CLIP: An Optimizing Layout Generator for Two-Dimensional CMOS Cells*

Avaneendra Gupta†§ and John P. Hayes†

{*avigupta, jhayes*}*@eecs.umich.edu*

† Advanced Computer Architecture Laboratory
Dept. of EECS, University of Michigan
Ann Arbor, MI 48109, U.S.A.

§ Design Technology, Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95052, U.S.A.

Abstract

We present a novel technique CLIP for optimizing both the height and width of CMOS cell layouts in the two-dimensional (2-D) style. CLIP is based on integer-linear programming (ILP) and proceeds in two stages: First, an ILP model is used to determine a 2-D layout of minimum width W_{cell}. Then, another model generates a 2-D layout that has width W_{cell} and requires a minimum number of routing tracks. Run times are in seconds for circuits with up to 16 transistors. For larger circuits, we extend CLIP to a hierarchical method HCLIP that places series-connected transistors contiguously. This reduces run times by up to three orders of magnitude, and still yields optimal results in over 80% of cases.

1 Introduction

The objective of cell layout synthesis is to minimize the cell area subject to constraints. For one-dimensional (1-D) layouts, which use a single pair of parallel P and N diffusion rows, minimizing both cell width and height can yield up to 80% savings in area over width minimization alone [9]. Moreover, height reduction can reduce wire lengths and improve cell performance. In two-dimensional (2-D) layouts, which allow multiple P/N rows, height minimization can have a larger impact on area and performance.

Even in the constrained 1-D style, most techniques that address both width and height minimization are heuristic [1, 4, 8, 11, 12]. Only a few methods [9] are exact in that they explore the entire range of possible layouts. The 2-D style has been relatively little studied and the few techniques proposed are also ad hoc [12, 14, 16]. Tools such as *Virtuoso* [3] support 2-D layout; they use heuristics that can handle large cells but yield sub-optimal layouts.

In [5, 6], the authors proposed an optimal technique based on integer-linear programming (ILP) to generate minimum-width 2-D layouts. While the technique is viable for practical-sized circuits, it does not consider cell height. In this paper, we present a new ILP-based technique called *CLIP* (*Cell Layout via Integer Programming*) that generates 2-D layouts of minimum height from among all layouts of minimum width. We then extend *CLIP* to a hierarchical method *HCLIP* that clusters series-connected transistors, which are placed contiguously in the layout. *HCLIP* reduces run times by up to three orders of magnitude and still yields optimal results in over 80% of the circuits tested.

2 Width Minimization

The 2-D cell layout style is illustrated in Fig. 1; its assumptions are listed in Table 1. If W_r is the width of the r-th row, then **the 2-D cell-width minimization problem** is stated as follows: Place the pairs in a given number of rows such that the maximum width among all rows is minimized, i.e., minimize width W_{cell}, where

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

As discussed in [5], the width W_r of row r is given by:

$$W_r = t_r + c_r - 1 + v_r$$

* This research was supported by a grant from Intel Corporation.

DAC 97, Anaheim, California
(c) 1997 ACM 0-89791-920-3/97/06 ..$3.50

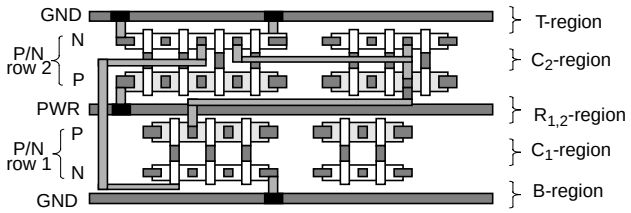

Fig. 1 : The 2-D cell layout style with routing regions

1. Only dual CMOS circuits of fixed structure are considered.
2. Alternate rows are flipped to allow power rails to be shared among adjacent rows.
3. P and N transistors belonging to a pair are vertically aligned so that their terminals that are on common nets can be connected using vertical wires.
4. Intra-cell routing is restricted to polysilicon and metal1.
5. Terminals on adjacent diffusion rows are routed in the channel between the rows.
6. Diffusion gaps do not permit a wire to be routed through them. Also, no wires are routed over diffusions which are strapped with diffusion-to-metal contacts. Hence, routes that span across rows must be routed along the sides of the cell.

