Layout Driven Logic Restructuring/Decomposition

Massoud Pedram and Narasimha Bhat
Department of Electrical Engineering and Computer Science
University of California, Berkeley CA 94720

Abstract

As feature sizes decrease and chip sizes increase, the area and performance of chips become dominated by the interconnect. In spite of this trend, most existing synthesis systems relegate the interconnect optimization to physical design. Physical design is, however, too far down in the design pipeline to meet the performance specifications by itself. Therefore, it is necessary for synthesis tools to share part of this optimization. In this paper, we present techniques to integrate interconnection optimization with logic restructuring and technology decomposition phases of logic synthesis. Our approach is based on a point placement of a Boolean network which is used to guide the synthesis process by providing accurate estimates on wiring area and delay. The placement solution is incrementally updated as intermediate Boolean nodes are extracted or eliminated during the decomposition or elimination procedures. Combining these techniques with layout-driven technology mapping enables us to to produce a synthesis solution and a "companion" placement solution for a given combinational logic circuit simultaneously. Using these techniques, we are able to generate circuits with smaller area and higher performance.

1 Introduction

Excessive factorization based on common kernel extraction during the technology independent phase of logic synthesis has favored gates with high fanout count and increased path delay. Inordinate attention to minimizing the active cell area has tended toward gates with high fanin count which often increase routing congestion during the final layout and increase interconnection lengths. Attempts have been made to alleviate these problems. Abouzeid et al. [1] describe an algebraic decomposition procedure based on lexicographic expressions of Boolean functions in order to reduce gate and wiring areas. This approach aims at expressing the set of logic functions in a form which after technology mapping leads to layered structured cones with ordered input injection. Murgai et al. [2] propose a kernel extraction procedure for Table Look-Up PLD's whose objective is to minimize the number of wires created as a result of replacing a node in the network by one of its kernels and the corresponding residue. Our approach which follows [9] is different from these approaches, in that placement and logic synthesis are integrated and that wiring cost is estimated explicitly using placement information.

The major issue in decomposition is the identification of common subexpressions. Sharing of such expressions across the design reduces the complexity of the synthesized network. Brayton et al. [3] proposed the notion of kernels in algebraic expressions and showed how to use kernels to find multiple cube factors which are common to two or more expressions. Their algorithm selects a kernel which produces the biggest cost reduction in terms of the number of literals. This kernel selection policy tends to minimize the active gate area after technology mapping. Here, we present a decomposition procedure with the objective of minimizing interconnects in the synthesized network.

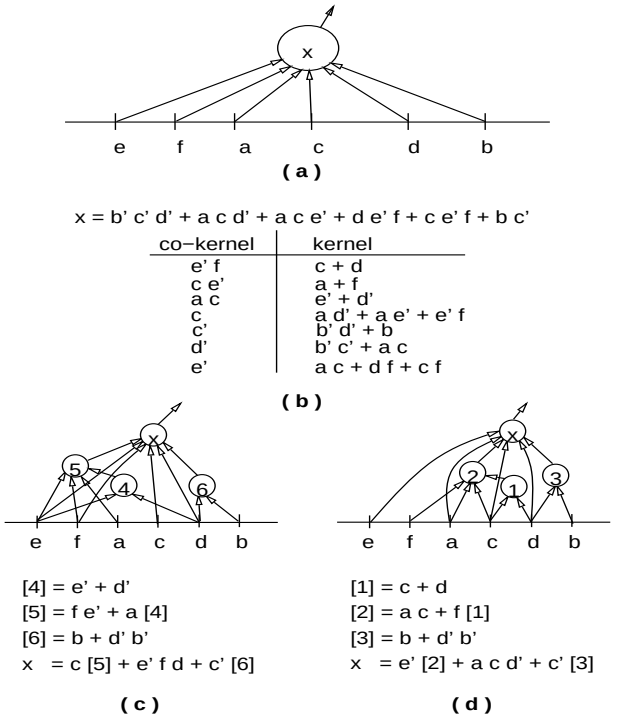

Figure 1.1: Layout-driven kernel extraction

We shall motivate incorporating the wiring estimates into the kernel selection phase with the following example. Assume that the Boolean network has been placed and that node and its fanin nodes have positions as shown in Figure 1.1a. The logic expression for node is equal to . Figure 1.1b shows all the kernels and co-kernels for this expression. The value of kernel (that is, the number of literals saved if this kernel is extracted and made it into a new node) is one. Similarly, the value of kernel is one. Figure 1.1c shows the final decomposition when is extracted while Figure 1.1d shows the final decomposition when is extracted. Both decompositions result in 16 literals (one less than the original 17 literals). However, the interconnect length for 1.1c is more than that of 1.1d. The reason is that the input signals for kernel are coming from sources on the placement plane which are placed far apart while those for kernel are coming from sources which are placed near one another. Therefore, the placement information can be used to guide the decomposition procedure in order to minimize the interconnect (often at the expense of little or no cost in terms of the literal count.)

