Template Style Considerations for Sea-of-Gates Layout Generation

Glenn D. Adams and Carlo H. Séquin

Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley CA 94720

Abstract

SoGOLaR (Sea-of-Gates Optimized Layout and Routing) is a program that generates functional cells for static CMOS circuits in the Sea-of-Gates layout style. Our generator is flexible and general enough to produce efficient layout for a wide variety of Sea-of-Gates template geometries. SoGOLaR has been used to make quantitative comparisons of several different templates with respect to area utilization and cell routability. We present the results of our comparative study and draw some conclusions for the selection of a basic Sea-of-Gates template geometry.

1 Introduction

In the Sea-of-Gates layout style, chips are fabricated by adding connectivity layers to a substrate of pre-fabricated transistor templates. The layout of such chips involves the assembly of low-level functional cells by placement and routing tools. Whereas the chip-level placement and routing is mostly automatic, the layout for the library of functional cells is normally still done manually. SoGOLaR (Sea-of-Gates Optimized Layout and Routing) is a program that generates functional cells for static CMOS circuits in the Sea-of-Gates layout style. The input may be specified either as a list of boolean expressions where each expression specifies a multi-level AOI tree or as a schematic level netlist. The program uses algorithmic procedures to generate the layout of the corresponding circuit in symbolic form.

One application for this program is the regeneration of a fixed cell library in response to a change in fabrication technology. Here the role of the generator is to relieve the maintainer of the library from re-doing all the layouts by hand. Because it works at the transistor level, the program can generate layouts for a range of fabrication technologies and template styles. Regenerating the library cells automatically means that chip-level tools can make use of a more extensive library of cells than could be maintained manually.

Another application for SoGOLaR is the generation of customized cells as part of a more flexible approach to automatic layout. SoGOLaR can synthesize complex circuits directly, normally resulting in greater area efficiency than could be obtained by connecting a collection of primitive library components to generate the same function. Secondly, SoGOLaR can synthesize cells to meet layout constraints such as special aspect ratios, non-rectangular shape, or fixed I/O pin assignments. This flexibility gives chip level layout tools an added degree of freedom in generating the physical design of the chip.

Besides providing support for chip-level placement and routing systems, SoGOLaR can serve as a framework for evaluating the quality of the base array templates used in different Sea-of-Gates layout systems. Currently, there is a wide variation in the layouts of the base templates used by different ASIC chip fabricators. Furthermore, it is not easy to characterize the differences in template layouts, and one cannot predict how well a functional cell can be mapped to a particular template style without creating a layout for it. Manually laying out cell libraries for a wide variety of template styles is very tedious. Furthermore, it is insufficient to evaluate the effect of the various parameters influencing template design by looking at small cells alone. With an automatic cell generation tool, however, one can quickly implement functional cells of varying complexity on a variety of templates. Our generator, SoGOLaR, provides the capability to choose from a variety of basic template topologies and also to vary the actual template size and the relative spacing of transistors within a template. We can therefore observe the effects of these parameters on cell routing as well as on the routability of a macrocell or of a complete chip. By implementing macrocells directly from the individual transistors, we gain additional flexibility over an approach based on library cells, and we can study the effects of template design in a more global context.

In the next section of this paper we describe our layout generation tool, emphasizing those capabilities that make it useful for a template comparison study. Subsequently we state our assumptions about the chip-level environment in which SoGOLaR and the cells it generates would be used, and explain our method of template comparison. Finally, we present the results of our comparative study and draw some conclusions for the selection of a basic Sea-of-Gates template geometry.

2 The SoGOLaR Generator

One of our goals for SoGOLaR was to build a tool that was flexible and general enough to accommodate a wide range of template styles. We view the problem of transistor-level layout generation as a special type of placement and routing problem. Traditionally, the generation task is divided into distinct steps; Figure 1 lists the sub-tasks in the SoGOLaR framework.