Table 1: Assumptions underlying the 2-D cell layout problem

where t_r, c_r, and v_r are the numbers of pairs, chains, and inter-row wires, respectively, in row r. The technique of [5] implicitly models diffusion sharing—it generates an exhaustive set of transistor chains and then determines the smallest subset of chains that covers each pair. However, since the position and orientation of each pair is unknown, the routing and, in turn, the height cannot be determined.

CLIP-W. We now describe an ILP-based width minimization method called *CLIP-W* which explicitly models transistor-pair locations, orientations, and diffusion sharing. The following parameters must be determined to specify a 2-D layout: the row, location, and orientation of each pair, the diffusion sharing among adjacent pairs, and the vertical nets that connect transistor terminals across different P/N rows.

Table 2 lists the input and derived circuit parameters for *CLIP-W*. To represent the position of each pair in the 2-D plane, we introduce place-holders, called *slots*, in each row. For a circuit with *numPairs* pairs, a 2-D placement in *numRows* rows requires at least *maxSlots* = $\lceil numPairs / numRows \rceil$ slots in each row. Slots are numbered in increasing order from the left. We also define sets of integers *slots* = {1, 2,...*maxSlots*}, *rows* = {1, 2, ..., *numRows*}, and *orients* = {1, 2, 3, 4}, the four possible orientations for each pair. The 0-1 array *share* is such that *share*[p_i, o_i, p_j, o_j] = 1 if pairs p_i and p_j can share diffusions in orientations o_i and o_j, respectively, when p_j is placed to the right of p_i.

The basic variables for each pair are represented by 0-1 arrays *X*[*pairs*, *slots*, *rows*] and *Xor*[*pairs*, *orients*]. While *X*[*p*, s, *r*] = 1 implies that pair *p* is placed in the *s* slot of row *r*, *Xor*[*p*, *o*] = 1 states that *p* is placed in orientation *o*. To model diffusion sharing, we define *nogap*[*slots*, *rows*] where *nogap*[s, *r*] = 1 if adjacent slots *s* and *s* + 1 do not have a gap between them.

The goal (cost function) of *CLIP-W* is to minimize W_{cell}, where $W_{cell} \geq W_r$ for each row *r*. Now, W_r can be derived as follows:

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

The constraints of *CLIP-W* are described below.

1. *Pair inclusion*: A pair must be placed in one slot and orientation.

$$\sum_{s \in slots} \sum_{r \in rows} X[p, s, r] = 1 \qquad \text{for all } p \in pairs$$

Parameters	Interpretation
1. *numPairs, numRows, maxSlots*	The number of pairs, rows, and slots
2. *pairs, rows, slots, nets*	The set of pairs, rows, slots, and nets
3. *PpairNets, NpairNets*	*PpairNets*[*p*] = {gate, source, drain nets of P trans. of pair *p*} (*NpairNets* is similarly defined for N trans.)
4. *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)
5. *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)
6. *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 (1–3) and derived (4–6) parameters for *CLIP-W*

$$\sum_{o \in orients} Xo[p, o] = 1 \qquad \text{for all } p \in pairs$$

2. *Slot occupancy*: We force the first slot in each row to be filled with exactly one pair, and slots to be filled in a left-justified order, i.e., slot *s* should be occupied before its neighboring slot *s* + 1.

$$\sum_{p \in pairs} X[p, 1, r] = 1 \qquad \text{for all } r \in rows$$

$$\sum_{p \in pairs} X[p, s-1, r] \geq \sum_{p \in pairs} X[p, s, r] \qquad \text{for all } r \in rows,\ s \in slots$$

3. *Diffusion sharing*: *Nogap*[*s, r*] can be defined by the following logic equation, for every $p_i, p_j \in pairs$ and $o_i, o_j \in orients$:

$$\begin{aligned} &nogap[s, r] \qquad (1)\\ &= \textbf{or}\ (p_i \text{ is in slot } s \text{ of row } r\ \textbf{and}\ p_j \text{ is in slot } s+1 \text{ of row } r\\ &\quad \textbf{and}\ p_i \text{ in orient } o_i\ \textbf{and}\ p_j \text{ in orient } o_j \text{ and } share[p_i, o_i, p_j, o_j] = 1)\\ &= \textbf{or}\ \{X[p_i, s, r]\ \textbf{and or}\ \{X[p_j, s+1, r]\ \textbf{and}\ merged[p_i, p_j]:\\ &\quad \text{for all } p_j \in pairs\}: \text{for all } p_i \in pairs\} \end{aligned}$$