This project is supported in part by Semiconductor Research Corporation under grant number 91-DC-008 and National Science Foundation under grant number MIP-88-3711.

2 Logic Restructuring

In this section, techniques for multiple-cube common factor extraction (which subsumes the node decomposition problem) and elimination targeted toward minimizing the total interconnection length of the synthesized network are discussed. These techniques can be combined with those targeted toward minimizing the total number of literals in the factored form representation of the network in order to simultaneously minimize the routing and active cell area.

2.1 Kernel Extraction

The extraction algorithm follows that presented in [4, 5]. We concentrate on selection and stopping criteria for layout-driven extraction. In particular, we shall describe the kernel extraction algorithm in detail since the cube extraction algorithm is simpler and follows a similar technique.

The area value of a candidate kernel is considered first. In order to find useful intersections of kernels (which correspond to common multiple-cube divisors between two or more expressions), it is beneficial to construct the co-kernel kernel-cube matrix as in [6]. A row in this matrix corresponds to a kernel (and its associated co-kernel), and a column corresponds to cubes which are present in some kernel. The entry is non-zero if kernel contains kernel-cube . The product of the co-kernel for a row and the kernel-cube for a column yields a cube of some expression. For reference, the cubes of the original expressions are numbered from 1 to . The number of the cube resulting from the product of the co-kernel for row and kernel-cube for column is placed at position in the co-kernel kernel-cube matrix. A rectangle of this matrix identifies an intersection of kernels; this kernel-intersection is a common subexpression in the network.

The area value of a rectangle – denoted by – measures the difference in the number of literals in the network if that rectangle is extracted and made into a new node. The number of literals after the rectangle is selected is given by

where is one plus the number of literals in the co-kernel for row and is the number of literals in the kernel-cube for column . The number of literals before extracting the rectangle is where is the number of literals in the cube which is covered by position of the co-kernel kernel-cube matrix. Then,

The process terminates when falls below some user-defined literal saving threshold.

In order to reduce the routing complexity, a kernel-selection policy which chooses a kernel with the greatest cost reduction in terms of the interconnection length is proposed. In particular, the interconnect value of a rectangle – denoted by – measures the difference in the total wire length in the network if that rectangle is extracted and made into a new node . In order to calculate this wire length, node must be assigned a position and positions of nodes that was extracted from must be updated. That is, positions of all nodes : must be recalculated. An exact solution requires solving a local placement problem which optimally places and nodes with respect to their current fanin and fanout nodes in the current network. This local placement problem can be solved efficiently by formulating a quadratic optimization problem with I/O pins located at the boundary of a polygon (and not necessarily a rectangle). (For example, see [7, 8].) The

