Minimizing the Routing Cost During Logic Extraction*

Hirendu Vaishnav, Massoud Pedram

Department of EE - Systems

University of Southern California

Los Angeles, CA 90089

Abstract-*This paper describes techniques for reducing the routing cost during logic extraction. Two routing cost functions derived from the global structure of a boolean network are analyzed and the effectiveness of each cost function is compared against the conventional literal savings cost function. Experimental results obtained with these routing cost functions are presented and discussed in detail.*

I. Introduction

Recent advances in CMOS technology aiming to reduce device sizes and to put more functionality on the chip generate circuits in which routing dominates the chip area and circuit timing. Hence, addressing routing issues at all levels of design abstraction has become a necessity. Structural and functional flexibilities available at higher levels of design abstraction can indeed be exploited to greatly impact the routing cost of the circuit. However, it is very difficult to accurately estimate the routing cost at these higher levels. In this paper, we first propose new cost functions for logic synthesis that reflect the post-layout routing cost and then use them during logic extraction.

Logic extraction is the process of identifying subexpressions common to two or more functions, which are then extracted as intermediate nodes in a multi level circuit. The goal of extraction is to minimize the chip area by sharing logic across the network. Literal count has been traditionally used as the objective function during extraction as it correlates well with the gate area. However, minimizing the gate area does not always reduce the chip area. One reason for this is that extraction mechanisms based on literal minimization often produce circuits with large number of fanouts per node, increasing the routing overhead. Indeed, such increase in the routing overhead may undo any area savings due to literal minimization. Hence, it is imperative that routing cost be considered during logic extraction.

Recently, a number of researchers have addressed routing optimization during logic synthesis. Saucier et al. [11] proposed a *lexicographical* extraction mechanism to reduce the routing cost of the circuit by deriving and maintaining an order amongst primary inputs of the circuit. Pedram et al. [6] proposed a mechanism in which an incrementally updated companion placement is used to estimate the routing area and delay during logic synthesis. Vaishnav et al. [7, 12] proposed a mechanism to reduce the routing overhead by minimizing the crossing number of a circuit during technology decomposition and fanout optimization. However, the problem of identifying accurate routing measures that can be used during earlier phases of logic synthesis has been left unaddressed. In this paper, two routing measures and the corresponding routing-driven extraction procedures are described. These procedures improve the routing area by localizing the signal fanouts and by evenly distributing the signals across the network. Some important empirical observations regarding the effect of extraction of divisors with low literal savings on the routing area of the circuit are also presented.

The remainder of this paper is organized as follows. Section II gives a brief review of the algebraic extraction procedures. The proposed routing measures are described in Section III. Section IV describes the corresponding routing-driven extraction mechanisms. Experimental results are presented in Section V. Conclusions are given in section VI.

II. Background

Initial work on extraction was reported by Roth and Karp [10] and by Lawler [4]. Both these approaches apply boolean techniques for extraction. However, computational complexity of these techniques render them impractical for large circuits. Hence, recent approaches use the following procedure along with algebraic division methods to derive common divisors [2, 1, 8]. In the following, we give an overview of the algebraic extraction techniques.

First, all candidate divisors associated with nodes in the network are generated and common divisors are detected. A divisor may consist of a single cube or multiple number of cubes. Multiple cube divisors which can not be further divided by a single cube, i.e., cube-free divisors, are called kernels. The quotient of the division process is referred to as a *cokernel*. A cost value is then assigned to each divisor. This cost value has traditionally been the number of literals saved due to extraction of the divisor. A divisor that has the best cost is then selected and introduced in the network. After dividing the fanout nodes of a divisor by the selected divisor, the cost of the remaining candidate divisors might change. Hence, it is necessary to update the weight of remaining candidate divisors after each extraction.

A mechanism to identify and extract common kernels

*This research was supported in part by the NSF's Research Initiation Award under contract No. MIP-9223812.

32nd ACM/IEEE Design Automation Conference ®

(based on rectangle covering problem) was first proposed by Brayton et al. [2, 1]. Rajski et al. [8] simplified the problem by considering only cube-free double-divisors (i.e., kernels with two cubes) and single divisors with two literals. This approach improves the run time while not sacrificing the quality of the resultant circuits.

