Chen and S. L. Chow, "The Layout Synthesizer: An Automatic Netlist-to-Layout System," *Proc. 26th Design Automation Conf.*, pp. 232-238, June 1989.

[5] R. Fourer, D. M Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Duxbury Press/Wadsworth Publishing, Belmont, CA,1993.

[6] A. Gupta, S-C. The, and J. P. Hayes, "XPRESS: A Cell Layout Generator with Integrated Transistor Folding," *Proc. European Design & Test Conf.*, pp. 393-400, March 1996.

[7] A. Gupta and J. P. Hayes, "CLIP: An Optimizing Layout Generator for Two-Dimensional CMOS Cells," *Proc. 34th Design Automation Conf.*, pp. 452-455, June 1997.

[8] T. W. Her, and D. F. Wong, "Cell Area Minimization by Transistor Folding," *Proc. European Design Automation Conf.*, pp. 172-177, 1993.

[9] D.D. Hill, "Sc2: A Hybrid Automatic Layout System," *Proc. Int'l Conf. on CAD*, pp. 172-174, Nov. 1985.

[10] Y-C Hsieh, et al., "LiB: A CMOS Cell Compiler," *IEEE Trans. on CAD*, vol. 10, pp. 994-1005, Aug. 1991.

[11] C-Y Hwang, et al., "An Efficient Layout Style for Two-metal CMOS Leaf Cells and its Automatic Synthesis," *IEEE Trans. on CAD*, vol. 12, pp. 410-424, March 1993.

[12] E. Malavasi and D. Pandini, "Optimum CMOS Stack Generation with Analog Constraints," *IEEE Trans. on CAD*, vol. 14, pp. 107-122, Jan. 1995.

[13] R. L. Maziasz and J. P. Hayes, *Layout Minimization of CMOS Cells*, Kluwer, Boston, 1992.

[14] C.L. Ong, J.T. Li, and C.Y. Lo, "GENAC: An Automatic Cell Synthesis Tool," *Proc. 26th Design Automation Conf.*, pp. 239-244, June 1989.

[15] J-M. Shyu, A. Sangiovanni-Vincentelli, J. P. Fishburn, and A. E. Dunlop, "Optimization-based Transistor Sizing," *IEEE J. of Solid-State Circuits*, vol. 23, pp. 400-409, Apr. 1988.

[16] K. Tani, et al., "Two-Dimensional Layout Synthesis for Large-Scale CMOS Circuits," *Proc. Int'l Conf. on CAD*, pp. 490-493, Nov. 1991.