27.5

Implicit Enumeration of Structural Changes in Circuit Optimization

Victor N. Kravets Prabhakar Kudva

IBM TJ Watson Research Center
Yorktown Heights, NY 10598

{kravets,kudva}@watson.ibm.com

ABSTRACT

We describe an implicit technique for enumerating structural choices in circuit optimization. The restructuring technique relies on the symbolic statements of functional decomposition which explores behavioral equivalence of circuit signals through rewiring and resubstitution. Using rigid, yet practical, formulation a rich variety of restructuring candidates is computed symbolically and applied incrementally to produce circuit changes with predictable structural effects. The restructuring technique is used to obtain much improved delays of the already optimized circuits along with their area savings. It is also applied to analyze benefits of optimizing circuit topology at the early steps of synthesis targeting its routability.

Categories and Subject Descriptors

B.6.3 [**Logic Design**]: Design Aids -- Automatic Synthesis, Optimization; J.6 [**Computer-Aided Engineering**]: Computer-aided design (CAD).

General Terms

Algorithms, Measurements, Experimentation, Theory

Keywords

Decomposition, Technology Mapping, Physical Synthesis, Re-synthesis, Optimization

1. INTRODUCTION AND MOTIVATION

As the IC technologies scale down, their fabrication processes continue to have more pronounced downstream effects on the circuit quality. As a result, the traditional optimization techniques applied at the early stages of a design flow become less effective when targeting improved circuit timing, power dissipation, routability or yield. Similarly, the typical action of iterating through logic and layout synthesis, trying to fix problematic parts of a circuit using conventional point algorithms and quality estimates tends to loose its productivity. In this paper we limit the scope of the problem to the development of a circuit optimization technique which transforms the design for a desired structural change. The development of such technique is motivated by the increased importance of structural properties of a circuit in its final layout quality. Producing placement-friendly circuit topologies therefore becomes new dimension in circuit optimization, in addition to the traditional synthesis measures such as area and delay.

In recent years integration of logic synthesis in the inner loop of the physical domain optimization has become a viable approach when addressing advances in IC technologies. Although the attendant benefit of such approach is apparent, it is faced with new technological constraints such as spatial location of the objects, their interconnection patterns, amount of free space in their proximity, and provided cell library. Thus circuit restructuring must confront the increased emphasis on wiring and predictability of a change. Consequently transformations which re-decompose logic relying on the literal-saving techniques are not likely to satisfactory accommodate complex layout constraints, and would yield diminishing returns. Instead, the demand is placed on the optimization techniques which consider signal dependencies in a circuit layout, and implications of a given technology on circuit quality.

DAC 2004, June 7-11, 2004, San Diego, California, USA

The main contribution of this paper is an efficient way for generating a rich variety of circuit restructuring choices. Its generation is performed relying on new computational mode that implicitly generates feasible wire replacements at a gate inputs, while allowing fanin-limited logical change of its functions. The technique explores functional equivalence of a gate's output signal, and is derived symbolically from classical notions of functional decomposition [17] operating in true Boolean domain.

Rather than producing rewiring change through single-node resubstitution [2,18,19], simplification at a gate using don't cares [4,18], single-node resubstitution, or explicitly analyzing effects of wire replacements using ATPG techniques [14], the technique implicitly explores logical changes at a fanin. Unlike the traditional approaches targeting minimization of literal count, the computation first solves for a rewiring pattern, and fixes logic of its gates only at the later step. The changed logic of a gate is then mapped directly to a cell, or redecomposed in terms of the library primitives.

The predictable effects of such transformations open new improvement opportunities for the designs which fail to meet required timing or area constraints. It allows for aggressive correction of critical paths, area recovery or improved wireability. The proposed circuit restructuring technique has been developed specifically for this purpose, and we therefore evaluate it in the resynthesis-based optimization. During the optimization, the circuit restructuring driver picks a circuit gate, selectively changes its fanin signals and reimplements its new logic. We perform remapping of the changed logic following steps of the constructive approach described in [11,12].

The paper is organized as follows. We begin with the overview of related work in Section 2. Section 3 gives a formal statement of logic restructuring choices using symbolic decomposition of a function. Implementation of the circuit restructuring technique is described in Section 4. Experimental results are described in Section 5. Section 6 gives conclusions and suggestions on further improvements to the restructuring framework.

2. RELATED WORK

Several techniques that address convergence of designs were proposed relating closer technology independent and technology dependent stages of multi-level synthesis. In [1] a remapping technique is applied as a post-processing step to reduce the power and delay of critical circuit sections. In [15] algebraic division [4] is cleverly interleaved with technology mapping to permit a wider, albeit still local, exploration of the implementation space. Approaches based on SPFDs [20] were also studied to improve topology of optimized circuits. They target simplification of individual nodes based on flexibility that arise in their neighborhood.