In our approach, only double-divisors and two-literal single divisors are used. Hence, the basic framework in our approach for routing driven extraction is similar to that of [8]. However, to adapt their extraction procedure to perform routing-driven extraction, two basic questions need to be answered: (1) How to estimate the routing cost of a candidate divisor during extraction? (2) How to minimize this routing cost during extraction?

III. Estimating the Routing Cost

Many circuit parameters contribute to the routing cost. These parameters can be classified in two categories: parameters which are dependent on the local structure of the boolean network; and those which are dependent on the global structure of the boolean network. Local parameters characterize the routing cost of a node or a net in the circuit whereas global parameters characterize the routing cost of the boolean network as a whole. For example, number of pins on a net is a local parameter affecting the routing cost. Clearly, a multi-pin net requires larger area to route compared to a two-pin net. In this paper, we emphasize one such local parameter, namely, the *fanout range* of a node. Fanout range of a signal is defined as the span of the fanouts of a node in terms of some geometrical (e.g., placement positions of the fanouts) or topological (e.g., logical depths of the fanouts) measure. If fanouts of a node are within a limited range, the routing length of the output net will be less than that of a net with widely distributed fanouts. The collection of fanout ranges for all the nets in the network constitutes a global parameter affecting the routing cost. In this paper, two global routing measures characterizing the routing cost based on the fanout ranges of the nodes will be presented and analyzed.

A. Fanout Ranges

An ideal scenario for minimizing the routing overhead is when the routing length of each net is minimum and the routing is equally distributed across the chip. Minimization of the routing lengths can be achieved by localizing the fanout ranges during logic synthesis. Localization implies, bringing all the nodes belonging to a net close to each other in terms of some geometrical or topological distance metric. During logic synthesis the network is still being modified, making it difficult to estimate post-placement geometric distances between nodes. A reasonable estimate of the relative positions of nodes can be obtained from a companion placement solution as in [6]. Alternatively, one can use the topological distances among various nodes. Experimental results show that geometric distances are reliable when the global interconnection structure is almost finalized, e.g., during technology dependent phase of logic synthesis. During technology independent phase, topological distances are more representative of the post-placement geometric distances.

Let us denote the set of fanouts of a node i by O_i and the depth of node i by d_i ($d_i = 0$ if i is a primary input).

Definition III.1 *The* **fanout interval** *of a node i, is given by $[b_i, e_i] = [min_{j \in O_i}\{d_j\}, max_{j \in O_i}\{d_j\}]$ where b_i and e_i correspond to the beginning and the end of the fanout interval, respectively. The* **fanout range** *R_i of node i is calculated as $R_i = e_i - b_i$.*

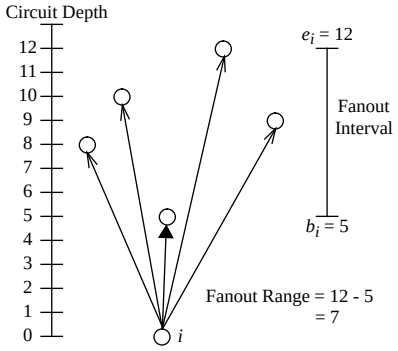


Fig. 1. Illustration of fanout ranges and fanout intervals

These definitions are illustrated in Figure 1. To confirm the correlation between fanout ranges and actual wiring cost, we calculated the wire lengths of nets with different fanout ranges after placement and routing. These circuits were area-optimized using Sis, placed using Gordian [3], and routed using TimberWolf global router [5] and Yacr2 [9] channel router. The results for three benchmark circuits are presented in Figure 2. As shown in Figure 2, netlength increases with an increase in the fanout range. The correlation is especially good for nets with low fanout range. This indicates that for larger nets, placement randomizes the effect of fanout ranges on the netlength. However, as shown in Figure 3, about 95% of nets have fanout range of 10 or less. Moreover, these nets contribute about 80% of the total netlength. Hence, it is appropriate to use fanout ranges to capture the relative routing costs during logic synthesis.