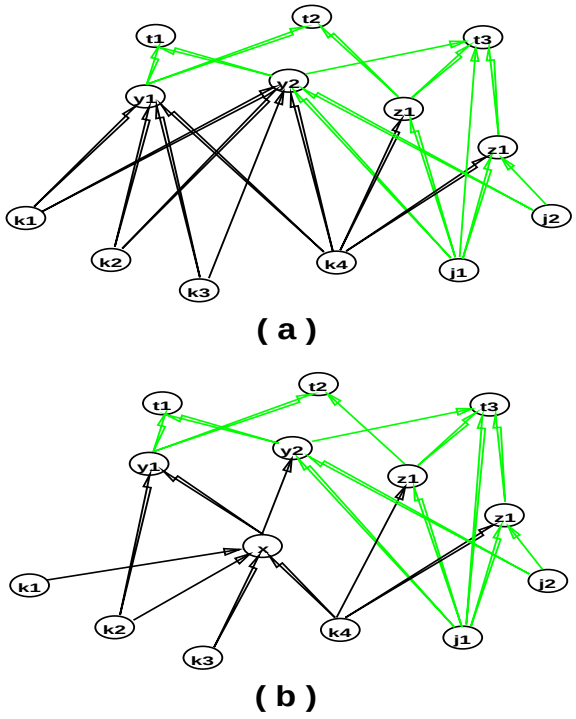

Figure 2.2: Interconnect value of an extracted kernel

exact solution, however, takes more time than is acceptable during the kernel selection phase, and therefore, an approximate solution is calculated: Positions of nodes are made fixed and is placed with respect to its fanin nodes and nodes. Since only one node needs to be placed, the placement update problem is easily solved by placing at the center-of-mass of its fanin and fanout net enclosing rectangles [9].

After assigning a position to , the new interconnection length is computed as

where ( ) denotes the wire length needed to connect node and its fanout nodes in the old (new) network before (after) extraction.

Figure 2.2 shows an example of interconnect value computation for a kernel. The dark lines in Figure 2.2a (2.2b) identify the connections which must be considered for calculating ( ).

The kernel extraction process terminates when the ratio drops below some user defined wire saving threshold. In order to optimize both literal count and wiring, the value of a kernel is a function of both and .

As new kernels are extracted, the number of nodes and the structure of the network changes. Therefore, the network and its corresponding global placement on layout plane must be updated accordingly. After a new node is created, positions of and 's are re-calculated by solving a quadratic optimization problem as described earlier.

Note that interconnect values for overlapping rectangles in the co-kernel kernel-cube matrix are implicitly handled – when kernel

is extracted, its fanin nets are updated and interconnect values of subsequent kernel extractions which overlap will be calculated based on this update.

2.2 Elimination

The elimination algorithm follows the outline of that presented in [4, 5]. Candidate vertices are selected according to some criterion and the elimination takes place if some constraints are satisfied. Elimination terminates when no candidate vertices can be found. We concentrate on the selection and acceptance criteria for layout-driven elimination.

The area value of an elimination is considered first. This value is the difference between the number of literals of the resulting network if the node is eliminated and the number of literals in the current network. This change in the total number of literals in the network (as a result of elimination) is computed by the formula given in [4, 5].

The wire length value of an elimination is defined as the difference between the total wire length in the resulting network if the node is eliminated and the wire length in the current network. It is computed in a straight-forward manner from the placement information regarding the node in question and its immediate fanins, immediate fanouts and immediate fanouts of the immediate fanins.

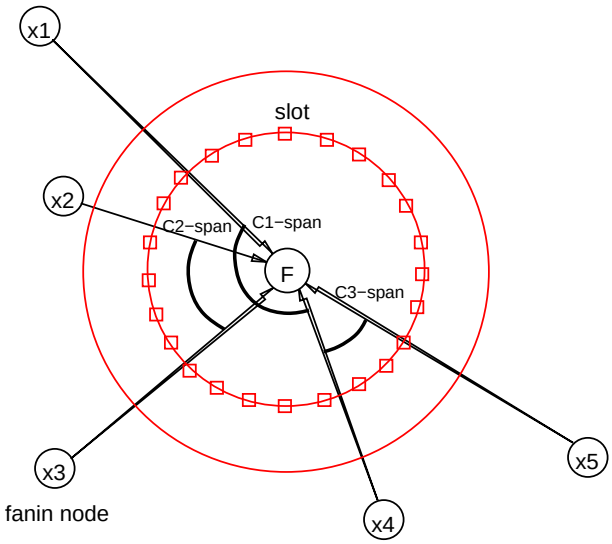

Figure 3.3: Cube ordering viewed as a linear assignment problem

3 Technology Decomposition

The procedure for converting an optimized Boolean network into the subject DAG is not unique and it is an open problem to determine which of the possible subject DAGs yields an optimum solution when an optimum covering algorithm is applied. The goal of our technology decomposition procedure is to find a circuit representation which provides a good starting point for the layout-oriented technology mapping. In particular, the logic function associated with nodes in the Boolean network are decomposed such that signals coming from nearby regions of the network enter the decomposition tree at topologically near point(s).

The decomposition process starts by constructing AND-OR trees implementing the sum-of-product form representation of the logic function associated with each intermediate node in the Boolean network. The function of AND subtrees is to compute the product terms (cubes) and that of the OR subtrees is to compute the sum of the product terms. The input signals to the AND subtrees and then the cubes in the OR subtrees are ordered. The conversion from the ordered AND-OR subtrees to the gates in base function set is straight-forward.

