Transistor Placement and Interconnect Algorithms for Leaf Cell Synthesis

Martin Lefebvre (1), Chong Chan (1), and Grant Martin (2)

(1) Department of Electronics, Carleton University, Ottawa, Canada K1S 5B6, (613) 788-4452
(2) Bell-Northern Research, P.O.Box 3511, Station C, Ottawa, Canada K1Y 4H7, (613) 763-4768

ABSTRACT

Picasso is a prototype leaf cell synthesis system capable of mapping a gate level description into a complete set of layout masks, based on the line of diffusion layout style. This paper describes Picasso's algorithms for transistor placement and routing of internal nets. The transistor placement algorithm uses a composite metric which applies connectivity and optimal chaining considerations simultaneously. The routing algorithm is capable of routing over transistor chains and makes efficient use of residual routing areas resulting from unequal transistor sizes.

I. INTRODUCTION

Leaf cell synthesis is the automatic generation of layout geometries for small circuits at the leaf level of the design hierarchy. It can generally be divided into three sub-problems: (i) transistor placement, (ii) routing of internal nets, and (iii) symbolic layout compaction. Picasso is a cell synthesis system capable of mapping gate level descriptions into layout masks representations. This paper presents a general overview of the system and describes in detail Picasso's algorithms for transistor placement and routing of internal nets. Other aspects of the system are presented in [16-18].

A number of algorithms have been presented in the literature to solve the transistor placement problem. These can be separated into two groups. Algorithms from the first group address the optimization of CMOS complex gates, which are combinatorial structures comprised of series-parallel combinations of transistors [1-6]. The objective is to generate an ordering of transistors that maximizes the number of common diffusion connections between them. The second group includes algorithms that can accommodate more general circuit descriptions [7-11],[13-15]. Common objective functions include optimal chaining of transistors through common diffusion connections [7-10], as well as the reduction of internal interconnect [8-12]. The nature of the routing problem that arises at this point depends on the target layout style. In most cases, it is reduced to a channel routing problem, because there is no overlap between the routing region and the active regions [7-8],[11]. Transistors are implemented in two distinct regions, normally at the top and bottom of the cell, with internal connections in the center.

II. THE PICASSO APPROACH

Picasso maps a gate-level description into a detailed physical representation based on the line-of-diffusion layout style. The input description is a net list comprised of logic gates and transmission gates, to allow the specification of both combinatorial and sequential circuits. The first sub-task of the synthesis process is to generate a transistor level net list corresponding to the input logic description. Logic primitives of the cell description are first assigned to clusters according to their amenability to efficient layouts in the line-of-diffusion layout style. The purpose of the clustering phase is twofold. First, it identifies complex gates and other structures for which well defined procedures exist to generate efficient transistor structures. Clustering also serves to simplify the transistor placement task by introducing a placement hierarchy. Then, a transistor net list is generated for each cluster. In some cases, such as for inverters or transmission gates, this is a very straightforward one-to-one mapping. In other situations, however it constitutes a one-to-many mapping where a number of constraints have to be considered. A graph-theoretical framework has been developed, for such complex cases, and is discussed in detail in [17]. Both the direct translation of simple gates and the graph-theoretical mapping of complex gates produce transistor net lists represented as two lines of diffusion. Hence, the relative ordering of transistors within clusters is defined at this point.

A detailed transistor placement can then be generated by finding a relative placement for the clusters. This is achieved by considering both optimal chaining and interconnect considerations simultaneously, as discussed in detail in Section 3. The routing algorithm, also described in Section 3, is responsible for assigning electrical nets to physical layers as well as for determining a relative placement of metal wires on a two-dimensional virtual routing grid. The routing region is allowed to completely overlap with the active regions where devices are implemented. Although routing over transistor chains is not new, our routing algorithm makes better use of residual routing areas resulting from unequal transistor sizes. Furthermore, the system may limit the height of the cell by assigning some nets as external to the cell.

Layout synthesis is the final step of the cell generation process, where the symbolic layout representation is mapped onto mask geometries. Objects of the symbolic representation (i.e. transistors, contact windows, etc.) are assigned a vertical position and become like the wooden balls of an abacus, which are constrained in one direction but free to move from left to right. The algorithm operates from left to right by processing each object in turn. A data structure called "flexible boundary" is used to represent physical constraints between layout objects [16]. This new compaction approach makes it possible for the system to determine the position as well as the shape of layout objects, during the compaction phase. In a sense, this is equivalent to treating wires as an infinite number of "jogs", all connected in series. Although the compaction method is essentially one-dimensional, the ability to determine the shape of objects at compaction time more than compensates for this apparent limitation.