Our overall strategy is to form a netlist of P/N-transistor pairs and then to place and route these pairs on a symbolic grid equivalent to

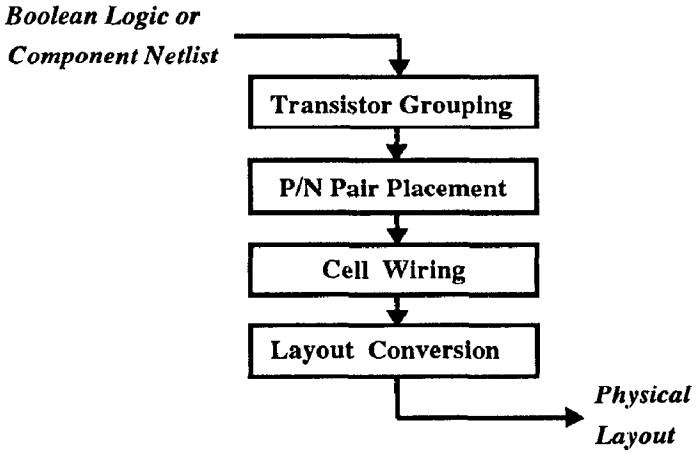

Figure 1: Tasks in the SoGOLaR cell generator

the track spacing in the first two metal layers. This strategy has the advantage that the mapping to a particular template layout is deferred as long as possible. Each of the algorithms that implement one of the above steps maintains generality and flexibility as much as possible. This allows us to handle without too much overhead the special constraints that may be imposed by a particular template style.

2.1 Transistor Grouping

The first step is to generate a netlist of transistor pairs from our input description. The exact operations that take place at this step depend upon the form of the input. If it is a network of logic functions expressed as AOI trees, we derive the netlist of transistor pairs directly from the logic function, using an algorithm similar to that found in Uehara [1]. If our input is already an explicit transistor netlist, the transistors must be paired in a different way. We first group transistors into *stages*, where a stage is found by walking diffusion paths from an output node until we reach either another output node or a power supply rail. Output nodes are any circuit node that connects to source/drain electrodes of both P- and N-types. We then pair P- and N-transistors within a stage based on shared gate connections first and subsequently based on shared diffusion connections. We first look at nets whose shared connections are unique, i.e., only one P/N pair shares that net in the stage. Then we use these pairs along with the netlist structure to derive the pairings for non-unique connections. This approach is similar to that found in SOLO [2]. We have found that these heuristics are sufficient to yield good transistor pairing in all the static CMOS circuits tested.

2.2 Transistor Pair Placement

In the second step, we must place the selected P/N-pairs onto the available sites in an array of Sea-of-Gates templates. This task differs from general block placement in that the transistor site allocation must take into account source/drain node sharing among transistors.

We start by defining an abstract rectangular array of possible P/N-pair positions, called *slots*, large enough to accommodate all P/N-pairs in the circuit and possibly some extra space for an estimated routing overhead. The number of rows and the number of slots within each row are chosen to yield a desirable aspect ratio for the cell. Placement now involves the assignment of transistor pairs to the P/N-slots in this array.

Among several other methods that we have implemented and evaluated, we generally rely on the method of simulated annealing [3] to perform this slot assignment when the quality of the final result is more important than the speed of execution as in our comparative studies of different S-o-G templates. Simulated annealing also makes it easy to combine different objectives in a single cost function such as maximizing diffusion sharing and minimizing net length. The special constraints of a particular template style can be taken into account easily by changing either the move generation process or the cost function that evaluates the result of each move. Given enough CPU time, this placement method routinely achieves the best overall results of all the placement methods tested.

This approach contrasts with other transistor layout systems such as BBC [4], which rely on hierarchical block placement to minimize net length and subsequently maximize diffusion sharing by looking for Euler paths. We have chosen a different approach because finding dual Euler paths in arbitrary transistor networks is difficult and because the phase that maximizes diffusion sharing can undo much of the work previously done to minimize the net length. In concept, our approach is most similar to SC2D [5]. However, SC2D relies on a placement algorithm that strictly partitions transistor groups into rows and then performs transistor level placement within one row at a time. We have found that in order to get highest quality results we must break the functional hierarchy and allow transistor pairs to move between rows.

The topological properties of a particular template, e.g., the number of transistors in each template, are taken into account during this slot assignment with suitable terms in the cost function. This makes the final mapping of the transistor pairs onto the template array a straightforward greedy placement of P/N pairs, performed by traversing each row left to right.