Here *merged*[p_i, p_j] = 1 if p_i can share diffusion with p_j, that is,

$$merged[p_i, p_j] = \textbf{or}\ \{Xor[p_i, o_i]\ \textbf{and}\ Xor[p_j, o_j]: \quad (2)$$
$$\text{for all } o_i, o_j \in orients \text{ such that } share[p_i, o_i, p_j, o_j] = 1\}$$

To prevent cyclic conditions in diffusion sharing, we ensure that a pair can share diffusion with at most one pair on either side:

$$\sum_{p_2 \in pairs} merged[p_1, p_2] \leq 1 \qquad \text{for all } p_1 \in pairs \quad (3)$$

$$\sum_{p_1 \in pairs} merged[p_1, p_2] \leq 1 \qquad \text{for all } p_2 \in pairs \quad (4)$$

The logical constraints (1, 2) are linearized in the final ILP model.

4. *Inter-row connectivity*: This is modeled as described in [5].

Experimental results. Table 3 presents results of solving *CLIP-W* for optimum-width 2-D layouts with the 0-1 solver *OPBDP* [2].

The optimal cell widths obtained by *CLIP-W* are about 15% smaller than those produced by the commercial tool *Virtuoso*; *Virtuoso*'s run times are in seconds in all cases. For circuits with as many as 24 transistors, *CLIP-W* has run times that are in seconds for minimum-width 2-D layouts. Run times can be reduced—by up to three orders of magnitude—by using hierarchical methods such as the circuit clustering approach proposed in Section 6.

In subsequent sections, we propose a model that extends *CLIP-W* to minimize the 2-D cell height in addition to the width.

3 Height Minimization

The height of a cell is determined by its *horizontal routing* (*track*) *density* [9], that is, the number of tracks needed to complete the cell's routing. This, in turn, depends on factors such as the layout style, the usage of metal, polysilicon, and diffusion layers, and the use of jogs and vias. The layout assumptions underlying the height minimization problem are summarized in Table 1.

Track density can be determined from the horizontal span of each net. A 2-D cell is composed of several routing regions as illustrated by the two-row layout in Fig. 1 which has five regions: B (cell bottom), C_1 (P/N channel of the first row), $R_{1,2}$ (*inter-row channel* between the P/N rows), C_2 (P/N channel of the second

#	Circuit	No. of trans.	No. of rows	Cell width		CPU time (secs)[a]	
				CLIP-W[b]	*Virtuoso*	*CLIP-W*	*HCLIP*
1.	Non-series-parallel bridge circuit [16]	10	1	6	6	0.03	0.03
			2	5	5	0.09	0.09
			3	4	5	0.07	0.07
			4	4	5	0.19	0.19
2.	2-to-1 multiplexer	14	1	8	8	0.06	0.04
			2	4	5	0.25	0.06
			3	3	3	0.06	0.04
			4	3	3	0.25	0.09
3.	Majority function $z = a.b + b.c + a.c$	18	1	10	10	0.2	0.05
			2	5	6	0.2	0.02
			3	4	5	12	0.07
			4	4	5	8	1
4.	Series-parallel circuit for $z = (a.b.c.d + e.f.g.h + (i + j).(k + l))'$ [9]	24	1	13	14	0.3	0.1
			2	9	10	11	0.7
			3	5	6	10	0.1
			4	5 (6)	6	390	0.1
5.	8-input NAND circuit	24	1	14	15	19	1
			2	7	8	9	5
			3	5	6	43	0.1
			4	4	4	58	0.7
6.	Series-parallel circuit for $z = (abcd + efgh + (i + j)(k + l)(m + n)(o + p))'$ [9]	32	1	18	19	79	4
			2	9 (10)	10	2	1
			3	8	10	2446	1
			4	6	9	5216	1

a. Run times are with *OPBDP* using its -h103 variable selection heuristic.
b. Numbers in brackets refer to the cell width with *HCLIP*, when different from the optimum value for the original circuit.

