Sharp-Looking Geometric Partitioning

S. Bapat and J. P. Cohoon

Computer Science Report No. TR-90-26
September 3, 1990

SHARP-LOOKING GEOMETRIC PARTITIONING

S. Bapat and J. P. Cohoon

Department of Computer Science
University of Virginia
Charlottesville, VA 22903

ABSTRACT

A new technique, named SHARP, is presented for the partitioning of VLSI integrated circuits. SHARP is a hill-climbing heuristic that is designed to be incorporated into a partitioning-based placement algorithm. Its important features include a geometric decomposition of the layout surface into a "#"-shaped region; a multi-objective function that more accurately represents wire usage than the standard min-cut criterion, and extensive use of Steiner trees. A series of experiments demonstrates that the SHARP technique produce very high quality partitions.

1. INTRODUCTION

The physical design process for VLSI circuits is often one of hierarchical decomposition. At all levels of the hierarchy, an important design step is partitioning the atomic *circuit elements* that compose the functional unit into a physical package. The physical package is realized typically as a collection of sub-packages or *modules* that are chosen such that together they optimize some predetermined figures of merit. The principal figures of merit are concerned usually with one or more of the following values: number of modules, size of modules, number of external connections required by any module, system delay [6, 7, 9, 13, 15, 17].

Circuit partitioning research has concentrated primarily on the min-cut partitioning problem which divides a circuit into two roughly equal-sized partitions in a manner that minimizes the inter-module connections. These research investigations have produced a variety of circuit element migration techniques that iteratively transform a given solution [6-9]. While these solution methods are predominantly greedy heuristics, they all use *hill-climbing* techniques to varying extents.

Min-cut partitioning methods have proven to be quite effective for their traditional application of circuit packaging, where a package can both fit a limited number of circuit elements and have a limited number of external terminals. This success led researchers to apply the method to other physical design problems--most notably VLSI circuit placement [1, 4, 5, 11, 12]. The goal of the placement step is to optimally position circuit elements onto a layout surface. The positioning of a circuit element has two basic components--one is to determine its location, and the other is to specify its orientation. An optimal assignment is typically one which allows the interconnection activity to automatically achieve its goals. These goals are often over-constrained and almost always include minimizing the total wire length and layout surface. Although placement is clearly a problem of at least two dimensions, min-cut partioning methods can produce a solution in the following manner.

- Apply the partitioning method to construct two partitions. Elements in different partitions are constrained to lie in different halves of the package.
- The algorithm is applied recursively and separately to the two partitions. The recursion terminates when no partition has more than one circuit element in it.

The technique is demonstrated graphically in Figure 1.

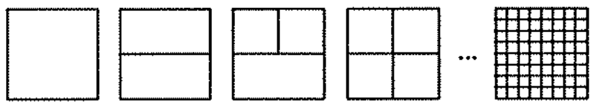

Figure 1. — Min-cut partitioning-based placement.

Placement researchers realized that the above method is too simple; to achieve acceptable solutions the sub-problems cannot be dealt with in isolation. For example, prediction techniques such as terminal propagation [5] and in-place partitioning [11] are used to help place a given circuit element by considering how its nets enter its partition block. With such techniques, min-cut-based placers (MCP) can be effective. This was demonstrated by Suaris and Kedem [20] in their comparison of an MCP using terminal propagation with the state-of-the-art, simulated annealing-based placer TIMBERWOLF [18]. For benchmark circuit Primary 1 of the ACM/IEEE Physical Design Workshop [16], the MCP produced a solution whose layout surface and wire length were within 13% of TIMBERWOLF's with a running time speed-up factor of over 100. However, Suaris and Kedem observed that the MCP's solution quality was not consistent, and that as circuit instances grew larger, solution quality deteriorated. For example, MCP's solution quality for the larger benchmark circuit Primary 2, was worse than TIMBERWOLF's by over 20%. Although additional min-cut partitioning research is increasing partitioning quality [14, 19], the fundamental problem remains that a min-cut algorithm is one-dimensional.