2.3 Cell Wiring

A Sea-of-Gates macrocell presents a difficult area routing problem. Individual components and the wiring between them are tightly intertwined and the use of special routing structures such as channels is inappropriate. We have developed a general area router, called CODAR [6], with the needs of cell-level layout generation in mind. It employs a *global routing algorithm* that uses an estimate of congestion in various areas of the cell, based on the combined effect of the fixed devices, previously placed nets, and other types of obstacles. A first constructive routing phase results in a rough placement of all the nets in the cell in promising but not necessarily conflict-free locations. *Insulated* wire segments are used to cross obstacles or other wires on the same routing layer.

In the subsequent refinement phase, a *detailed routing algorithm* tries to rearrange the wiring so that it can be implemented without any conflicts. It relies on a shallow recursive search using local modification moves to relocate pieces of nets that contain insulated wire segments. If not all insulated wires can be eliminated by local modification moves, the program may rip-up an entire net and call again on the global routing algorithm to find a less congested course from which the search through local modification moves is restarted.

The two algorithms, global and detailed routing, are tightly integrated and work on the same data structure representing the virtual grid of tracks. This integration has resulted in a router that can solve difficult problems not solvable by other programs while exhibiting run-times that grow only moderately with the size of the routing problem.

At the end of this phase, the complete cell layout, consisting of the template geometry as well as the paths of all custom wiring is available in a coarse-grid symbolic format suitable for translation into the OCT database [7].

3 Context for the Cell Generator

A layout generator must produce cells that can be assembled by chip-level place and route programs. These programs may impose restrictions on the overall shape of the cells, the position of terminals, the availability of feedthroughs, and the set of wiring layers that may be touched. To put our comparison studies in a proper context, we outline here three general strategies for placement and routing and note their implications for our cell generator. In all of our studies we have assumed two layers of interconnect.

In the simplest place and route strategy, our generator would build small single-row cells which are then used by a row-based placement and routing system. Here the base template needs to have just enough height, i.e., Metal-1 tracks, so that all the cells typically found in a standard cell library can be routed in Metal-1 alone. This leaves the Metal-2 plane unencumbered for the chip-level router.

If one builds bigger macros that cover several rows and which cannot be done in Metal-1 alone, we try to find a solution with a minimal set of Metal-2 connections in the vertical direction. These nets can either be wired by the cell generator or they can be passed to the chip-level routing tool as additions to the net-list to be routed. In the latter case, the router would have to implement these connections along with all the others, but would start from a totally free Metal-2 routing space. In this strategy, we would need to leave some Metal-1 tracks open for routing through the cells in order to avoid large detours for connections between macros; we call this a *porous macrocell approach.*

Finally for large, self-contained, and highly structured macros such as a datapath, the cell generator is permitted to use Metal-2 freely. The final macro is then seen as an opaque box through which the chip-level tool cannot route at will. However, the cell generator will add feedthroughs in selected areas and make them available to the chip-level tool in the form of extra terminals. The placement of these structured macros would be accomplished by a floorplanner which would specify to the macrocell generator the routing feedthroughs desired.

4 Template Comparison Method

For our comparison study, we have implemented on several template types a select spectrum of cells ranging from small examples with only 20 P/N pairs to medium sized ones with over 100 pairs. For these cells we have studied the minimum number of horizontal routing tracks and the maximum density of transistors within rows for which a particular cell could still be completely wired. To simplify our evaluation task and provide uniformity, we restrict our cell set to combinational logic, implemented as static CMOS circuits with dual networks for pull-up and pull-down. These examples were processed by the MIS logic synthesizer [8] and mapped to a representation consisting of small logic gates readily available in a standard cell library. In placing the resulting transistor pairs, we use as tight a packing as possible in order to evaluate the effects of each template style on routing congestion. In routing the cells, we use a wiring grid with uniform pitch and permit the use of both metal layers in either direction, but rely on a routing cost function that strongly discourages the use of second level metal in the horizontal direction.

4.1 Template Styles

Figure 2 shows the template styles included in our study. Our template set includes a four-transistor template developed at UC Berkeley by Lorraine Layer [9], a seven-transistor cell from Siemens that

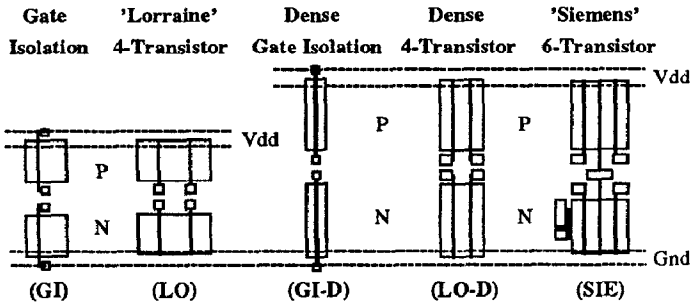

Figure 2: Template styles used in comparison

permits efficient implementation of a static RAM cell, and a gate-isolation topology with uninterrupted transistor chains of both P- and N- type. All of the template styles mirror alternating rows of templates to produce a NPPNNPPN. . . row pattern.