III. PLACEMENT AND ROUTING ALGORITHMS

This section describes Picasso's placement and interconnect algorithms. It also illustrates the behaviour of the system on a sample problem. The example cell is a JK flip-flop, shown in Figure 1. This cell exhibits a number of features...
that make it difficult to realize. The logic description is a mixture of random logic and transmission gates that implement the front end and latch stages respectively. There are two long feed-back nets (Q and Q-) that complicate the routing phase. Finally, there are two feed-back gates (one for each latch stage) implemented with weak transistors. To complicate things further, we will assume a constraint on the height of the cell which allows a maximum of six horizontal metal tracks for routing internal connections.

3.1. CLUSTER PLACEMENT

Transistor placement can be completed by defining a relative ordering for the clusters, by permutation and mirroring alone, without changing the transistor structures associated with each cluster. Although common diffusion connections have a general tendency to reduce the amount of metal interconnect inside a cell, optimal chaining alone is not a sufficient metric to evaluate possible placements. On the other hand, generating a placement based solely on interconnect, while ignoring the potential for chaining through common diffusions is also unacceptable. The cluster placement technique used by Picasso takes advantage of both metrics by defining an "affinity" function, which combines the two criteria.

3.1.1. The Affinity Function

The affinity $A(x,y)$ between two clusters $x$ and $y$, reflects the likelihood of the two clusters to form common diffusion connections, as well as the degree of connectivity between them. It is defined as

$$A(x,y) = A_m(x,y) W_m + A_c(x,y) W_c,$$

where $A_m(x,y)$ is the affinity function based on morphology, and $A_c(x,y)$ is the affinity function based on connectivity. The parameters $W_m$ and $W_c$ are user-defined weighting factors used to fine tune the performance of the system to specific requirements.
A data structure called the "connectivity graph" is used to represent connectivity between clusters. A cluster x is mapped onto a vertex Vx in the graph, and the weights assigned to non-directed graph edges reflect the degree of connectivity between clusters. For the purpose of computing weights, connections between clusters are first assigned a unique identifier based on the following definitions:

- Internal Connections - An internal connection corresponds to a signal that appears in the list of inputs for one cluster and the list of outputs for another, or in the output list of two different clusters.
- Type-1 Connections - A type-1 connection is an internal connection associated with a signal that is not a cell-level output.
- Type-2 Connections - A type-2 connection is an internal connection associated with a signal that appears in the list of cell outputs.
- Type-3 Connections - A type-3 connection represents cell-level inputs connected to more than one cluster.

The weight of an edge between two clusters x and y is defined as:

\[ E(x,y) = (n_1*w_1) + (n_2*w_2) + (n_3*w_3), \]

where \( n_1, n_2 \) and \( n_3 \) are the number of type-1, type-2 and type-3 connections respectively, and \( w_1, w_2, \) and \( w_3 \) are the weights assigned to each type.

Type-1 connections are normally assigned a relatively heavy weight, since they are likely to connect all three cell regions. A lighter weight is assigned to type-2 connections due to their amenability to an external implementation. Indeed, since cell outputs are very likely to occupy a track in a routing region adjacent to the cell, implementing a portion of such connections outside the cell may represent a minimal area overhead. Finally, type-3 connections are assigned the lightest weight for two reasons. The first is that common input connections are less likely to occupy the center of the cell and hence, may cover long distances in either the p-region or the n-region. The other reason is that should the metal connections not all be feasible simultaneously, connections between repeated inputs are the first to be considered for an out-of-cell implementation.

The connectivity-based component of the affinity function, \( Ac(x,y) \), is defined as:

\[ Ac(x,y) = E(x,y) + a*\{E(x,u)+E(u,y)\}, \]

where \( E(x,y) \) is the weight of the edge from Vx to Vy in the connection graph. The second member in the equation is a second order connection component that represents connections between x and y through a third cluster u.

3.1.2. Placement Algorithm

Following the above definition of cluster affinity based on morphology and connectivity, two \( n \) by \( n \) tables can be constructed representing the affinity of each pair of clusters based on either criteria. A total affinity figure is then computed for each cluster, which is the sum of all the entries in both tables for that cluster. The cluster with the highest total affinity figure is placed first, and becomes the only element of the "set of placed clusters." While clusters remain, the next cluster to be placed is selected by computing the affinity of each remaining cluster to the "set of placed clusters." The selection of a placement for a new cluster, relative to the set of placed clusters, is performed by selecting one of eight possible cases. The new cluster can be placed at either end of the chain and with either one of two possible orientations. The placed cluster at either end of the chain can also be mirrored with respect to the y-axis, if such a move produces a better placement solution. The merit of each possible placement is evaluated, again by considering the two placement metrics of morphology and connectivity.