In order to obtain the input signal ordering, one refer to the companion placement solution for the Boolean network. Each multi-pin net signal is modeled by a star connection from the source toward the sinks. By circularly traversing around each node, a unique ordering is determined for the input signals to the node. This ordering is directly related to the positions of the fanin nodes with respect to the node in question. Next, cube ordering is achieved by setting up a linear assignment problem. slots are placed on an imaginary inner circle around the node, and the projections of the fanin signals into an imaginary outer circle around the node are found. Then, linear assignment cost matrix is set up whose entry corresponds to the cost of assigning cube to slot . This entry is equal to zero if slot falls inside the shortest circular span for the immediate support of cube . Otherwise, the cost is proportional to the angular distance of slot from the nearest end of the support span of cube . (See Figure 3.3.)

A linear assignment algorithm [10] is run on the matrix . Since rows in the cost matrix correspond to the "floating" cubes and columns correspond to the slots, the linear assignment determines a cube assignment with the minimum sum-cost. The cube ordering is easily derived from the cube positions obtained by the above linear assignment procedure. The process of ordering input signals, cubes and then primitive gate decomposition is recursively applied to all nodes in the Boolean network in order to produce the subject DAG.

4 Placement Relaxation

After logic synthesis stage, a net list of gates and a companion placement solution are available. The placement solution, however, has overlapping gates and has not yet been mapped to rows (in case of standard cell layout methodology) or to slots (in case of sea-of-gates style). The objective of global relaxation step is to reduce gate overlaps and produce even distribution of gates over the layout image. An additional goal is to make the placement solution feasible. Two basic approaches are generally used for mapping a global placement result to legal locations: (1) Perform a minimum squared error linear assignment which maps the cells in the global placement to the legal positions simultaneously; (2) Use a hierarchical bi-partitioning technique to obtain a feasible placement solution.

We have adopted the top-down bi-partitioning heuristic in the following way. The placement procedure consists of alternating and interacting global optimization and partitioning steps. In particular, for a circuit with gates, the placement procedure goes through $_2$ steps in order to produce a detailed placement. Now, assume that an initial placement solution for the circuit and two parameters and are given. These parameters specify the start and finish conditions for the relaxation procedure, that is, relaxation begins when number of modules per hierarchical region is and ends when this number is . Let $_2$, $_2$, then 0.

Our objectives are 1) to maintain structure of the initial placement solution by skipping earlier global optimization and partitioning steps and 2) to distribute gates evenly over the circuit boundary by doing global optimization and partitioning at the later steps. The placement procedure is, therefore, modified such that it goes

through steps only, thereby, achieving the relaxation goal without drastically disturbing the initial placement solution. Note that the final mapping to rows is performed when 1.

5 Experimental Results

The benchmarks were first optimized for minimum area using the *rugged script* [11]. This script produces the *area optimized* circuits. Next, the *delay script* (which does a quick decomposition, resubstitution, depth reduction, redundancy removal and full simplification) was run on the area optimized circuits to produce the *delay optimized* ones. The literal count results are given in Table 1.

Table 2 shows comparisons between LILY and MIS2.1 results in terms of active cell area, total chip area and total interconnection length (for area optimized circuits and mapping in area mode). In general, LILY's mapper tends to use smaller gates, larger active cell area (avg. 1%) but smaller total chip area (avg. 3%) and interconnection length (avg. 3%).

Table 3 shows comparisons between LILY and MIS2.1 results in terms of total chip area and longest path delay (for delay optimized circuits and mapping in timing mode). The delays are based on a 1 standard cell library. Both MIS2.1 and LILY delays are computed after detailed placement, and the wiring delays are included during the delay calculation. LILY shows an average area improvement of 27% and delay improvement of 12% compared to MIS2.1.