Table 3: Minimum width 2-D placements with run times

row), and T (cell top). We assume that all nets in the P and N diffusions of the same P/N row are routed in its P/N channel. Thus, a net that appears on both sides of an inter-row channel is routed to connect just one of its terminals on each side.

In order to determine the track density in each region, the cell layout is considered to be made of vertical columns, where each column represents a transistor terminal. Then, the number of tracks T_R required in a routing region *R* can be expressed as follows:

$$T_R = \textbf{max}\ \{\text{number of nets that span column } c\text{: } c = 1, 2, 3, ...\} \qquad (5)$$

Assuming equal transistor sizes, we define the total height H_{cell} of the cell as follows:

$$H_{cell} = \sum_{R \in regions} \text{No. of tracks in routing region } R = \sum_{R \in regions} T_R$$

T_R depends on the nets that occur in each vertical column which, in turn, depends on the position and orientation of each transistor. Thus, to compute T_R, we must determine the nets that must be routed horizontally in each column. This has traditionally been called the *channel routing* problem [13]. While channel routing considers both horizontal and vertical constraints, cell synthesis methods have generally ignored vertical constraints [11, 9].

The fundamental problem is to determine whether a net *n* requires a track in a column *c*. We define 0-1 variables *net* such that *net*[*n, c, r*] = 1 if net *n* exists on a terminal in column *c* of row *r*. We define 0-1 variables *span* such that *span*[*n, c, r*] = 1 if net *n* requires a track in column *c* of row *r*. Then, the total number of nets that span column *c* of row $r = \Sigma\ span[n, c, r]$, $n \in nets$. We use a dynamic programming approach to define the following two conditions under which *span*[*n, c, r*] = 1:

- If *net*[*n, c, r*] = 0, then (*span*[*n, c* – 1, *r*] = 1 **and** *net*[*n*, c_2, *r*] = 1 for some $c_2 > c$)
- If *net*[*n, c, r*] = 1, then (*span*[*n, c* – 1, *r*] = 1 **or** *net*[*n*, c_2, *r*] = 1 for some $c_2 > c$)

If *x* = *net*[*n, c, r*], *y* = *span*[*n, c* – 1, *r*], and *z* = **or** {*net*[*n*, c_2, *r*]: $c_2 > c$}, *span*[*n, c, r*] can be defined by the logic equation below:

$$span[n, c, r] = \bar{x}\,.\,(y\,.\,z) + x\,.\,(y + z) = \textbf{majority}(x, y, z) \qquad (6)$$

The majority-function constraint is equivalent to the following pair of linear inequalities:

$$span[n, c, r] \geq (x + y + z - 1)/2$$
$$span[n, c, r] \leq (x + y + z)/2$$

Based on the above algorithm, we now present an ILP model *CLIP-WH* that minimizes both the cell width and height.

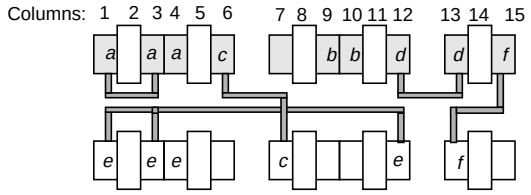

Fig. 2 : Routing track requirements with and without diffusion gaps

4 Width and Height Minimization

The **2-D cell width and height minimization problem** is defined as follows: Place the transistors in a given number of rows such that the layout has a minimum height among all layouts with minimum width. The ***CLIP* method** proceeds in two stages:

1. Use *CLIP-W* to find W_{cell}, the minimum 2-D cell width.
2. Use *CLIP-WH* to find a 2-D layout of minimum height H_{cell} from among all layouts of minimum width W_{cell}.

CLIP-WH. We define one column for each of the three terminals—source, gate, and drain—of the transistors placed in a row. Figure 2 shows how the columns are numbered in a given row.