The one-dimensionality of min-cut partitioning led Suaris and Kedem [20, 21] to develop their *quadrisection* approach that simultaneously partitions a circuit into four quadrants, rather than the traditional two halves. By attaching non-uniform weights to horizontal, vertical, and diagonal crossings, either vertical or horizontal cuts can be favored. Thus, the quadrisection technique allows some routing congestion balancing to occur with respect to the upper and lower quadrants and to the left and right quadrants. Suaris and Kedem's experiments with quadrisection indicate that the method's solution quality is competitive with TIMBERWOLF, while running in only a tenth of the time. In spite of their generalization, some problems remain. For example, simultaneous congestion balancing is limited to quadrants in different halves, when the preference in practice is to balance cuts on opposite sides of the same halve (eg., a standard cell channel has uniform height in most design methodologies). As another example, the estimate of the routing area required by a net remains crude, as the coarseness of the partitioning allows only straight-line, single bend, and horse shoe connections (with rotations) to be considered.

To overcome these problems, we propose a new partitioning method that is more strongly influenced by the geometry of the layout surface. It is tuned for intra-package connections rather than inter-package connections. The method is named, SHARP, as the layout circuit surface is decomposed geometrically into nine regions in a manner that resembles a musical sharp. This is demonstrated graphically in Figure 2(a). In this figure, the nine partition blocks $S = \{S_1, .., S_9\}$ are canonically ordered. In Figure 2(b), the twelve interior SHARP boundary segments $C = \{C_1, .., C_{12}\}$ are labeled similarly.

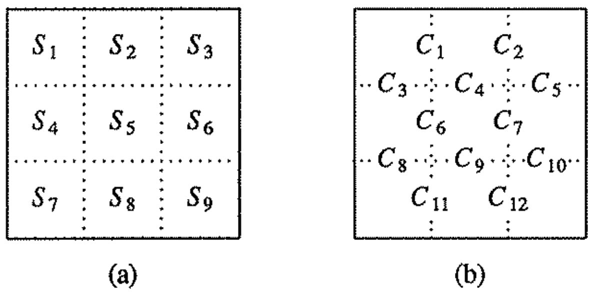

Figure 2. — Canonical SHARP labelings.

2. The SHARP-LOOKING PHILOSOPHY

The SHARP decomposition was selected as it is the smallest, nontrivial, symmetric decomposition that allows contiguous regions of the circuit surface that share similar routing features and problems (e.g. congestion) to be processed as unit. This property ensures that all its computations are readily tractable. For example, in determining the preferred interconnection given a net's block decomposition, every minimum-length Steiner tree form can be considered for the net since there are on average less than five such Steiner forms per decomposition with no net decomposition requiring the consideration of more than 192 different Steiner tree forms. Similarly, in determining favorable alternative decompositions for a net after moving one or more of its circuit elements from one block to another, there are on average only two new Steiner tree forms that need be considered. Also, since the total number of minimum-length Steiner tree forms is less than three thousand, these forms can be precomputed once and used via a hashing or an appropriate indexing scheme.

The trees given in Figure 3 are the six possible minimum length Steiner tree forms corresponding to the given block distribution of terminals.

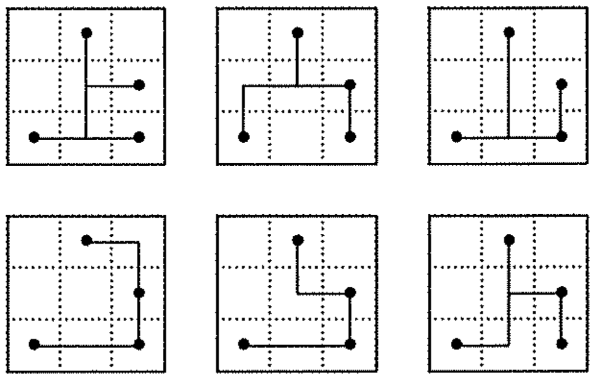

Figure 3. — Net block decomposition with six different Steiner tree forms.

As most partitioning algorithms are one-dimensional in nature, their optimization function consists of a single criterion, and as noted above partitioning-based placement algorithms use the min-cut criterion. However, the true principal figure of merit for evaluating placement quality is layout surface size. Once the circuit elements have been chosen, this reduces to minimizing the routing region. For most design methodologies, minimizing the routing region has two primary components: minimizing total wire length and minimizing channel height. Therefore, it is these two criteria that SHARP uses to evaluate partition quality. The result is a better estimate of the expected wire usage. This is demonstrated graphically with respect to wire length in Figure 4. In Figure 4(a) the bullets represent terminal locations for some net and the edges indicate a minimum length connection form for that net. This is in fact one of the connection forms that SHARP considers for the net. This contrasts to the less-desired connection in Figure 4(b) that an MCP would naturally model if the initial bisection had been horizontal. It can be demonstrated that in general a SHARP-based placer is less susceptible than an MCP to decomposition ordering.

