Doubly Folded Transistor Matrix Layout

Lukas P.P.P. van Ginneken, Jos T.J. van Eijndhoven, Jos A.H.C.M. Brouwers

Design Automation Section, Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
tel: 31-40-473710, fax: 31-40-455925

ABSTRACT - We present a flexible module generator that lays out transistor net lists. New is the formulation of this layout problem as a two dimensional folding problem. The folding algorithm uses an elegant hierarchical divide and conquer technique. The aspect ratio and pin positions can be controlled within a wide range, while the area remains approximately constant. Accurate control of the aspect ratio and pin positions is important in combination with top down floor planning. The mask generator uses a small library of adaptable transistors with parameters like length, width, positions of the terminals, and an optional diffusion implant. Compared to other automated layout styles, the new module generator makes smaller and more flexible layouts. The layout of the modules can be customized with respect to all major design parameters: function, speed, design rules, aspect ratio and pin positions.

1. INTRODUCTION

Stepwise refinement is an approach for the design of complex systems. It implies the adaptation of smaller sub structures to a more global structure that was designed earlier. In the design of VLSI the approach leads to floor planning before module design [7]. From the floor plan important design parameters such as the aspect ratio of the modules and the positions of the terminals can be derived. The best aspect ratio of the blocks can be computed in polynomial time after the floor plan topology has been determined [9]. In an automated environment module generators must be capable of taking these parameters into account.

A good module generator must be able to design a large class of modules for different design parameters [12]. To have a maximum of flexibility in generating different circuits, the functional specification of the circuit is given as a net list of transistors. The same problem can be solved with the gate matrix layout style [6, 8], but we do not require transistors with the same gate signal to be on the same polysilicon strip. This requirement results in complex optimization problems and hampers efficient compaction. By dropping this requirement we arrive at a more elegant and symmetrical optimization problem: the two dimensional folding problem. Like the gate matrix style, this new layout style can handle any circuit of transistors, while the transistor sizes may vary. In addition, it can control the shape and pin positions accurately.

The two dimensional folding problem is the problem of assigning horizontal and vertical strips to rows and columns. The strips have an interconnection pattern that must be realized, but the order of the connections is unconstrained. Of course the strips in the same row respectively column must not overlap. The vertical strips represent the signal nets, while the horizontal strips represent the transistors.

The two dimensional folding problem was first posed in [3] and a simulated annealing solution was proposed. An improved simulated annealing algorithm was described in [11]. Our solution to this optimization problem uses a hierarchical divide and conquer approach [10] similar to the approach of [1] to placement and routing. The algorithm allows for a large amount of freedom in choosing the aspect ratio and pin positions. Different amounts of folding in both dimensions are used to get the desired aspect ratio.

2. THE CIRCUIT LAYOUT PROBLEM

The circuit to be layed out is represented as a net list: a circuit C is a bipartite graph $C(T,N,P)$. T is the set of transistors, N is the set of signal nets and P is the set of pins. The set of pins $P \subset T \times N$ contains the edges of the graph, which stand for the transistor terminals.

The folding algorithm assigns the signal nets to columns and the transistors to rows, such that they do not overlap. The signal nets are implemented in metal, while the transistors use diffusion and polysilicon. A couple of configurations of the pins of the transistor are possible. The gate can be the middle pin, or it can be one of the other two (see fig.1). Another possibility is that the gate is connected to the source or the drain.

Since the folding algorithm assumes that all transistors are equally high, it is desirable that the transistors have a small and almost uniform height. The height of the transistors should preferably be close to the width of a horizontal polysilicon or diffusion wire. Notice that any pin order of the transistor can be realized. This leaves total freedom to the folding algorithm to choose the order of the signals.