The template styles differ in several respects. To make our results meaningful, we taxonomize these differences. First, we distinguish between a template's *topology*, i.e., the way transistors are combined into tilable units, and its *dimensions*, i.e. its width and height expressed as the number of metal tracks in the corresponding directions. Our comparison covers three topologies: the gate-isolation or continuous diffusion strip topology, the oxide-isolated topology with four transistors per isolation cluster, and the oxide-isolated topology with six transistors per cluster[1]. Our studies indicate that the effects of template topology can be considered separately from those of template dimension.

In its original form, the Siemens cell provides nine unblocked Metal-1 tracks running across the P/N transistor pairs. The smallest version of the 'Lorraine' cell has only four such tracks, but is readily adjustable in height through the 'Mariner' [10] template generator program. Similarly there are differences in the horizontal dimensions of these templates. The three transistor pairs in the Siemens cell are at the same spatial pitch as the overlying Metal-2 tracks. The basic 'Lorraine' cell spaces the two transistor pairs by an additional Metal-2 track.

To separate the influence of the various template parameters, we added to our study versions of the four-transistor and gate-isolation topologies that use only one vertical track per transistor pair. In our routability studies, we treat template height as a parameter that may be varied independent of template topology. This allows us to search for the limit of cell routability for a given template style.

4.2 Comparison Metrics

To capture the effects of template topology, we must choose a metric that is independent of the dimensions of a template. First, we consider *site utilization,* namely the fraction of P/N transistor slots used for implementing the netlist, out of all the slots available in the templates that are fully or partially used. For gate-isolation topologies site utilization is given by:

$$SU = \frac{NLP}{NLP + I}$$

where NLP is the number of transistor pairs in the netlist and I is the number of isolation gates needed. For oxide-isolated templates, the formula is

$$SU = \frac{NLP}{n * T}$$

[1]For non-memory structures we ignore the 7th transistor in the Siemens cell

Example	#P/N Pairs	# Rows	Site Utilization GI	4T	6T
meimpb	21	1	75%	95%	70%
pmoren	30	1	68%	100%	67%
crenab	36	2	69%	100%	67%
calu	43	2	69%	98%	72%
iadr	48	2	72%	100%	70%
decode	60	2	68%	100%	67%
cbmux	77	3	69%	99%	73%
ciadr	95	3	69%	99%	67%
maskmx	105	3	67%	99%	67%
iseset	125	3	68%	99%	71%
AVERAGE	–	-	69.4%	98.9%	69.1%

Table 1: Site Utilization Results

where T is the number of template sites with at least one transistor pair used, and n is the number of transistor pairs in one site. This measure does not include leftover space due to unevenness of the length of the occupied parts of individual template rows.

The true measure of the effectiveness of a template style is not site utilization but rather *area utilization* which is expressed by the number of functionally used transistors per unit area. This measure is obviously a strong function of template size, with the smaller, denser templates having an apparent advantage. However, the templates still must be large enough to accommodate the internal wiring of the cell. Although the size and aspect ratio of a template are influenced by many considerations, in this paper we focus on the effect of template dimension on cell routability. First we evaluate the utilization of horizontal space for each of the selected template styles, then we look at the routability of our examples in order to determine minimum height requirements for each style. Lastly, we present results on the overall cell area for the template styles tested.

5 Results

5.1 Utilization Studies

5.1.1 Effects of Template Topology

In Table 1 we present site utilizations for the three topologies studied. The four-transistor template has the best utilization; the placements in this template style leave at most one template partially filled. This topology benefits from the fact that we need only find groups of two transistor pairs to completely utilize a template. The gate isolation and six-transistor topologies both use approximately 70% of the available sites for transistor pairs.