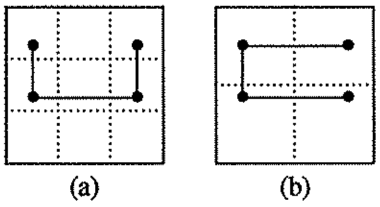

Figure 4. — Modeling wire usage.

Through use of a congestion map based on C, SHARP can better estimate channel height. Depending upon the circuit design methodology in use, SHARP can be configured to control cuts in a variety of ways and combinations. For example it cannot only control congestion by limiting the number of cuts across any one line whether horizontal or vertical, but it can also balance the number of cuts that span different lines or even parts of different lines. Thus if it is desired, SHARP can favor balancing jointly or independently the cut congestion across C_1 and C_2, C_6 and C_7, C_{11} and C_{12}, C_3 and C_8, C_4 and C_9, or C_5 and C_{10}.

Just as SHARP's optimization function is more complete than the min-cut criterion, so is the SHARP solution itself. Besides returning an assignment of circuit elements to partition blocks as a standard partitioning algorithms does, SHARP also returns a suggested Steiner tree form for each net to achieve the optimal expected use of the layout surface. This additional information makes it easier for a SHARP-based placer to incorporate a global router.

In the below section, we describe in further detail a partitioning algorithm based on the above SHARP concepts.

3. SHARP PARTITIONING

The basic SHARP algorithm is given in Figure 5. As shown there, the algorithm is a greedy one that essentially alternates between improving the two wire usage components. We found that this alternation strengthened SHARP's hill-climbing abilities.

The initial partition is constructed using a simplified clustering algorithm [2]. However, we are also considering alternative constructions using techniques such as a genetic algorithm [3]. The initial Steiner tree form is selected randomly from one of minimum length forms.

Net length minimizations are performed iteratively. During each iteration, the circuit element cluster I whose inter-block movement induces the greatest reduction in wire length is relocated to the desired block. The circuit elements in I are then *frozen* in that block for the remainder of the step. In addition, non-frozen circuit elements that share a net with a circuit element in I have their inter-block preferences updated.

```
algorithm
    compute Steiner tree forms
    construct initial partition
    for each net $u$ do
        assign to $u$ one of its minimal length Steiner trees
    end
    while partition quality is improving do
        perform net length minimizing circuit element
            movements
        perform congestion reduction through alternative
            minimum length Steiner tree selections
        perform congestion reduction through circuit
            element movements
    end
    perform congestion reduction through alternative
        Steiner tree selections
end
```

Figure 5. — Basic partitioning algorithm.

During the next two steps, the congestion map is examined to see if better balancing can be achieved. In the first of these two steps, alternative minimal length Steiner tree forms are considered for the various nets. Since no module movement is being done here and since only minimal length Steiner trees are considered, the effect on the wire usage is limited to improving congestion (i.e., there is no increase in the wire length component). As in the net length minimization step, a priority ordering is established--nets are examined in an ordering based on the amount of possible congestion improvement.

The second congestion reducing step uses circuit element movement to improve solution quality. As in the net length minimization step, the circuit elements are examined in priority order, and are frozen for the step once they have been moved. However, in this step the circuit elements are ranked with respect to possible congestion improvement rather wire length improvement. Since circuit elements moves are being made with respect to congestion improvement, this step can increase total wire length. Similarly, the net length minimization step can increase the total congestion.

During both circuit element movement steps, it may be the case that the currently most desirable circuit element move would cause a partition block to overloaded. Such overloading is initially permitted, but the amount of overloading is reduced with each pass of the loop. As a further hill-climbing feature, SHARP can be configured to use multiple priority queues so that the best feasible circuit element move is performed. It can also be configured to the find the best feasible pair or even the best feasible chain of circuit element moves.