In the past a set of powerful ATPG-based techniques was developed to explore rewiring flexibility in a design [8,9,14]. Relying on the principle of wire-addition-and-removal they explicitly examine a rewiring change, and then evaluate its effect on the network functionality to determine feasibility of a change. In theory these rewiring changes are capable of all possible network transformations [14], and their practical value was demonstrated in meeting various design optimization objectives.

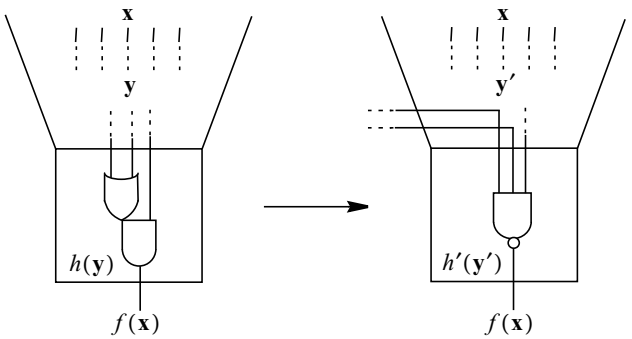


Fig. 1. Circuit restructuring through signal rewiring and resubstitution. The transformation changes h and its support without modifying function f.

More recently, a resynthesis loop based on the symbolic formulation of functional decomposition been proposed to restructure collapsed regions of a circuit [12,16]. The drawback of this approach, is that its restructuring change is localized by the boundary of a region that is selected using topological properties of a circuit, ignoring global functional behavior of a circuit. Furthermore, restructuring techniques which rely on partial collapsing (see, e.g. [21]), tend to "scramble" large portions of a circuit destroying its good parts, and are difficult to apply within the physical domain.

3. SYMBOLIC COMPUTATION OF CIRCUIT RESTRUCTURING

This section introduces a circuit restructuring technique for the optimization of combinational blocks in a design. It allows for implicit enumeration of possible rewiring and resubstitution candidates, and establishes feasibility of desired signal rearrangements in a circuit.

3.1 Restructuring choices

Given a circuit gate we would like to explore possible replacements to its input connections which could change its internal logic, but keep functional behavior of its output signal intact. Fig. 1 gives an illustration of such a transformation where input connections to gate `oa21` are changed, and the gate is replaced with the 3-input `nand`. Such resubstitution in general does not always exist, and requires logic surrounding the change satisfy certain functional conditions. We establish feasibility of a replacement relying on the functional decomposition.

Let $g_1(\mathbf{x}), \ldots, g_k(\mathbf{x})$ be functions of the rewiring candidate signals $y_1, \ldots, y_k$, and let $f(\mathbf{x})$ be a function of the output signal of the considered gate h. Since functional dependence on a signal implies its structural dependence in a circuit, we make a practical assumption that these functions are expressed in the input domain of a fanin cone of h. The functions of k rewiring candidates can be viewed as a single multi-output decomposition function $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_k(\mathbf{x}))$, and their output signals can be represented by a k-vector $\mathbf{y} = (y_1, \ldots, y_k)$. We then say that a replacement exists if there is a k-variable function $h(\mathbf{y})$ such that

$$f(\mathbf{x}) = h(\mathbf{g}(\mathbf{x})) \tag{1}$$

Solving above equation for h we are able to determine logic for h.

We illustrate our notation and show how equation (1) establishes feasibility of a replacement in the example below.

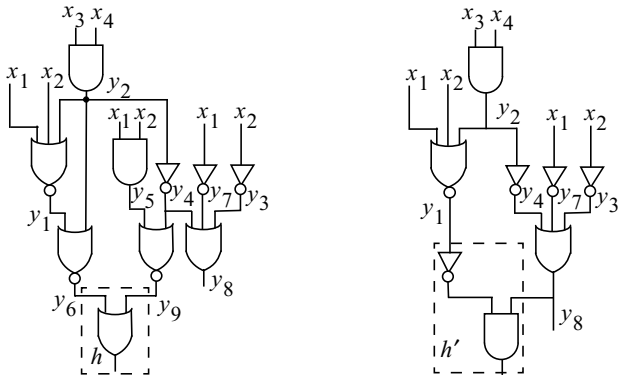
Example 3.1 Fig. 2a shows a schematic of a circuit which we optimize exploring new support to its gate h. Function of its output signal is expressed in terms of the circuit's input boundary as

$$f(\mathbf{x}) = x_1\bar{x}_2 + \bar{x}_1x_3x_4 + x_2\bar{x}_3 + x_2\bar{x}_4$$

where $\mathbf{x} = (x_1, x_2, x_3, x_4)$. Its optimized implementation in Fig. 2b is achieved by changing logic of h from $y_6 + y_9$ to $\bar{y}_1y_8$. The functional feasibility of this replacement is confirmed by establishing consistency of

$$f(\mathbf{x}) = \overline{g_1(\mathbf{x})} \cdot g_8(\mathbf{x}) = \texttt{and(inv(}g_1(\mathbf{x})\texttt{)}, g_8(\mathbf{x})\texttt{)}$$

where $g_1(\mathbf{x}) = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2\bar{x}_4$ and $g_8(\mathbf{x}) = \bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4$.