The figures shown for the gate isolation and six-transistor topologies do not represent the absolute maximum site utilization possible. A balance must be found between maximizing source/drain sharing and minimizing net length, and some potential abutments are ignored because they result in a placement with too high a net length. This effect is most pronounced in the six-transistor topology. Also, the use of simple logic functions limits the number of transistors that may be abutted to form a diffusion string. The use of larger AOI-gates for our examples would preferentially benefit the gate isolation and six-transistor topologies.

5.1.2 Effects of Transistor Density

To quantify the effect of site utilization on area efficiency, we must look at *transistor density* or the number of transistor sites per unit area. Since the height of a template can be readily varied by adding more Metal-1 tracks, we are concerned most about horizontal density. Here we group the template styles into two classes. One class of templates uses two vertical tracks per P/N pair, as exemplified by the Lorraine template and by the (loose) gate-isolation topology. In the other class, adjacent transistor pairs are packed under adjacent vertical routing tracks. This is exemplified by the Siemens six-transistor template as well as by the dense versions of the four-transistor and gate-isolation topologies. Both oxide-isolation topologies use one additional vertical track in the isolation zone between template groups. The Siemens template uses one vertical track per site for the 7th transistor and for substrate contacts. Both four-transistor templates uses a vertical track for substrate contacts for every two template sites. In the gate-isolation templates considered, substrate contacts are placed in the regions between rows and do not add vertical tracks.

Topology	Label	Horizontal Density Intrinsic	Effective
Lorraine	LO	4/11	0.36
Gate Isolation	GI	1/2	0.35
Siemens	SIE	6/11	0.38
Dense Four Tr.	LO-D	4/7	0.56
Dense Gate Iso.	GI-D	1/1	0.69

Table 2: Transistor Density Results

Table 2 shows the ratio of P/N pairs to vertical tracks for each template style. We refer to this metric as the *intrinsic horizontal density* for that template style. This value is lowest for the Lorraine template and highest for the dense gate-isolation style. We also show an *effective horizontal density* which is the intrinsic horizontal density multiplied by the average site utilization from Table 1. Because of their high site utilizations, the dense versions of the four-transistor template and gate-isolation topology exhibit the highest effective density and thus the highest area utilization of all the styles shown.

5.2 Cell Routability Study

Our method for studying cell routability consists of generating a placement for each cell in each template style and then evaluating the number of horizontal routing tracks needed to route that placement. In evaluating cell routability, we begin with a template height that is sufficient to accommodate the wiring in the cell without difficulty. We then search for the exact number of necessary horizontal tracks by routing the cell while blocking horizontal tracks. In blocking horizontal tracks we always proceed inward from the outermost tracks in each row.

Our results are presented in Table 3. In reporting our results, we have selected placements with aspect ratios slightly greater than one. The number of rows for each example is given in Table 1.

5.2.1 Template Height Requirements

The results in Table 3 indicate that for loosely packed template styles six to seven tracks are sufficient to accommodate the wiring of a small cell in a single row. Medium sized cells may also be wired in this height by using two rows for their layout. However, the increased wiring demands of larger cells soon overwhelm the capacity of minimum height templates regardless of the number of rows used in placement. While increasing template height provides greater routing capacity, the added height does not significantly increase the complexity of examples that could be routed. This occurs because the density of routing in the center of the cell increases much more quickly than at the edges, thus much of the extra space added

Template Style	Size of Example #P/N Pairs (See Table 1)									
	21	30	36	43	48	60	77	95	105	125
	Minimum # Tracks needed per row									
LO	5	6	7	6	6	7	9	10	11	12
GI	5	5	7	6	6	7	8	10	11	12
SIE	5	5	6	6	5	5	8	9	10	11
LO-D	5	8	9	8	7	8	10	12	12	15
GI-D	5	8	10	9	8	10	11	13	13	16

Table 3: Routability Results

by increasing template height cannot be used for internal wiring. This congestion affects the loosely packed and the densely packed template styles equally.

5.2.2 Effects of Horizontal Density

Table 3 also shows that the routability of a cell is a function of the horizontal density for a particular template style. In particular, the dense version of the four-transistor template (LO-D) requires one to two more routing tracks to route an example than does the Lorraine cell (LO). The dense gate isolation template (GI-D) requires three to four tracks more than its loosely packed counterpart (GI). The dense cells suffer from a lack of space for vias and wiring turns. This space is very important when routing output nets and making connections between the inputs of different transistor pairs. In template styles with two vertical tracks per P/N pair, these turns and vias may be placed to one side of the polysilicon landings. In topologies with high horizontal density, these turns must be placed either above or below the landing, thus blocking horizontal routing tracks.