The final step of the algorithm also attempts to improve the congestion. Unlike the previous congestion improvement steps, SHARP does not require that the alternative Steiner tree forms be of minimal length. Although the number of such tree forms increases, the number remains practical and the computation cost is worth the increase in solution quality. For example, on average there are approximately 53 non-minimal length distinct Steiner tree forms per net block decomposition with a maximum number of 192 distinct forms per decomposition.

The running time of the partitioning algorithm is dominated by the cost of the `while` loop. Since this loop in practices iterates only several times, the expected running time of the algorithm is proportional to the cost of a single pass. While it is true that no more m movements can be made in either of the circuit element movement steps, where m is the number of circuit elements, the priority of a circuit element can change multiple times. Using analysis similar to Fiduccia and Mattheyses [6], we can demonstrate that the total number of priority queue operations is on the order of p, where p is the total number of terminal pins. Since the maximum number of minimum length Steiner trees per net block decomposition is independent of the circuit instance (i.e., a constant), the total work performed as a result of circuit element movement or alternative Steiner tree selection also remains proportional to p. Thus, the running time of a loop iteration is proportional to $p \log m$, since priority queue manipulations (eg., insertions, deletions) are readily done in logarithmic time.

4. EXPERIMENTAL RESULTS

While the results discussed in this section are definitely encouraging when compared to standard min-cut partitioning (SMC) algorithms, they must be viewed as preliminary since SHARP is still undergoing refinement with respect to its hill-climbing capabilities. As a result of this refinement, we expect to achieve significant additional improvements. It must also be noted that directly comparing a SHARP decomposition to one produced by an SMC using iterative decomposition is not strictly fair--SMCs were not designed to produce such decompositions. This SMC inability is demonstrated strikingly in our first table. However, its inability should not be excused--placements are not one-dimensional, and the tools that produce placement solutions should reflect this characteristic. Since coarse routings such as SHARP's inter-block Steiner routing, are not produced by SMCs, the comparison is also unfair to SHARP, as one of its important outputs is ignored. Lastly, we note that we expect shortly to be able to compare solutions produced by a SHARP-based placer with those produced by an MCP. Our final set of experiments and analysis will reflect this.

Table 1 compares with respect to total inter-block wire length, the performance of SHARP and a Fiduccia and Mattheyses-type SMC [6] on a random circuit instance (Random 1). Random 1 has 84 circuit elements and 153 nets and was explicitly designed to be hard with respect to decomposition [10]. The table demonstrates that SMC's wire usage for both average and worst case examples is more than twice that of SHARP's. In fact, SHARP's worst case usage is approximately half of SMC's average case usage.

Method	Random 1 Routing Length	
	Avg	Max
SMC	264	291
SHARP	103	138

Table 1. — Comparing wire usage.

Table 2 also compares the performance of SHARP and the SMC. However, this comparison is with respect to channel usage (i.e., height and imbalance). SHARP's solution quality is superior to SMC's for both components of channel usage. For all four provided statistics, SHARP's solution quality was at least 50% better than SMC's solution quality.

Method	Random 1 Channel Congestion			
	Height		Imbalance	
	Avg	Max	Avg	Max
SMC	25	45	7	25
SHARP	15	26	3	10

Table 2. — Comparing channel characteristics.

SHARP was also run on benchmark circuit Primary 1 of the ACM/IEEE Physical Design Workshop [16]. Its performance characteristics were similar to those for Random 1. It also interesting to compare SHARP's solution quality to that of randomly generated partition decompositions. While SHARP's solutions are on average 4.5 times better than randomly generated solutions with respect to total routing length and channel height, the channel imbalances are comparable and both are on the order of 5 - 15%. Although it is not surprising that the randomly generated solutions have consistent channel usage given that the number of nets is approximately 1000, it is interesting to note SHARP was able to maintain the channel distribution while reducing the total wire length by approximately 400%.

5. CURRENT RESEARCH ACTIVITY

We are currently developing a family of physical design tools that make full use of SHARP's properties. For example, we are designing both parallel and sequential placers with built-in global routers. Other SHARP research is pursuing further refinement of the multi-objective function and the development of schedules for trading off the wire usage components, as well as the amount of partition overloading.

6. SUMMARY

A new physical design technique, named SHARP, is presented for VLSI geometric partitioning. SHARP is a multi-objective, hill-climbing heuristic that is designed to be incorporated into a partitioning-based placement algorithm. Experimental analysis indicates that SHARP produces very high quality partitions that are more suitable for placement than those produced by conventional min-cut algorithms.