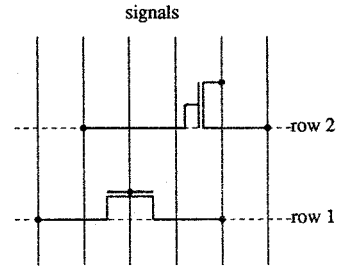

Figure 1. Transistors assigned to rows

We formally state the circuit layout problem as follows: The circuit is to be realized on a grid of *rows* and *columns*. The set of grid points is represented by $Z \times Z$. The layout of a circuit is determined by a *net assignment function* $\phi: N \rightarrow Z$ that assigns nets to columns and a *transistor assignment function* $\psi: T \rightarrow Z$ that assigns transistors to rows. Let $\nu(t)$ denote the set of connected transistors of net n: $\nu(n) = \{t \in T \mid (t,n) \in P\}$. The *span* σ of a net $n \in N$ is an interval of

rows defined as $\sigma(n) = [\min_{t \in v(n)} \psi(t), \max_{t \in v(n)} \psi(t)]$. The spans of nets that are assigned to the same column are not allowed to overlap:

$$\forall_{n_i, n_j \in N} [\phi(n_i)=\phi(n_j) \Rightarrow \sigma(n_i) \cap \sigma(n_j)=\emptyset]$$

Since the problem is symmetric the same goes for the transistors: $v(t)$ represents the set of nets connected to transistor t and $\sigma(t)$ represents the span of a transistor. In the remainder of this paper this duality will not be explicitly mentioned. The objective of the folding algorithm is to find a valid ϕ and ψ subject to some cost function, for instance area.

The pin positions to the periphery are handled by introducing four pseudo strips. Each of these strips is assigned to a predetermined side of the matrix during the folding. In this way the pins are forced to the correct side. In addition to that it is possible to enforce certain constraints on the order of the peripheral pins.

We may classify gate placement, which has been shown to be NP-complete [5], as a special case of two dimensional folding. Therefore, the assumption that two dimensional folding is at least NP-complete seems valid, although proof is not provided here.

3. THE FOLDING ALGORITHM

The algorithm that was used is a divide and conquer heuristic [10]. The design is repeatedly subdivided by straight orthogonal cutting lines. After each vertical division it is decided which nets are placed to the right of the cutting line and which are placed to the left. After each horizontal division the transistors are partitioned into two groups. After the kth horizontal cut the transistors are partitioned into k+1 subsets $T_0..T_k$.

$$T = \bigcup_{i=0}^{k} T_i \qquad \forall_{T_i, T_j} [T_i \cap T_j = \emptyset]$$

Similarly the nets are partitioned into $\ell+1$ subsets $N_0,..,N_\ell$ after ℓ vertical cuts. The sets are ordered in the grid space, that is, the sets imply a constraint on the functions ϕ and ψ.

$$\forall_{n \in N_i, h \in N_j} [i<j \Rightarrow \phi(n)<\phi(h)]$$

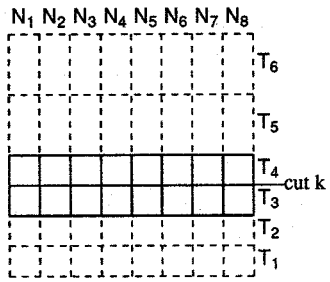

Figure 2. A 2xn subproblem

When a subset N_i is partitioned into two subsets N_i and N_{i+1} this implies a restriction of ϕ (although a solution remains always possible). The index of remaining sets N_j with $j>i$ increases by one. As the exact assignment has not yet been determined the span of a net will be defined as

$$\bar{\sigma}(n) = [\min\{i \mid T_i \cap v(n) \neq \emptyset\}, \max\{i \mid T_i \cap v(n) \neq \emptyset\}]$$

To make an estimate of the resulting size of the array, and to evaluate the consequences of cutting line decisions, we use bounds on the number of rows needed for a subset of transistors, or the number of columns needed for a subset of nets. The number of columns needed is equal to the number of overlapping nets. In the worst case all nets that can overlap actually will overlap. The maximum number of columns needed for a subset of nets is therefore given by

$$\mu(N_i) = \max_j \#\{n \in N_i \mid j \in \bar{\sigma}(n)\}$$

Notice that this is the exact number of columns if $\sigma=\bar{\sigma}$. A lower bound for the number of columns is determined by the number of nets that cross horizontal boundaries:

$$\delta(N_i) = \max_j \#\{n \in N_i \mid j \in \bar{\sigma}(n) \wedge j+1 \in \bar{\sigma}(n)\}$$