In addition to the parameters of *CLIP-W* and W_{cell}, we have *maxCols*, the total number of columns in each row. Let the set *cols* = {1, ..., *maxCols*}. We define arrays *net*[*nets*, *cols*, *rows*] and *span*[*nets*, *cols*, *rows*] of 0-1 variables as described in Section 3. The variables *height*[*rows*] and *interRow*[*rows*] are used to represent the height of the P/N and inter-row channels, respectively; since they are integers, they are represented using arrays of boolean variables.

The goal of *CLIP-WH* is to minimize the cell height H_{cell}, i.e.,

$$\text{minimize } H_{cell} = \sum_{r \in \text{ P/N channels}} height[r] + \sum_{r \in \text{ inter-row channels}} interRow[r]$$

We now describe the constraints in *CLIP-WH*; these are in addition to the constraints of *CLIP-W.*

1. The constraint on the height *height*[*r*] of the P/N channel in row *r*, defined by equation (5), is linearized as follows:

$$height[r] \geq \sum_{n \in nets} span[n, c, r] \qquad \text{for all } c \in cols$$

2. *Cell width*: For each row *r*, we ensure that its width is $\leq W_{cell}$.

$$W_{cell} \geq 2 \times \sum_{p \in pairs} Xrow[p, r] - \sum_{s \in slots} nogap[s, r] + v_r - 1$$

3. *Net presence*: If *c* represents a diffusion terminal (*c* = 1, 3, 4, 6, ...), *net*[*n*, *c*, *r*] depends on the pair placed in the corresponding slot, and its orientation. For example, for *c* = 4, *net*[*n*, 4, *r*] = 1 if there is a pair placed in slot *s* = 2, and its orientation causes its diffusion terminal on net *n* to appear on its left. Thus, if *c* is a left diffusion column (*c* = 1, 4, 7, ...), *net*[*n*, *c*, *r*] is given by:

$$\begin{aligned} net[n, c, r] \geq X[p, (c+2)/3, r] \ \textbf{and} \qquad & \text{for all } p \in pairs \\ (Nsrc[p, n] \ \textbf{and} \ (Xor[p, 1] \ \textbf{or} \ Xor[p, 3]) \\ \textbf{or} \ Ndrn[p, n] \ \textbf{and} \ (Xor[p, 2] \ \textbf{or} \ Xor[p, 4]) \\ \textbf{or} \ Psrc[p, n] \ \textbf{and} \ (Xor[p, 1] \ \textbf{or} \ Xor[p, 2]) \\ \textbf{or} \ Pdrn[p, n] \ \textbf{and} \ (Xor[p, 3] \ \textbf{or} \ Xor[p, 4])) \end{aligned}$$

Net[*n*, *c*, *r*] is similarly defined for right diffusion columns (*c* = 3, 6, ...). If *c* is a gate column (*c* = 2, 5, ...), *net*[*n*, *c*, *r*] is independent of the orientation of the pair placed in that slot, and is given by:

$$net[n, c, r] = \sum_{p \in pairs} \textbf{if } (Ngate[p, n] \ \textbf{or} \ Pgate[p, n]) \ \textbf{then } X[p, (c+1)/3, r]$$

Since the above constraint is an equation, it can be directly incorporated into the definitions for *span*[*n*, *c*, *r*], thereby eliminating the variables *net*[*n*, *c*, *r*] for *c* = 2, 5, 8, etc.

4. *Net span*: The constraint for *span*[*n*, *c*, *r*] is given by equation (6). However, a few special cases, illustrated in Fig. 2, must be considered to accommodate the presence of diffusion gaps.

- Net *a* requires a track in columns 1, 2, and 3; hence, *span*[*a*, 1, *r*] = *span*[*a*, 2, r] = *span*[*a*, 3, *r*] = 1. Also, since columns 3 and 4 are connected by diffusion sharing, we set *span*[*a*, 4, *r*] = 1.