7. ACKNOWLEDGEMENTS

This research was supported in part by the Virginia Center for Innovative Technology through grant 5-30971 and by the Defense Advanced Projects Research Agency (DARPA) through contract N00014-89-J1699. Their support is greatly appreciated.

8. REFERENCES

1. M. A. Breuer, Min-Cut Placement, *Design Automation & Fault-Tolerant Computing 1*,4 (October 1977), 343-362.
2. J. P. Cohoon and S. C. Losen, Efficient Heuristics for the Multi-Partitioning Problem, *Proceedings of the Information Sciences and Systems Conference*, Princeton, NJ, 1986.
3. J. P. Cohoon, S. U. Hegde, W. N. Martin and D. S. Richards, Distributed Genetic Algorithms for the Floorplan Design Problem, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, to appear 1991.
4. L. I. Corrigan, A Placement Capability Based on Partitioning, *16th Design Automation Conference Proceedings*, San Diego, CA, 1979, 406-413.
5. A. E. Dunlop and B. W. Kernighan, A Procedure for Placement of Standard-Cell VLSI Circuits, *IEEE Transactions on Computer-Aided Design cad-4*,1 (January 1985), 92-98.
6. C. M. Fiduccia and R. M. Mattheyses, A Linear-Time Heuristic for Improving Network Partitions, *19th Design Automation Conference Proceedings*, Las Vegas, Nevada, 1982, 175-181.
7. B. W. Kernighan and S. Lin, An Efficient Heuristic Procedure for Partitioning Graphs, *Bell System Technical Journal 49*,2 (February 1970), 291-307.
8. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, *Science 220*,4598 (May 13, 1983), 671-680.
9. B. Krishnamurthy, An Improved Min-Cut Algorithm for Partitioning VLSI Networks, *IEEE Transactions on Computers C-33*,5 (May 1984), 438-446.
10. B. Krishnamurthy, Constructing Test Cases for Partitioning Heuristics, *IEEE Transactions on Computers*, September 1987, 1112-1114.
11. D. P. LaPotin and S. W. Director, Mason: A Global Floorplanning Approach for VLSI Design, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems cad-5*,4 (October 1986), 477-489.
12. U. Lauther, A Min-Cut Placement Algorithm for General Cell Assemblies Based on a Graph Representation, *16th Design Automation Conference Proceedings*, San Diego, CA, 1979, 1-10.
13. E. L. Lawler, K. N. Levitt and J. Turner, Module Clustering to Minimize Delay in Digital Networks, *IEEE Transactions on Computers C-18*,1 (January 1969), 47-57.
14. T. Ng, J. Oldfield and V. Pitchumani, Improvements of a Mincut Partition Algorithm, *IEEE International Conference on Computer-Aided Design Proceedings*, Santa Clara, CA, 1987, 470-473.
15. C. A. Palesko and L. A. Akers, Logic Partitioning for Minimizing Gate Arrays, *IEEE Transactions on Computer-Aided Design cad-2*,2 (April 1983), 117-121.
16. B. T. Preas and K. Roberts, *IEEE/ACM SIGDA Physical Design Workshop: Placement and Floorplanning*, Hilton Head, SC, April 1987.
17. B. T. Preas and M. J. Lorenzetti, eds., *Physical Design Automation of VLSI Systems*, Benjamin/Cummings, Menlo Park, CA, 1988.
18. C. Sechen and A. Sangiovanni-Vincentelli, The Timber-Wolf Placement and Routing Package, *IEEE Journal of Solid-State Circuits SC-20*,2 (1985), 510-522.
19. C. Sechen and D. Chen, An Improved Ojective Function for Mincut Circuit Partitioning, *IEEE International Conference on Computer-Aided Design Proceedings*, Santa Clara, CA, 1988, 502-505.
20. P. R. Suaris and G. Kedem, An Algorithm for Quadrisection and Its Application to Standard Cell Placement, *IEEE Transactions on Circuit and Systems cas-35*,3 (March 1988), 294-303.
21. P. R. Suaris and G. Kedem, A Quadrisection-Based Combined Place and Route Scheme for Standard Cells, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems cad-8*,3 (March 1989), 234-244.