5.3 Cell Area Results

In Table 4 we present the area of the maximally horizontal cells listed in Table 3 with the entries in the table sorted by template area per transistor pair. The data in this table represent *routing area*, this being the area of the cell using the minimum template height needed to complete the routing of that cell. In computing area per transistor pair, we chose a template height of eight free tracks for all styles.

From the table we see that while the dense gate-isolation pair has by far the lowest area per transistor pair, its routing area is only slightly smaller than that of the dense four-transistor cell. This occurs because the greater horizontal density of the gate isolation cell is offset by its requirement for increased template height. The loosely packed template styles all have significantly larger routing areas, indicating that these styles provide more vertical routing space than is needed by our examples.

Lastly, in Figures 3 through 5 we present the layout of a cell with 36 P/N pairs for each of the three topologies used in our study.

5.4 Global Context

The implications of these results at the chip level depend on the place and route strategy chosen. In a purely row-based design, we would need only the minimum number of tracks to make small cells routable, i.e. five to seven unblocked tracks for the Lorraine and gate-isolation template, and up to ten tracks for the dense versions of these templates. The porous macrocell approach would require some tracks beyond this minimum. Our study of cells with medium complexity has shown that the center of a macrocell becomes congested rather quickly. Extrapolating this result to the chip level indicates that extra routing rows will be needed regardless of the size of template. Given this, it seems reasonable to keep the height of the template relatively small to gain more flexibility in adding routing space. Based on our results and observations, we conclude that two to four extra tracks over the minimum would be adequate for any template style in a porous macrocell place and route strategy.

Template geometry also affects circuit performance. First of all, taller templates can provide larger transistors with larger drive capability. This can enhance circuit speed by reducing the delay for long interconnects at the cost of higher power consumption. However, this effect is mitigated by the fact that the intrinsic delay of a small net is independent of transistor size, as long as the FET gate capacitances dominate the wiring capacitances. Also, several smaller transistors may be operated in parallel to selectively reduce the delay of long interconnects. Secondly, templates with high horizontal density have a smaller diffusion area per P/N pair and hence smaller parasitic capacitance. Thus, among templates of equal area the tall dense template styles will exhibit reduced delays over the short loose styles at the cost of increased power consumption.

6 Conclusions

SoGOLaR is an effective cell generator for small to medium size cells in a Sea-of-Gates layout style. For best results it places individual P/N-transistor pairs with simulated annealing using a suitable cost function to minimize netlength and to maximize diffusion sharing. Internal cell routing is performed by CODAR, a congestion-directed router that combines global and detailed routing algorithms in an effective manner. SoGOLaR has been used to make quantitative comparisons of several different Sea-of-Gates templates with examples of cells of varying complexity.

While template types must be evaluated in the context of the overall chip layout tools used, we have reached several conclusions by comparing template styles at the macrocell level alone. All of the templates we studied are able to achieve high area utilization, with the four-transistor template showing best utilizations on an input consisting of simple logic gates. To realize this high utilization, templates must be tall enough to accommodate the internal wiring of the more complicated cells found in a library such as JK-flipflops, or counters. We found that this requires at least five horizontal tracks in Metal-1 for templates with a low horizontal density, with a few additional tracks needed for templates with high horizontal density. This additional space is used primarily for via placement.

Looking at larger macrocells, we found that adding tracks above the minimum required does not substantially increase the size of cell that can be successfully routed. This implies that it is pointless to try to provide enough free tracks within a template to handle the wiring congestion at the chip-level. Despite this, we conclude that adding two to four extra tracks per row is useful. In a porous macrocell layout strategy, the extra space helps prevent long wiring detours around macrocells. Also, we have found that the range of aspect ratios over which a cell can be routed is increased by adding extra tracks. This flexibility aids a floorplanner in optimizing the physical layout of the chip.