B. Fanout Range Based Routing Costs

Let $R(N)$ denote the routing cost of a boolean network N. The concept of fanout ranges can be used to estimate

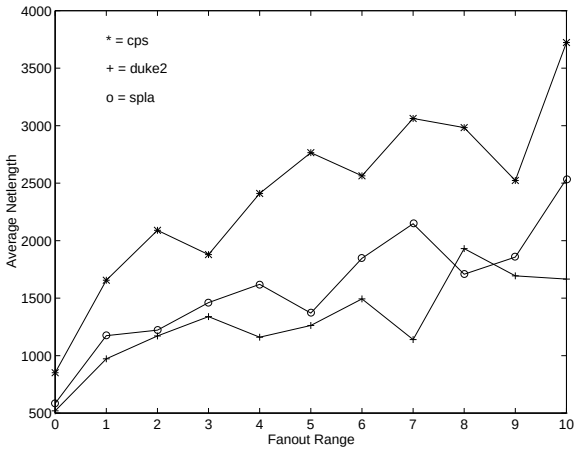


Fig. 2. Average netlengths for different fanout ranges for three benchmark circuits.

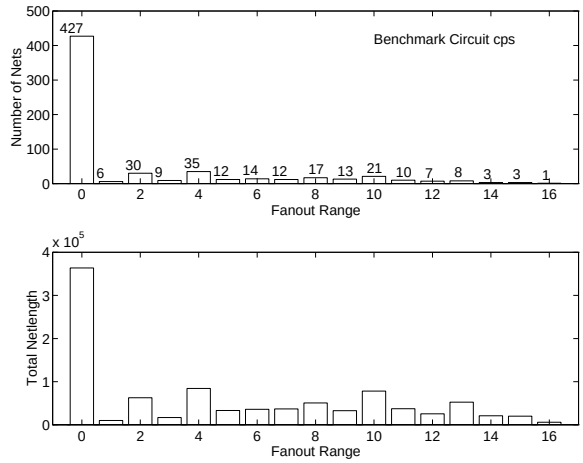


Fig. 3. Number of nets with different fanout ranges and sum of their netlengths in circuit *cps*.

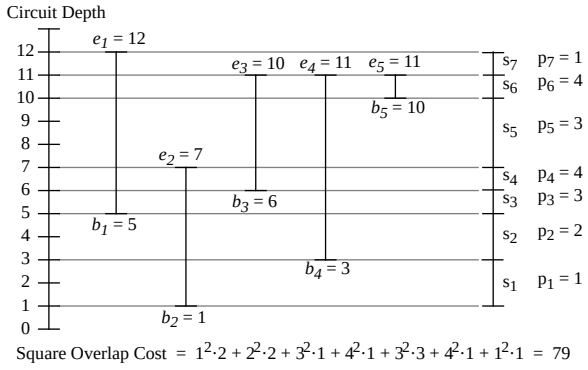


Fig. 4. Illustration of the square overlap function

this routing cost as:

$$R(N) = \sum_{i \in N} \{max_{j \in O_i}\{d_j\} - min_{j \in O_i}\{d_j\}\} \qquad (1)$$

In this case, we are adding up the fanout ranges of all the nodes in the network to determine the routing cost of the network. However, this objective function does not attempt to distribute the fanout ranges across the circuit. Uniform distribution of the signals across the chip tends to reduce congestion in the circuit. Apart from creating routing difficulties, congestion results in dead area in other parts of the chip, increasing the total chip area significantly. We propose a modified objective function that achieves minimization as well as distribution of fanout ranges simultaneously.

Suppose the fanout interval $[b_i, e_i]$ for each node $i \in N$ is known. Let M_N denote the sorted set-union of all b_i's and e_i's, and S_N denote the set of segments formed between two adjacent elements in M_N. Furthermore, let p_r denote the number of fanout intervals that overlap with segment $s_r \in S_N$, and l_r denote the length of segment s_r (see Figure 4). Since reducing the fanout interval overlap tends to uniformly distribute the signals across the network, we define the following cost function:

$$R(N) = \sum_{s_r \in S_N} F(p_r, l_r) = \sum_{s_r \in S_N} {p_r}^2 l_r \qquad (2)$$

This function reduces the fanout range overlap by minimizing the sum of the square of overlaps. Note that $F(p_r, l_r)$ can be any other function that discourages interval overlaps[1].