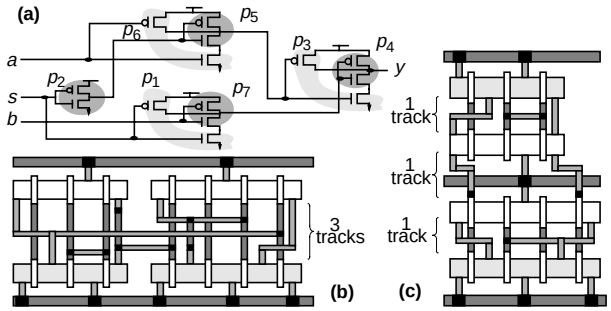

Fig. 3 : (a) 2-to-1 multiplexer and its layouts in (b) one and (c) two rows

- However, net *b* appears only in columns 9 and 10 that share diffusion. Hence, *span*[*b*, 9, r] = *span*[*b*, 10, *r*] = 0.
- For net *c* that appears in columns 6 and 7 separated by a diffusion gap, *span*[c, 6, *r*] = *span*[*c*, 7, *r*] = 1.
- For nets such as *d* that appear on the same (P or N) diffusion of two pairs separated by a gap, *span*[*d*, 12, *r*] = *span*[*d*, 13, *r*] = 1.

Thus, the constraint for *span*[*n*, *c*, *r*] for *c* = 3, 6, 9, etc. is split into two: while (7) considers all columns $c_2 \geq c + 2$, (8) considers column *c* + 1 on its right and takes into account the absence of a gap, represented by *nogap*[*c*/3, *r*], between columns *c* and *c* + 1.

$$span[n, c, r] = \textbf{majority}(span[n, c-1, r], net[n, c, r], \ \textbf{or} \ \{net[n, c_2, r]: c_2 \geq c+2\}) \quad (7)$$

$$span[n, c, r] = \textbf{majority}(span[n, c-1, r], net[n, c, r], (net[n, c+1, r] - nogap[c/3, r])) \quad (8)$$

In a dual CMOS circuit, since each pair has a common gate net, the track density of a gate column *c* is no greater than the maximum of the track densities of its adjoining diffusion columns *c* – 1 and *c* + 1. Hence, the variables *span*[*n*, *c*, *r*] are eliminated for gate columns, and constraints (7, 8) are suitably modified to consider *span*[*n*, *c* – 2, *r*] instead of *span*[*n*, *c* – 1, *r*].

5. *Inter-row channel routing*: As discussed earlier, the routing problem in an inter-row channel *r* is defined as follows: For each net that appears on both sides of *r*, select one terminal to be connected from each side such that the overall track density in *r* is minimized. *CLIP-WH* assumes that each net that appears on both sides of an inter-row channel requires a separate routing track. Hence, the density of an inter-row channel *r* is equal to the total number of nets on both sides of *r*. For example, the two-row layout in Fig. 3*b* requires two nets to be routed in its inter-row channel. Although both nets can be routed in one track, *CLIP-WH* assigns two tracks, one for each net. Thus, we have

$$interRow[r] = \sum_{n \in nets} (n \text{ on top diffusion } \textbf{and } n \text{ on bottom diffusion})$$

Both the terms in the above equation are available as variables that model inter-row connectivity, and can be used directly.

5 Experimental Results

We have applied *CLIP-WH* to the circuits of Table 3. The cell height H_{cell} obtained and the run times of *CLIP-WH* using *OPBDP* are given in Table 4. In most cases, the increase in H_{cell} is much less than the decrease in W_{cell} when the layout changes from one to two rows. This can translate into significant area savings. These savings are less pronounced for three and four-row layouts since both W_{cell} and the number of tracks are seen to change very little while the number of transistor rows increases. As an example, Fig. 3*a* shows a 2-to-1 multiplexer with its seven P/N pairs highlighted. Its optimal *CLIP-WH* layouts in one and two rows are shown in Figs. 3*a* and *b*, respectively.

CLIP-WH's overall run times for optimum layouts are in seconds for medium-sized circuits with up to 16 transistors. Moreover, an optimal solution is found in a relatively short time; the remaining time is utilized in verifying optimality. Hence, the ILP solver may be prematurely terminated to yield near-optimal, or

possibly even optimal, solutions in practical time. For larger circuits, we propose a practical hierarchical method *HCLIP* in the next section that extends our technique to circuits with over 30 transistors while yielding layouts that are at or near the optimum.