a) original circuit b) optimized circuit

Fig. 2. Circuit optimization through logic change of h from `or(`y_6, y_9`)` to `and(inv(`y_1`)`, y_8`)`. The transformation results into better overall circuit implementation.

The replacement results in removal of gates y_5, y_6 and y_9 from the original circuit. ■

3.2 Existence of a feasible replacement

Detecting if candidate signals $\mathbf{y}$ form feasible support for logic h reduces to solving equation (1). Solutions to the equation also list replacements for h. We derive its solutions using the interval notation for functional decomposition that is consistent with [12].

Functions of the selected candidate signals $\mathbf{g}$ impose constraints on h. These constraints are fully described by the consistent value assignments to the $\mathbf{x}$ and $\mathbf{y}$ signals. The characteristic function for these assignments is given below:

$$c(\mathbf{x}, \mathbf{y}) = (y_1 \equiv g_1(\mathbf{x}))\ldots(y_k \equiv g_k(\mathbf{x})) \tag{2}$$

Values of function h must agree with values of f in the on-set of c; in all other points of c function h can be selected arbitrarily. Using interval notation [5] we express this selection flexibility as

$$h(\mathbf{x}, \mathbf{y}) \in [c \cdot f, \bar{c} + f] \tag{3}$$

The above interval solution to h contains functions that depend on the signals of $\mathbf{x}$, as well as functions whose dependence on $\mathbf{x}$ is merely syntactic – in effect they are functionally independent of $\mathbf{x}$. To be consistent with the decomposition template in (1), functions that are not independent of $\mathbf{x}$ must be removed from the interval. We achieve this by applying signal abstraction to the bounds of the interval using existential and universal quantification of $\mathbf{x}$:

$$h(\mathbf{y}) \in [\exists\mathbf{x}(c \cdot f), \forall\mathbf{x}(\bar{c} + f)] \tag{4}$$

It is important that this process of abstraction does not remove any function from the interval in (3) that is functionally independent of $\mathbf{x}$.

For the set of functions represented by the interval to be non-empty its lower bound should not exceed its upper bound, i.e. it must be that

$$\exists\mathbf{x}(c \cdot f) \leq \forall\mathbf{x}(\bar{c} + f) \tag{5}$$

The consistency condition imposed on the upper and lower bounds of the interval can be stated in terms of the equation

$$\overline{\exists\mathbf{x}(c \cdot f)} + \forall\mathbf{x}(\bar{c} + f) = 1 \tag{6}$$

Example 3.2 To check whether connections y_1 and y_8 to h form feasible support we instantiate (4) with f and

$$c(\mathbf{x}, y_1, y_8) = (y_1 \equiv g_1(\mathbf{x}))\ldots(y_8 \equiv g_8(\mathbf{x}))$$

Abstracting signals $\mathbf{x}$ we obtain interval

$$h \in [\bar{y}_1y_8, \bar{y}_1y_8 + y_1\bar{y}_8]$$

The interval is not empty, implying that the reconnection is feasible. The interval also represents a set of two possible functions for h. Thus, new logic for h can be selected either as $\bar{y}_1y_8$ or as $y_1 \oplus y_8$. (The Fig. 2b circuit selects $\bar{y}_1y_8$ since it provides a favored implementation in modern technology.) ■

Note that interval notation in (3) has a simple extension for accommodating external don't cares and don't cares of forward of h logic by placing them in the off-set of c.

3.3 Enumeration of fanin-limited replacements

The overall variety of n candidate rewirings y_i could be very large, and we therefore would like to impose a fanin limit k on the support size of h. A support limit also provides us with a controlling "knob" for a desired logic change at h. Explicit enumeration process which examines all k-subsets from the given set of n signals candidates is likely to be formidable. Instead we use an implicit technique to compute feasible signal subsets. The technique relies on the abstraction mechanism which generates symbolic encoding of all solutions from a parameterized computational form (6).

The parameterization of (6) is performed by extending function c with a set of n coefficients $\mathbf{s}$ which encode possible support selections for h:

$$C(\mathbf{x}, \mathbf{y}, \mathbf{s}) = ((y_1 \equiv g_1(\mathbf{x})) + \bar{s}_1)\ldots((y_n \equiv g_n(\mathbf{x})) + \bar{s}_n) \quad (7)$$

In this parametrization an assignment to the $\mathbf{s}$ coefficients induces c in the form of (2) for a particular support. A signal y_i structurally connects to h iff $s_i = 1$.