Assuming that all other fanout intervals remain the same, reducing the fanout range of one node will result in a monotonic decrease in the value of the above function. Likewise, keeping all other fanout intervals fixed, if the fanout interval of one node is gradually shifted in such a way that the fanout range overlap is reduced (i.e., signals are more distributed) the value of the above cost function decreases monotonically. Hence, minimization of above cost function will result in signal distribution as well as signal localization.

IV. Routing Driven Extraction

For both routing costs described in equations (1) and (2), we implemented a corresponding routing-driven extraction procedure. The greedy scheme of conventional extraction algorithm was used with all divisors restricted to be either double divisors, or single divisors with two literals as in [8]. From the list of candidate divisors, conventional approaches choose a divisor which provided the best literal savings while we choose a "good" divisor that optimizes the routing cost. A "good" divisor is selected from the list of candidate divisors based on their literal savings potential to ensure that the reduction of routing area is not achieved at the cost of a substantial increase in the active area.

The basic strategy to select "good" divisors from the list of candidate is as follows: if maximum literal savings from any divisor is M, select all divisors with literal savings within p% of M. In our experiments, the divisor was selected only if the literal savings for that divisor was within 10% of the maximum literal savings.

A. Updating the Fanout Ranges

Calculating the routing cost based on equations (1) and (2) requires a recalculation of fanout ranges of the nodes in the network for each candidate divisor.

Lemma IV.1 *If the depth of a node changes after extraction, it will increase by one.*

Proof Follows from the monotone speed-up property of a unit delay model and the fact that a depth increase of one for a fanin of a node can lead to a depth increase of at most one for the node. ■

Extracting a divisor D may only change the depth of the nodes in its transitive fanout cone TFO_D. Hence, fanout range of a node may change only if it fans out to a node in this TFO_D. We recursively traverse transitive fanout cone of the extracted node until we arrive at a node whose depth does not change due to the extraction. After identifying the nodes whose depth changes, i.e., increases by one, we check the immediate fanins of these nodes to update the range of these nodes. In the worst case, this might imply a complete depth update of the network and a traversal of each fanin connection resulting in the time complexity of $O(E)$ for a boolean graph $G(V, E)$. However, this range update can be done more efficiently by considering only the *depth determining edges* as explained next.

[1] In fact, defining $F(p_r, l_r) = p_r l_r$ gives us the same cost function as given by equation 1.

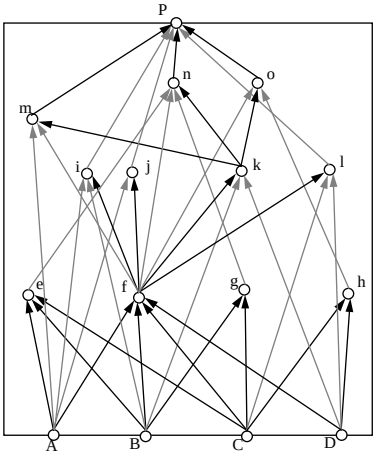
B. Improving the Efficiency of Fanout Range Update

Before the extraction, we identify all the depth determining edges (DD Edges), i.e., all edges ij such that $depth(j) = depth(i) + 1$. The concept of DD edges is shown in Figure 5. Each such edge indicates a depth dependency, i.e., if a DD edge connects output of node i to the input of node j, increasing the depth of i or extracting a divisor from j containing i will increase the depth of node j by one. We refer to such an extraction as an extraction *along a DD edge*.

Theorem IV.2 *Fanout range of a node in the network may change only if a candidate divisor is extracted along a depth determining edge.*

Proof If a divisor is not extracted along a DD edge, depth of none of the fanout nodes of the divisor will change. Hence, fanout range of all other nodes in the network will remain unchanged. ■

Hence, if a divisor is not extracted along a DD edge, the routing costs based on equations (1) and (2) remain unchanged except for the new fanout range of the extracted node. In this case, equation (1) and equation (2) can be computed in $O(1)$ and $O(maxDepth)$, respectively where $maxDepth$ is the depth of the resultant circuit.