Since the terminals of a transistor are not allowed to overlap there is another lower bound:

$$\gamma(N_i) = \max_{t \in T} \#(v(t) \cap N_i)$$

These upper and lower bounds can be used to estimate the final size of the array. A dimension of the matrix is estimated as the mean of the upper and lower bounds. The area can be estimated as

$$\left(\sum_{i=0}^{\ell} \frac{\max(\delta(N_i), \gamma(N_i)) + \mu(N_i)}{2}\right) \cdot \left(\sum_{i=0}^{k} \frac{\max(\delta(T_i), \gamma(T_i)) + \mu(T_i)}{2}\right)$$

When a cut is made the transistors or nets are partioned into two subsets. The heuristics transfer strips between the two subsets while trying to improve the area. The heuristic uses a control method commonly used in mincut algorithms [4].

The shape is controlled by selecting the orientation of the next cutting line. A vertical cutting line tends to make the array wider and lower, a horizontal line makes the array higher and narrower. The orientation is chosen such that the estimated array size is corrected in the direction of the desired aspect ratio. This is an accurate control mechanism because the repeated cuts allow for corrections that gradually become smaller.

4. ASSIGNMENT WITHIN PARTITIONS

When the following criterion is satisfied, further cuts cannot improve the result.

$$\forall_{S \in \{T_i, N_i\}} [\mu(S)=1 \vee \mu(S)=\max(\gamma(S), \delta(S))]$$

When a subset S does not have mu(S)=1 then the elements of this set do not have a completely determined assignment function. An assignment with a minimum number of rows or columns can easily be constructed using a left edge algorithm. The left edge algorithm places two nets with an overlapping span on different columns.

$$\forall_{n_1, n_2 \in N_i} [\bar{\sigma}(n_1) \cap \bar{\sigma}(n_2) \neq \emptyset \rightarrow \phi(n_1) \neq \phi(n_2)]$$

This assignment realizes the upperbound μ. Notice that it is now possible to exchange the columns within a subset without creating any overlaps between horizontal strips. This freedom can be used to further optimize the net assignment with respect to another criterion.

When extra long or extra wide transistors are used, it is advantageous if enough space is available between the pins to allow a horizontal orientation. In fig.3 the transistor to the left does not have enough space to lie flat. To the right, enough space between the terminals is created for a flat implementation by exchanging two signal nets.

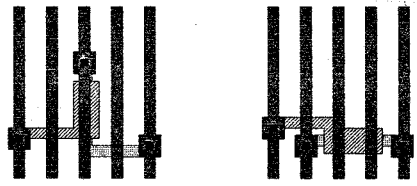

Figure 3. The influence of exchanging two signals

Let for each transistor t the nets connected to gate, drain and source be $\{g_t, d_t, s_t\}=v(t)$ and let $w(t), l(t)$ denote the width and length of t. The number of columns needed by transistor t is

$$S(t)=\text{entier}\left(\frac{\max(w(t)+c_1, l(t)+c_2)}{p}+1\right).$$

p is the metal pitch and c_1 and c_2 are constants. An assignment that allows sufficient room for all transistors satisfies:

$$\forall_t \; |\phi(s_t)-\phi(d_t)| \geq S(t)$$

Such a perfect assignment may not always be possible. In that case a solution is preferred that leads to a small number of violations. In a vertical orientation the size of the transistor determines the height of the row. A violation has a larger influence when the size of the transistor is larger. We therefore optimize the sum of sizes of the transistors that cannot be placed horizontally:

$$\sum_{\{t \mid S(t) > |\phi(s_t)-\phi(d_t)|\}} S(t)$$

Then columns are reordered to allow more horizontal transistors. This problem can be solved exactly using a dynamic programming technique. Its complexity grows exponentially with $\max_i \#N_i$, so it is only efficient when the partitions are small. To avoid this problem, a smaller subset of partial solutions could be retained as a heuristic.