Template		Complexity of Example $\#P/N\,Pairs$									
Style	Area per P/N Pair	21	30	36	43	48	60	77	95	105	125
		Cell Routing Area in Track Units Square ($\lambda^2/64$)									
LO	66/2	545	825	1089	1210	1320	1815	2860	3696	4455	5544
GI	24/1	533	821	1223	1344	1424	2050	2956	4164	5091	6198
SIE	77/3	660	940	1296	1440	1584	1980	3024	4590	5184	6120
LO-D	42/2	346	630	819	924	924	1260	1960	2737	3024	4190
GI-D	15/1	290	585	825	924	910	1380	1952	2646	2862	4032

Table 4: Area Results

7 Acknowledgements

This work was supported in part by the Semiconductor Research Corporation under contract number SRC 82-11-008.

References

[1] T. Uehara and W.M. VanCleemput. Optimal layout of CMOS functional arrays. *IEEE Transactions on Computers*, C-30(5):305–312, 1981.

[2] D. G. Baltus and Jonathan Allen. SOLO: A generator of efficient layouts from optimized MOS circuit schematics. In *Proc. 25th Design Automation Conf.*, pages 445–452, 1988.

[3] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. Optimization by simulated annealing. *SCIENCE*, Vol. 220(4298):671–680, 1983.

[4] T. A. Hughes, R. Salama, and W. Liu. BBC: A module generator for back-to-back cells. In *Proc. ICCAD-86*, pages 440–443, 1987.

[5] D. D. Hill. SC2D: A broad-spectrum automatic layout system. In *Proc. IEEE 1987 CICC*, pages 729–732, 1987.

[6] P.-S. Tzeng and C. H. Séquin. Codar: A congestion-directed general area router. In *Proc. ICCAD-88*, 1988.

[7] D. Harrison, P. Moore, and A.R. Newton. Data management and graphics editing in the Berkeley Design Environment. In *Proc. ICCAD-86*, pages 24–27, 1986.

[8] R. Brayton, E. Detjens, et al. Multiple-level logic optimization system. In *Proc. ICCAD-86*, 1986.

[9] L. Layer. *Analysis of Sea-of-Gates Template and Cell Library Design Issues.* Master's thesis, University of California at Berkeley, 1987.

[10] W. A. Christopher. *Mariner: A Sea-of-Gates Layout System.* Master's thesis, University of California at Berkeley, 1989.

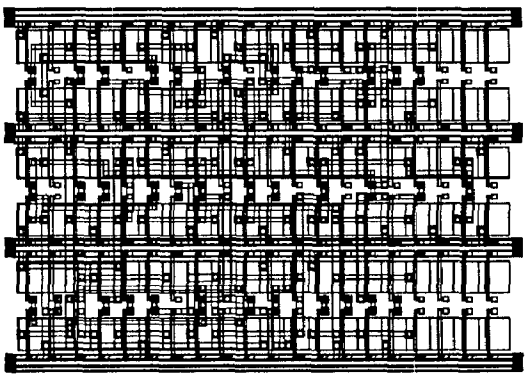

Figure 3: Example layout for gate isolation template (GI)

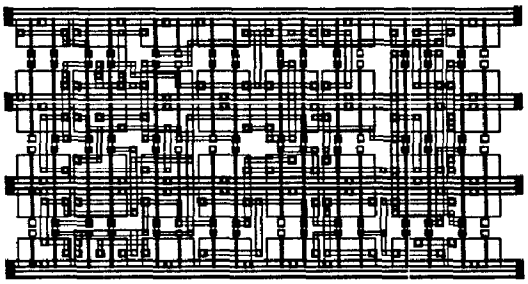

Figure 4: Example layout for four-transistor template (LO)

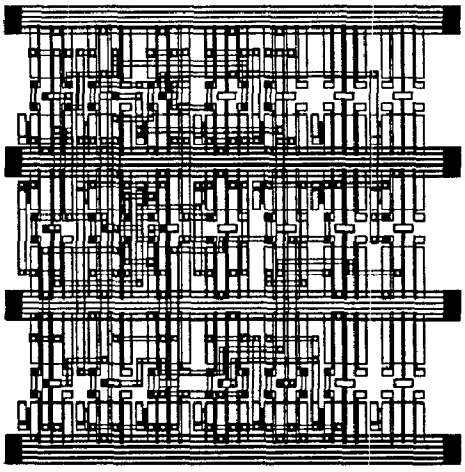

Figure 5: Example layout for six-transistor template (SIE)