Divisors	Fanouts	DD Signatures
A + B	e, f	{e, f}
AC + B	e, f	{e, f}
C + D	f	{f}
BC	f	{f}
B + C	f, g	{f, g}
B + D	k	--
CD	h, l	{h}
B + e	i, k	{k}
e + fg	n	--
fh	o	--
ij + l	P	--
mn	P	{P}

DD Signature Set
{{e,f}, {f, g} , {h}, {k}, {f}, {P}}

Fig. 5. Illustration of the DD edges and DD Signature Set. Solid lines correspond to DD edges

Apart from allowing a more efficient network traversal for depth update, the concept of DD edge can be used to reduce the number of updates required for each set of "good" divisors as explained next. We define as *depth dependent signature* (DD signature) the set of fanouts along DD edges of the divisor. For example, in Figure 5 divisor CD fans out to nodes h and l. However, since only depth of h will change due to the extraction of CD, the DD signature of CD is $\{h\}$. For each such divisor, the nodes whose depths changed due to the extraction are identified by identifying all the nodes which are reachable on DD edges from the each node belonging to the DD signature of that divisor.

From the set of candidate divisors, *all* distinct DD signatures are identified as *depth dependent signature set* (DD signature set). The size of the DD signature set corresponds to the number of routing cost calculations needed to identify the best divisor minimizing the routing cost. It should be noted that further minimization of the DD signature set can be obtained by reducing a DD signature containing all depth dependent fanouts of a node i into node i itself. For example, in Figure 5 signature $\{P\}$ can be reduced to $\{m\}$ or $\{n\}$. However, in this case it does not lead to further minimization of the DD signature set. In any case, the size of DD signature set is upper bounded by the number of candidate divisors under consideration. Thus, the concept of DD signature set is used to reduce the number of times $R(N)$ needs to be computed.

V. Experimental Results

Before presenting results of our extraction mechanisms, we produce in Table I, the results obtained by optimizing circuits in Sis using the rugged script (which uses "fx" to perform extraction). All circuits including the recommended set of two level examples were mapped using the Sis mapper with Lib2 gate library in area mode, placed using Gordian [3] and routed using TimberWolf global router [5] and Yacr2 detailed router [9]. Since our fanout range based cost functions are likely to be more effective when the layout of the circuit is directional, all the input pads were placed along the bottom edge of the chip and all the output pads were placed along the top edge of the chip. These results are used as a basis for comparison with our experiments.

Table I. Results after routing circuits optimized using "script.rugged"

Ckt	Route Area	Gate Area	Chip Area	Delay
b12	160	96	256	14.41
cordic	159	108	267	17.17
cps	6447	1316	7762	88.12
duke2	1496	481	1977	38.79
ex1010	24067	2622	26689	523.40
ex4	1188	516	1704	17.09
misex2	222	125	347	13.41
misex3c	1471	508	1979	75.92
pdc	2018	643	2661	37.12
rd84	218	152	370	23.91
spla	2362	700	3062	47.07
9sym	502	234	736	26.02
alu4	1936	606	2542	54.43
apex2	874	362	1236	27.23
apex3	14019	1722	15741	123.84
apex4	22252	2525	24777	599.54
rd73	137	88	225	23.46
table3	4855	993	5848	202.90

In all of our experiments, "fx" in the rugged script was substituted by the corresponding extraction procedure. The rest of the rugged script was kept intact.

A. Low Weight Divisors and Single Divisors

During our experiments, we observed that extraction of low literal savings divisors, while improving the literal count of the circuit, usually led to worse routing and chip area. The same was also true for single divisors. This is due to the fact that literal savings per occurrence of a single divisor is limited by the size of the single divisor.

By not extracting these divisors, for large circuits, although the literal count and the active cell area increased, the chip area decreased substantially. This is due to the fact that the existing placement/routing algorithms can handle clusters of nodes that are weakly connected much better than those which are strongly connected. For small circuits, however, if the additional literal savings due to extraction of these low weight divisors is significant, then the savings in gate area compensates for the increase in routing area. If the additional literal savings due to extraction of such low weight divisors is small compared to overall literal savings, increased routing overhead dominates the potential reduction in gate area and hence, such divisors should not be extracted.

Hence, in our experiments, we did not extract single divisors and any divisor with less than two literal savings for circuits with 2000 literals or more. These circuits are *ex1010, apex3 and apex4*. For the remaining circuits, low weight divisors and single divisors were extracted as long as they accounted for 10% or more of the total literal savings.