Another optimization is possible by reassigning the transistors of a subset to rows. If the pins of two transistors connect to the same signal and if they use the same layer, they are allowed to overlap. This means that the lower bound γ as defined in the previous section is not always completely valid for estimating the width of a partition of transistors. Therefore a left edge algorithm that allows such terminals to overlap, can sometimes save a row.

5. MASK GENERATION

The program uses a small library of standard transistors that have parameters for the length and width of the transistor, parameters for its pin positions, and for an optional diffusion implant. For a particular design, the parameters are substituted using a macro processor. These transistors contain most of the technology dependent information. The library can be easily designed using an interactive layout editor. A preprocessor converts the layout description to a parameterized description.

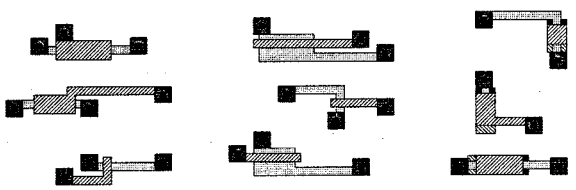

Figure 4. Some models from the nMOS library

The mask generator does a 'smart' compaction since it not only compacts, but also tries different geometries for each transistor. For each transistor a number of options are generated that differ in shape. From these options, one is picked that fits best. The library can be expanded to include more different options. This will give the mask generator more freedom to find a good solution.

The mask generator works from the bottom to the top while it places the transistors one by one. During the compaction two contours, one for polysilicon and one for diffusion, are maintained to indicate the occupied area. Pins are allowed to overlap, if they are connected to the same signal and use the same layer. Therefore the diffusion pins only occupy area in the polysilicon contour, and polysilicon pins only occupy area in the diffusion contour. A transistor is selected with the top of the contour and the area increase as criteria. Also the mirror images of each option are tried. Of course the selections of the transistors are mutually dependent, but no decisions are traced back.

6. EXPERIMENTAL RESULTS

The folding must be able to control the aspect ratio accurately. Within a large range, this aspect ratio should not influence the area of the module very much. Three layouts for a logic circuit were generated with different aspect ratios. In this experiment all terminals were assigned to the top of the layout.