6 Circuit Clustering

An *and-stack* [5] of size n is a group of $n \geq 2$ transistors connected in series. The circuit in Fig. 3*a* has three pairs with and-stacks: (p_1, p_7), (p_3, p_4), and (p_5, p_6). Since the nets that connect two series-connected transistors (*internal nets*) do not connect to any other terminal, they do not require straps when these transistors are placed using diffusion abutment. This allows the transistors to be placed closer, which reduces area and enhances performance. Hence, most designs lay out and-stacks as single contiguous units.

HCLIP. We now outline *HCLIP* (*Hierarchical CLIP*), an extension of *CLIP* that efficiently implements and-stacks of arbitrary size. For each stack S with pairs p_i–p_{i+1}–...–p_j, it introduces constraints on the relative placement and diffusion sharing of its constituent pairs.

Let *stacks* be the set of and-stacks. Let *Ssize*[S] specify the number of pairs in stack S with *Spairs*[S, *Ssize*[S]] containing its list of pairs, ordered by their connectivity in S. We define 0-1 variables *Srow*[*stacks*, *rows*] where *Srow*[S, r] = 1 if stack S is placed in row r. Further, we define variables *Sdir*[*stacks*] where *Sdir*[S] = 0 if stack S is placed unflipped (p_i–p_{i+1}–...–p_j), and 1 if flipped (p_j–p_{j-1}–...–p_i). The constraints described next are in addition to those in *CLIP-WH*.

1. *Stack placement*: Each stack must be placed in exactly one row.

$$\sum_{r \in rows} Srow[S, r] = 1 \qquad \text{for all } S \in stacks$$

Also, all pairs of a stack S must be placed in the same row as S.

$$Ssize[S] \times Srow[S, r] = \sum_{i \in 1..Ssize[S]} Xrow[Spairs[S, i], r]$$

2. *Stack pair placement*: Adjacent pairs of a stack S must be placed in contiguous slots. If *Sdir*[S] = 0, then the slot values of pairs p_{i+1} and p_i must differ by 1; if *Sdir*[S] = 1, then this difference is –1.

$$\sum_{s \in slots} \sum_{r \in rows} s \times X[Spairs[S, i+1], r] \qquad \text{for all } i \in Ssize[S]$$

$$-\sum_{s \in slots} \sum_{r \in rows} s \times X[Spairs[S, i], r] = 1 - 2 \times Sdir[S]$$

3. *Stack diffusion sharing*: Adjacent pairs of S must share their diffusions. While *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.

$$\sum_{i \in 1..Ssize[S]-1} merged[Spairs[S, i], Spairs[S, i+1]] = (Ssize[S] - 1) \times (1 - Sdir[S])$$

$$\sum_{i \in Ssize[S]..2} merged[Spairs[S, i], Spairs[S, i-1]] = (Ssize[S] - 1) \times Sdir[S]$$

Constraints (3, 4) for *merged*[p_i, p_j], that permit a pair to be merged with at most one pair on its left and right sides, implicitly ensure that all pairs of a stack are merged in the same direction.

Experimental results. Table 4 presents the values of W_{cell}, and H_{cell} and the associated run times obtained with *HCLIP*. Where possible, we have compared these values with the corresponding optimum values for the non-hierarchical layout. For the circuits presented, W_{cell} with and-stacking is optimum in all but one case. In addition, the number of tracks (H_{cell}) required with and-stacking is optimum in over 80% of cases. Also, cell heights of *HCLIP* are 25% smaller on the average than those obtained using *Virtuoso*.

The run times of *HCLIP* are up to three orders of magnitude better than *CLIP*'s. Also, the first optimum solution is found in just a few seconds. Thus, and-stacking can extend the ILP-based technique to larger circuits while still yielding layouts whose widths and heights are at or near the optimum.