We used GORDIAN package for global placement, PACT pad placement program, TimberWolf 4.2 global router, and YACR detailed router during and after synthesis.

circuit	original	area opt	delay opt
C1355	1032	552	832
C1908	1497	535	841
C3540	2934	1283	1629
C432	372	219	317
C5315	4369	1763	2494
C880	703	414	558
apex6	904	732	1002
apex7	289	243	334
rot	764	664	797

Table 1: Multi-level benchmarks: number of literals in factored form (means full_simplify could not be used)

circuit	MIS2.1			LILY		
	inst. area ^{2}	chip area ^{2}	wire length	inst. area ^{2}	chip area ^{2}	wire length
C1355	0.458	1.311	136.7	0.454	1.257	129.7
C1908	0.515	1.720	192.1	0.520	1.665	188.3
C3540	1.384	5.737	694.6	1.391	5.588	672.1
C432	0.246	0.742	84.2	0.258	0.696	77.8
C5315	1.700	7.694	920.9	1.721	7.611	903.9
C880	0.452	1.458	158.8	0.453	1.357	148.3
apex6	0.728	3.194	386.7	0.737	3.010	378.5
apex7	0.258	0.720	81.1	0.265	0.817	89.1
rot	0.747	3.035	376.6	0.759	2.929	369.8

Table 2: Comparison of the instance area, final chip area and interconnection length after detailed routing

circuit	MIS2.1		*LILY*	
	chip area ^{2}	delay	chip area ^{2}	delay
C1355	6.231	18.09	4.100	15.15
C1908	7.923	32.60	6.774	24.98
C3540	21.325	40.11	13.020	36.00
C432	3.215	24.00	3.051	20.81
C5315	28.331	28.04	17.45	25.06
C880	3.226	20.67	3.158	19.68
apex6	6.160	15.91	5.818	13.60
apex7	1.376	8.27	1.324	8.50
rot	5.803	15.78	5.921	15.34

Table 3: Comparison of the final chip area and longest path delay results after detailed routing

6 Future Work

We studied the effects of interconnect on circuit area and performance, presented appropriate models and computational procedures for capturing some of these effects during logic synthesis, and most of all, introduced techniques for coupling logic synthesis to placement and for maintaining simultaneous and interactive data representations in logic and layout domains. There is still much work to be done here. Extension of layout-driven approach to the synthesis of sequential networks and Field Programmable Gate Arrays, and investigation of layout-driven techniques for logic resubstitution, simplification and redundancy removal are a few of possible research directions.

References

[1] P. Abouzeid, K. Sakouti, G. Saucier and F. Poirot, "Multilevel synthesis minimizing the routing factor," *Proc. 27th ACM/IEEE Design Automation Conf.*, pages 365-368, 1990.

[2] R. Murgai, Y. Nishizaki, N. Shenoy, R. K. Brayton and A. Sangiovanni-Vincentelli, "Logic synthesis for programmable gate arrays," *Proc. 27th ACM/IEEE Design Automation Conf.*, pages 620-625, 1990.

[3] R. K. Brayton and C. McMullen, "The decomposition and factorization of boolean expressions," *Proc. Int. Symp. Circuits and Systems*, Rome, May 1982.

[4] R. K. Brayton, "Algorithms for multilevel synthesis and optimization," G. De Micheli, A. Sangiovanni-Vincentelli and P. Antognetti, Editors, *Design Systems for VLSI Circuits: Logic Synthesis and Silicon Compilation*, Martinus Nijhoff, 1987.

[5] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli and A. Wang, "MIS: a multiple-level logic optimization system," *IEEE Trans. on Computer-Aided Design*, Vol. CAD-6, No. 6, pages 1062-1081, November 1987.

[6] R. Rudell, "Logic synthesis for VLSI design," Ph.D. dissertation, University of California, Berkeley, 1989.

[7] J. M. Kleinhans, G. Sigl, F. M. Johannes and K. J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," *IEEE Trans. on Computer-Aided Design*, Vol. 10, No. 3, pages 356-365, March 1991.

[8] A. Srinivasan, K. Chaudhary and E. S. Kuh, "RITUAL: an algorithm for performance-driven placement of cell-based ICs," *Proc. Third Physical Design Workshop*, May 1991.

[9] M. Pedram and N. Bhat, "Layout driven technology mapping," *Proc. 28th ACM/IEEE Design Automation Conf.*, pages 99-105, 1991.

[10] R.E. Burkhard and U. Derigs, "Assignment and matching problems: solution methods with Fortran programs," Springer Verlag, 1980.

[11] H. Savoj, H. -Y. Wang, "Improved scripts in MIS-II for logic minimization of combinational circuits," *Proc. Int. Workshop on Logic Synthesis*, Vol. 3, 1991.