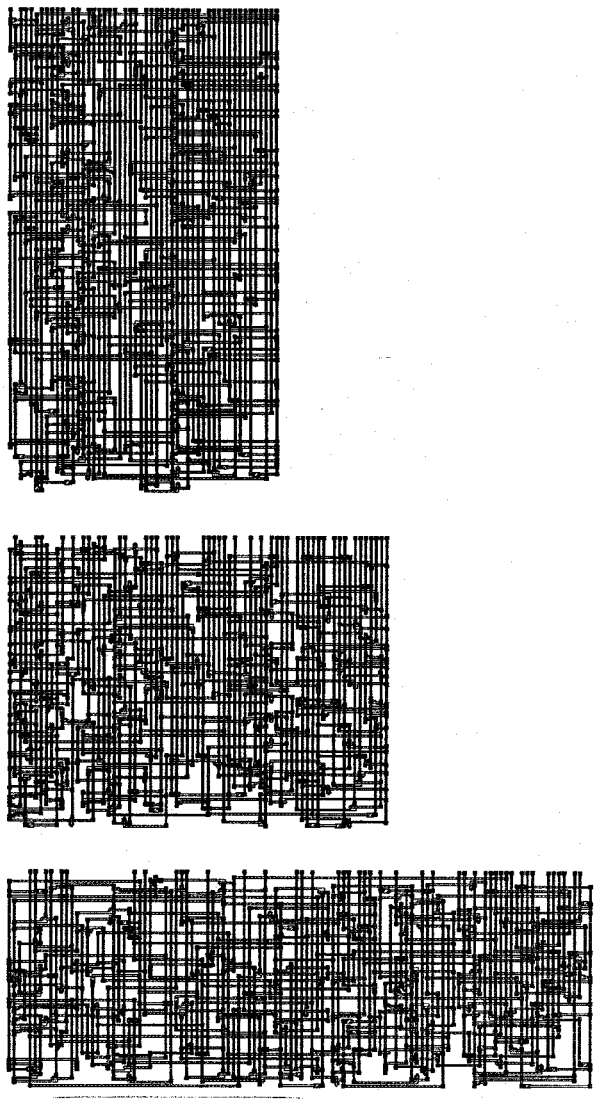

Figure 5. Three layouts for benchmark circuit 'five'

In fig.6 such results for ciruit 'cnt4' are shown in a graph. The drawn line indicates the resultant aspect ratio. It has an almost linear relationship with the desired aspect ratio. The dashed line indicates area, and remains more or less constant in the middle range. Large deformation costs extra, but may be worth while, depending on the environment of the module.

In this experiment signal nets with many terminals were cut into several nets before the folding using the mincut algorithm of [4]. These nets were connected by horizontal polysilicon strips. The number of mincut partitions depended on the desired aspect ratio: a wider module needs to more cuts. This allowed us to create much wider modules.

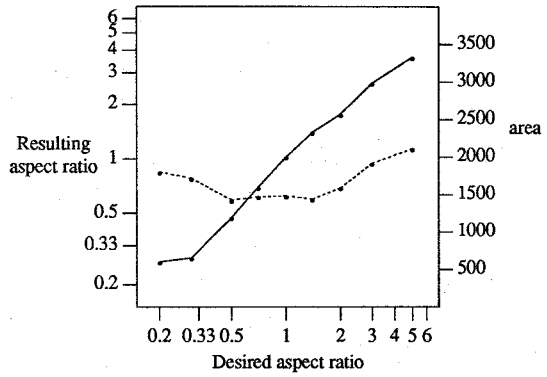

Figure 6. The aspect ratio and area versus desired aspect ratio

A number of experiments were done to compare this layout style with other automatic layout systems. All systems designed for the same nMOS process, with the same design rules. We compared the results to the results of a conventional gate matrix generator and a standard cell place and route system. The standard cells were automatically generated linear transistor arrays. Some of the net lists were extracted from layouts produced by the standard cell system. The area of the minimum enclosing box is shown in the table. Compared with the standard cell system the area was 34%-63% smaller. The comparison with a gate matrix implementation showed an 27%-56% improvement.

Inspecting the layouts shows that the new transistor matrix layouts contain no empty areas. The standard cell program suffers from empty area in channels and of unbalanced columns. Also, the positions of the transistors are not matched as precisely as in the folding. The gate matrix suffers from its grid based compaction. Both old methods exhibit wire congestion in the center of the layout that pushes layout elements apart over the entire width or length of the module. Some of these problems may be fixed, but even then it seems unlikely that the results can be matched.

comparison of layout methods				
example		area of box		
name	nr of xtors	2 dim folding	gate matrix	standard cell
hel84	12	0.043	0.072	-
data	24	0.088	0.166	-
adc	42	0.175	0.411	0.472
mp5	46	0.176	0.436	0.465
four	68	0.38	0.52	0.58
cnt4	96	0.57	1.35	-
loc	130	0.97	-	2.13
five	177	1.33	2.99	2.29
six	332	3.85	-	6.99

7. CONCLUSIONS

The transistor matrices can be parameterized with respect to all major layout parameters. It is shown that the shape and the pin positions can be controlled accurately, while the area remains more or less constant. To allow sufficient flexibility, it was necessary to split nets with many terminals into several strips. The program is interfaced to an automatic floor planning system. Using this system, the shape and pin positions can be determined automatically.