Figure 4 shows the detailed transistor placement obtained. Following cluster placement, adjacent clusters with compatible diffusion signals can be merged, by removing a number of diffusion contacts from the symbolic representation. Also, each signal net which has not been completely implemented via a common diffusion connection or a vertical polysilicon wire should be extracted from the symbolic layout and included in a net table. Both the updated symbolic representation and net table are supplied as inputs for the routing algorithm. Both structures are shown in Figure 4.

![Figure 4. Structures provided as inputs to the routing algorithm.](image)
3.2. ROUTING OF INTERNAL NETS

The routing problem which arises at this point differs significantly from the channel routing problem. The points to be connected may appear anywhere within the routing region, and are not restricted to the edges of a routing "channel". For that reason, the routing area is divided into three distinct regions: p-region, n-region and center-region. Four different net types are defined: (i) p-type, (ii) n-type, and (iii) center-type, for nets restricted to a single routing region, and finally (iv) cross-type, for nets that cover different regions. The reason for this differentiation by net types is simple. Nets which are restricted to a single region can normally be implemented with a single horizontal wire. These nets very seldom interfere with one another. Cross-type nets, on the other hand, will inevitably require both vertical and horizontal segments, hence causing blockages for other horizontal nets.

3.2.1. Assignment of Nets to Regions

The routing algorithm attempts to reduce the complexity of the task at hand, by first decomposing it into three loosely coupled sub-problems. Complete nets, or alternatively, segments of nets, are assigned to individual routing regions. The assignment of n-type and p-type nets to routing regions is self evident. These are handled first, and simply assigned to their respective region. The philosophy for cross-type nets consists of first determining the vertical position at which they should traverse horizontal region boundaries. Cross-type nets are thus reduced to a number of smaller net segments of one of the other three types. Center-type nets are handled last. Given that such nets are most often connections between two polysilicon wires, it may be possible for some of them to be partially assigned to either the p-region or the n-region in order to avoid blockages caused by vertical segments of cross-type nets.

3.2.2. Ordering Nets within Regions

Once all the nets have been broken down into segments, each associated with a distinct routing region, a topological ordering of nets within each region is determined. Vertical constraints are generated and are used to resolve conflicts. For example, an attempt is made to assign the segments of cross-type nets that run in either the p-region or the n-region as close to the center as possible. On the other hand, p-type and n-type nets are typically positioned as far from the center as possible.

Finally, the resulting two-dimensional ordering for metal wires is represented on a virtual grid. In addition, the routing algorithm determines the horizontal positions of the source and drain connections for transistors. Polysilicon contacts must therefore be added to the symbolic representation in order to reflect the positions of metal wires within the cell.

The result of the routing phase for the JK flip-flop example, is shown in Figure 5. All the net segments present in the net table have been instantiated as a collection of two point segments on the virtual routing grid. Furthermore, following the routing operation, a number of polysilicon wires have become obsolete and hence have been removed from the symbolic layout representation.

3.3. GENERAL ALGORITHM

One of the key requirements of cell compilation systems is to always produce a solution, regardless of the design constraints. With that objective in mind, each segment of each net present in the cell description is assigned a priority figure, before the cluster placement phase begins. In the event where it is impossible to route all the metal nets inside the cell area, the cluster placement process is repeated after the lowest priority net segment has been removed from the routing table and assigned as external to the cell.

Figure 5. Detailed symbolic representation for JK flip-flop.
Because of the constraint on the height of our example cell, and the presence of a large number of metal nets in the circuit description, the first two passes of the placement algorithm have failed to produce a routable solution. Each failed attempt has resulted in a net segment being removed from the net table and assigned as external to the cell. Consequently, the final layout geometries produced by the system require an external connection for segments of both feed-back nets Q and Q'.

IV. EXPERIMENTAL RESULTS

Figure 6 illustrates the final layout produced for the complete JK flip-flop example, using generic CMOS design rules. Computation times for the JK flip-flop, measured on an Apollo 4500, were 5.6 cpu seconds for placement and routing, and 3.2 cpu minutes for the layout synthesis. Implementation of the system is in Common Lisp.

Figure 6. Complete layout generated for JK flip-flop example.

V. CONCLUDING REMARKS

Picasso presents a number of enhancements over existing approaches which range from system level concepts to performance optimization of individual synthesis steps. Some of the salient features of the placement and routing algorithms include the ability to limit the height of the cell by allocating external tracks. The formation of logic clusters allows the introduction of a placement hierarchy which reduces the search space. A composite metric is used to guide the cluster placement algorithm which applies connectivity and optimal chaining considerations simultaneously. Although routing over transistor chains is not new, the routing algorithm makes better use of residual routing areas resulting from unequal transistor sizes.

ACKNOWLEDGEMENTS

This research was sponsored in part by Bell-Northern Research of Ottawa, the Natural Sciences and Engineering Research Council of Canada, and the Canadian Microelectronics Corporation.

REFERENCES