We implemented extraction mechanisms based on the measures given in equations (1) and (2). Equation (1) corresponds to the sum of ranges for nodes in the network, and hence, it is referred to as *range based extraction*. Likewise, extraction based on equation (2) is referred to as *overlap based extraction*.

B. Range Based Extraction

Using equation (1), a divisor which results in minimum increase (or maximum decrease) in the sum of fanout ranges of the nodes in the network is selected. These nodes are identified by the traversal technique described in subsection A. In the greedy extraction procedure, this implies that at every step, the increase in the sum of fanout ranges is minimized.

The results of this approach are given in table II. Each entry in the table is normalized with respect to the corresponding entry in table I. On average, we obtained 14% improvement in routing and 12% improvement in chip area. The active gate area of the circuits and the circuit delay remained about the same. The improvement on the recommended set of circuits is about 10% in route area and 9% in chip area with no change in the delay and gate area. It should be noted that circuits with literal count of 2000 or more (*ex1010, apex3*, and *apex4*), the routing area improved by 52% with a chip area improvement of 49% in spite of an increase of 32% in active area. The delay degraded by 4% for these circuits.

C. Overlap Based Extraction

Overlap based extraction minimizes the fanout range overlap function given by equation (2). The results are given in table III. Each entry in the table is normalized with respect to the corresponding entry in table I. On average, we obtained 13% improvement in routing and

Table II. Range based extraction results

Ckt	Route Area	Gate Area	Chip Area	Delay
b12	0.88	0.99	0.92	0.98
cordic	0.72	0.73	0.72	0.84
cps	0.96	1.00	0.97	0.89
duke2	0.84	0.99	0.88	0.94
ex1010	0.43	1.32	0.51	1.01
ex4	1.02	0.99	1.01	1.03
misex2	0.97	1.00	0.98	1.07
misex3c	0.86	0.97	0.89	1.07
pdc	0.98	1.03	0.99	1.01
rd84	1.05	0.98	1.02	1.11
spla	1.20	1.04	1.16	1.00
9sym	0.76	0.79	0.77	1.13
alu4	0.96	0.99	0.96	1.01
apex2	0.98	1.01	0.99	1.14
apex3	0.60	1.19	0.67	1.33
apex4	0.42	1.17	0.49	0.79
rd73	0.92	1.05	0.97	0.76
table3	0.88	1.00	0.90	1.06
AVERAGE	0.86	1.01	0.88	1.01

Table III. Overlap based extraction results

Ckt	Route Area	Gate Area	Chip Area	Delay
b12	0.93	0.99	0.95	0.98
cordic	0.72	0.73	0.72	0.84
cps	1.06	1.00	1.05	0.93
duke2	0.94	0.99	0.95	0.97
ex1010	0.43	1.32	0.52	1.01
ex4	1.00	0.99	1.00	1.02
misex2	0.97	1.00	0.98	1.07
misex3c	0.88	0.97	0.90	1.07
pdc	0.98	1.04	1.00	0.99
rd84	0.98	0.94	0.96	1.08
spla	1.18	1.04	1.14	0.99
9sym	0.79	0.79	0.79	1.07
alu4	0.95	0.99	0.96	1.01
apex2	0.96	1.01	0.97	1.13
apex3	0.63	1.19	0.69	1.32
apex4	0.42	1.17	0.49	0.79
rd73	0.91	0.98	0.94	0.91
table3	0.99	1.00	0.99	1.10
AVERAGE	0.87	1.01	0.89	1.02

11% improvement in chip area. The active gate area and the circuit delay remained about the same. The improvement on the recommended set of circuits is about 9% in route area and 8% in chip area for no change in the delay and gate area. For circuits with literal count of 2000 or more, the routing area improved by 51% with chip area improvement of 48% in spite of an increase of 32% in active area. Delay increased by 4%.

Thus, both routing measures result in similar improvements in the chip area and routing area. However, on average, the delay has remained the same. This is due to the fact that we are not directly trying to minimize the delay along the longest path, and hence, our algorithm has a random effect on the delay.

As expected from the increased time complexity, the runtimes of both these routing-driven extraction were an order of magnitude slower than the "fast extract".