Compared to conventional methods of automatic module generation a drastic improvement in area usage is demonstrated. This improvement was partially due to improved folding which fills the enclosing rectangle denser, and partially due to the simple and efficient compaction allowed by this method.

Stepwise refinement is applied in three different ways: The generator adapts the module to a global floor plan. Secondly, the module generator uses an elegant hierarchical divide and conquer algorithm to gradually refine the two dimensional folding. Finally, the transistor layouts are adapted to the wire plan designed by the folding.

For handling CMOS circuits, the matrix could be split into several n and p regions. The objective function of the folding should use separate density functions for n and p transistors. An extra layer of metal could easily be used by the cross connection strips.

8. ACKNOWLEDGEMENTS

This work was initialized by Jos van Eijndhoven while at IBM's TJ Watson Research Center. It was supported by the EEC under Esprit project nr. 991, and by the Foundation FOM under project nr. EEL.31.0417. The authors wish to thank Paul van Teeffelen for implementing the folding algorithm, and Mark Bartholomeus and Theo Deckers for their helpful comments and discussions.

REFERENCES

[1] M. Burstein, S.J. Hong, R. Pelavin: "Hierarchical VLSI Layout: Simultaneous Placement and Wiring of Gate Arrays", *Proc. IFIP Int. Conf. on Very Large Scale Integration*, Trondheim, 16-19 Aug. 1983, F. Anceau and E.J. Aas (eds.), pp. 45-60.

[2] G. De Micheli: "Multiple Constrained Folding of Programmable Logic Arrays: Theory and Applications", *IEEE Trans. on Computer Aided Design*, Vol. CAD-2, No. 3, July 1983, pp. 151-167, errata in Vol. CAD-3, No. 3, July 1984, p. 256.

[3] S. Devadas and A.R. Newton: "Genie: A Generalized Array Optimizer for VLSI Synthesis", *Proc. 23rd Design Automation Conf.*, Las Vegas, June 29 - July 2 1986, pp. 631-637.

[4] C.M. Fiduccia and R.M. Mattheyses: "A Linear Time Heuristic for Improving Network Partitions", *Proc. 19th Design Automation Conf.*, Las Vegas, 1982, pp. 175-181.

[5] T. Kashiwabara and T. Fujisawa: "NP-Completeness of the problem of finding a minimum clique number interval graph containing a given graph as a subgraph", *Proc. 1979 Int. Symp. on Circuits and Systems*, 1979, pp. 657-660.

[6] O. Wing, S. Huang, R. Wang: "Gate Matrix Layout", *IEEE Trans. Computer Aided Design*, Vol. CAD-4, No. 3, July 1985, pp. 220-231.

[7] L.P.P.P. van Ginneken and R.H.J.M. Otten: "Stepwise Layout Refinement", *Proc. Int. Conf. on Computer Design*, Port Chester NY, Oct. 8-11 1984, pp. 30-36.

[8] N. Deo, M.S. Krishnamoorty, M.A. Langston: "Exact and Approximate Solutions for the Gate Matrix Layout Problem", *IEEE Trans. Computer Aided Design*, Vol. CAD-6, 1987, pp. 79-84.

[9] R.H.J.M. Otten: "Efficient Floorplan Optimization", *Proc. Int. Conf. on Computer Design*, Port Chester NY, Oct. 31 - Nov. 3 1983, pp. 499-503.

[10] L.P.P.P. van Ginneken, J.T.J. van Eijndhoven, P.R.M. van Teeffelen, T.J. Deckers: "Soft Macro Cell Generation by Two Dimensional Folding", *Proc. Int. Symp. on Circuits And Systems*, Helsinki, Finland, June 1988, pp. 727-730.

[11] D.F. Wong and C.L. Liu: "Array Optimization for VLSI Synthesis", *Proc. 24th Design Automation Conf.*, Miami Beach FL, June 1987, pp. 537-543.

[12] C. Lursinsap and D. Gajski: "Cell Compilation with Constraints", *Proc. 21th Design Automation Conf.*, Albuquerque, June 1984, pp. 103-108.