Cct. #	No. of trans.	No. of rows	Cell layout: *CLIP*[b] Opt. W_{cell}	*CLIP*[b] Opt. H_{cell}	*Virtuoso* W_{cell}	*Virtuoso* H_{cell}	CPU time (secs)[a]: *CLIP-WH* First opt. sol.	*CLIP-WH* Final sol.	*HCLIP* First opt. sol.	*HCLIP* Final sol.
1.	10	1	6	4	6	4	0.1	0.5	0.1	0.5
		2	5	3	5	3	2	2.5	2	3
		3	4	3	5	4	4	5	4	5
		4	4	3	5	4	2	2	2	2
2.	14	1	8	3	8	3	0.3	1	0.05	0.2
		2	4	4	5	4	0.5	2	0.5	0.6
		3	3	5	3	7	1	8	0.3	1
		4	3	5	3	5	5	17	4	7
3.	18	1	10	5	10	6	0.8	5	0.2	0.5
		2	5	(7)	6	7	*	*	0.3	1
		3	4	6	5	7	237	3673	0.6	2
		4	4	(7)	5	8	*	*	0.3	15
4.	24	1	13	3	14	3	2	73	0.2	1.7
		2	9	(5)	10	7	*	*	9	16
		3	5	(7)	6	8	*	*	1	4
		4	5	(8)	6	9	*	*	3	55
5.	24	1	14	2	15	3	340	4,698	1	9
		2	7	(4)	8	6	*	*	15	53
		3	5	(4)	7	6	*	*	36	59
		4	4	(5)	4	7	*	*	2	31
6.	32	1	18	(4)	19	4	*	*	200	695
		2	9 (10)	(6)	10	7	*	*	1	10
		3	8	(8)	10	11	*	*	809	930
		4	6	(8)	9	14	*	*	65	410

a. Run times are with *OPBDP* using its -h3 variable selection heuristic. An asterisk implies that *OPBDP* did not terminate after 5,000 seconds.
b. Numbers in brackets are the width or height obtained with *HCLIP*, when different from the optimum value, if known, for the original circuit.

Table 4: Minimum width and height 2-D layouts with run times

7 Conclusions

We have presented a novel technique *CLIP* for simultaneous height and width minimization of 2-D cell layout. It combines diffusion sharing, inter-row connectivity, and routing density in a common problem space that can be efficiently searched for optimal solutions using branch-and-bound methods such as those of ILP solvers. When used with and-stack clustering, it generates optimal or near-optimal layouts for practical-sized circuits in seconds.

8 References

[1] D. G. Baltus and J. Allen, "SOLO: A Generator of Efficient Layouts From Optimized MOS Circuit Schematics," *Proc. 25th Design Automation Conf.*, pp. 445-452, June 1988.

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

[3] Cadence Design Systems, *Virtuoso Layout Synthesizer*, 1992-94.

[4] 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.

[5] A. Gupta and J. P. Hayes, "Width Minimization of Two-Dimensional CMOS Cells Using Integer Programming," *Proc. Int'l Conf. on CAD*, pp. 660-667, Nov. 1996.

[6] A. Gupta and J. P. Hayes, "A Hierarchical Technique for Minimum-Width Layout of Two-Dimensional CMOS Cells," *Proc. Int'l Conf. on VLSI Design*, pp. 15-20, Jan. 1997.

[7] D. V. Heinbuch, *CMOS3 Cell Library,* Addison-Wesley, Reading, Mass., 1988.

[8] Y-C Hsieh, C-Y Hwang, Y-L Lin, and Y-C Hsu, "LiB: A CMOS Cell Compiler," *IEEE Trans. on CAD*, Vol. 10, pp. 994-1005, Aug. 1991.

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

[10] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley, New York, 1988.

[11] 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.

[12] C.J. Poirier, "Excellerator: Custom CMOS Leaf Cell Layout Generator," *IEEE Trans. on CAD*, Vol. 8, pp. 744-755, July 1989.

[13] R. L. Rivest and C. M. Fiduccia, "A 'Greedy' Channel Router," *Proc. 19th Design Automation Conf.*, pp. 120-125, June 1991.

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

[15] T. Uehara and W.M. VanCleemput, "Optimal Layout of CMOS Functional Arrays," *IEEE Trans. on Computers*, vol. C-30, pp. 305-312, May 1981.

[16] H. Zhang and K. Asada, "An Improved Algorithm of Transistors Pairing for Compact Layout of Non-Series-Parallel CMOS Networks," *Proc. Custom Integrated Circuits Conf.*, pp. 17.2.1-17.2.4, 1993.