VI. Discussion

In this paper, we proposed routing costs that can be minimized during logic synthesis. By minimizing these routing costs, we can improve routing area and hence post-layout chip area, performance, and power consumption. Results of extraction procedures based on these routing costs were reported. The results indicate that these measures are effective in reducing the routing cost. However, we also observed that overall chip area is very sensitive to the active area, specifically, for smaller circuits. Hence, for smaller circuits, a deviation from literal savings cost function, if not controlled properly, may result in substantial increase in active area and hence, an increase in the overall chip area.

The depth based routing costs are intuitive in nature. However, these routing costs are difficult to control during extraction. Since the extraction procedure is greedy, choosing a divisor based on current depths of the nodes, which are going to change as a result of subsequent divisor extraction, might not minimize the routing cost function effectively. Another approach is to control order of node extraction such that parameters (e.g., logical depth) based on which the divisor was chosen remain valid through extraction. However, we suspect that this will restrict the extraction mechanism significantly, resulting in a substantial increase in active area. Moreover, using a "fast extract" as opposed to "general kernel" extraction could affect the routing negatively as "fast extract" often results in a large number of small divisors with a large number of fanouts and an increased circuit depth. It is likely that the results would improve even further if a better mechanism to minimize these routing cost were applied and if the extraction was not restricted to double divisors and single divisors with two literals. We also believe that these measures can be applied even more successfully to subsequent steps of logic synthesis where the depth of the nodes can be controlled more effectively.

Our future work will focus on developing other routing cost functions for logic synthesis and on experimental evaluation and and comparison of various functions. In particular, we will focus on characterizing circuits for which extraction based on some routing cost functions is very effective. Also, an increase in the active gate area could lead to an increase in the number of rows required to maintain unit aspect ratio. For large circuits, this often led to significant improvements in routing area. A characterization of the exact effect of number of rows on the routing area is thus needed. We intend to explore this topic further.

A final note is that our routing-driven approach does not improve the results for some circuits and is sensitive to the placement and routing tools used. A future area of research will focus on precise characterization of circuits for which the proposed approach is highly effective and those for which it is not effective. This question must really be addressed in the context of the layout tools used.

Acknowledgements

The first author would like to thank Juergen Kasper for his help with the initial implementation of the code and for stimulating discussions.

References

[1] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang. Multi-level logic optimization and the rectangular covering problem. In *Proceedings of the IEEE International Conference on Computer Aided Design*, Nov. 1987.

[2] R. K. Brayton and C. McMullen. The decomposition and facorization of Boolean expressions. In *Proceedings of the International Symposium on Circuits and Systems*, pages 49–54, Rome, May 1982.

[3] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *IEEE Transactions on Computer-Aided Design*, CAD-10:356–365, March 1991.

[4] E. L. Lawler. An approach to multilevel Boolean minimization. *Journal of the Association for Computing Machinery*, 11, July 1964.

[5] K. W. Lee and C. Sechen. A new global router for row-based layout. In *Proceedings of the IEEE International Conference on Computer Aided Design*, pages 180–183, November 1988.

[6] M. Pedram and N. Bhat. Layout driven logic restructuring / decomposition. In *Proceedings of the IEEE International Conference on Computer Aided Design*, pages 134–137, November 1991.

[7] M. Pedram and H. Vaishnav. Technology decomposition using optimal alphabetic trees. In *Proceedings of the European Conf. on Design Automation*, pages 573–577, March 1993.

[8] J. Rajski and J. Vasudevamurthy. The testability-preserving concurrent decomposition and factorization of boolean expressions. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 11, pages 778–793, June 1992.

[9] J. Reed, A. Sangiovanni-Vincentelli, and M. Santamauro. A new symbolic channel router: YACR2. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 208–219, July 1985.

[10] J. Roth and R. Karp. Minimization over Boolean graphs. *IBM Journal of Research and Development*, 6(2):227–238, April 1962.

[11] G. Saucier, J. Fron, and P. Abouzeid. Lexicographical expressions of boolean functions with applications to multilevel synthesis. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 12, pages 1642–1654, November 1993.

[12] H. Vaishnav and M. Pedram. Routability-driven fanout optimization. In *Proceedings of the 30th Design Automation Conference*, pages 230–236, June 1993.