Substituting C in (6) and universally abstracting $\mathbf{y}$ we obtain computational form for the characteristic function of feasible support subsets:

$$\chi(\mathbf{s}) = \forall\mathbf{y}(\overline{\exists\mathbf{x}(C \cdot f)} + \forall\mathbf{x}(\bar{C} + f)) \quad (8)$$

The above equation provides a computational core for the implicit generation of feasible rewirings at a given node, and can be directly implemented using binary-decision diagrams (BDDs) [6].

Subsets encoded by function χ can contain up to n signals. Depending on the desired k we eliminate unnecessary subsets constraining χ with an n-variable function whose on-set minterms have weight k (here weight is a number of positive literals in a minterm). Although such weight function represents an exponential number of minterms, it is very efficiently constructed and represented with BDDs.

4. IMPLEMENTATION OF CIRCUIT RESTRUCTURING

We implemented symbolic formulation of the presented technique to make it available as a circuit restructuring engine within a resynthesis loop. This section describes heuristics used in the engine implementation.

For a given circuit gate and a set of candidate rewiring signals the engine computes encoding of feasible supports of a gate using (8). The encoding is computed with respect to a boundary of a gate's fanin cone limited to the maximum depth of 5. The selection of candidate rewiring signals is restricted to the fanout cones of this fixed boundary. Since the set of candidate signals for a circuit gate could be large we limit its size at each instantiation of (8) to 16. Each set of selected candidate signals contains immediate inputs of a gate and a set of other signals from its restricted neighborhood. The selection procedure iteratively selects subsets of signals which meet this neighborhood until they are all examined.

The presented symbolic technique for enumerating restructuring choices entails following steps:

(a) establishing feasible input connections at a gate
(b) derivation of a function for a new logic for a gate
(c) implementing of the new function in terms of the library primitives

The restructuring step (a) relies on a pre-specified fanin bound to control the extent of a desired change: larger values imply increased resynthesis flexibility. The new gate function is then selected in step (b) from the interval given by (4) such that its BDD representation is minimized. To implement logic of the new function in step (c) we employ a constructive flow that successively decomposes the functions being synthesized directly in terms of the appropriately chosen decomposition primitives corresponding to the cells of a typical CMOS library.

5. EXPERIMENTAL EVALUATION

The experiments described in this section are conducted to examine potential benefit of the proposed restructuring technique at different stages of a design flow, and to illustrate its ability to consider various optimization objectives.

Table 1: Results of optimizing MCNC circuits using SIS- and the proposed restructuring technique.

circuit	SIS-1.2 script.delay		using restructuring optimization as post-processing			
	area	delay	area	delay	enum,sec	runtime,sec
C1355	1026	35.4	818	30.6	27	166
C1908	1191	48.8	850	40.1	71	345
C2670	1422	56.3	1210	39.9	53	242
C3540	2152	60.9	2111	51.3	432	1849
C432	320	43.9	296	40.1	17	96
C5315	4151	46.9	2958	40.3	242	1332
C6288	5690	150.2	4498	120.4	813	3498
C7552	5187	50.8	4304	40.7	582	3288
C880	783	36.2	680	28.6	92	394
Avg. improvements:			0.82	0.82		

5.1 Early timing-driven resynthesis

The large number of restructuring options provided by the technique increases the likelihood of meeting tight delay objectives while meeting area constraints. We first evaluate implementation of the restructuring technique performing resynthesis of circuits produced by SIS-1.2 [19]. The evaluation is performed on the larger circuits from the MCNC benchmark suite. We used `script.delay` to optimize circuits in SIS-1.2. It was modified to include technology mapping command `map -m 0 -AG` at the end of its technology-independent optimization. In all of the experiments the `mcnc` library was used.

We applied `script.delay` to a subset of large MCNC circuits; their names are given in the first column of Table 1. Columns two and three in the table correspond to the area and delay numbers obtained from applying the script to the circuits. To demonstrate incremental nature of the presented restructuring technique, and its ability to further improve quality of already optimized circuits we apply it to these circuits. They are post-processed invoking or restructuring technique in the delay minimization mode followed by the area optimization. Both of the calls have fanin increase constraint on the logic function change set to 1.

Results of the post-processing are given in the four right-most columns of Table 1. The first two of these columns provide data on the obtained circuit area and delay. The delay improves up to 30% percent, with the average of being 18%. The reduction in circuit delay comes with no area penalty. In fact, a significant area savings are observed in all of the examined circuits (18% on the average). The last two columns give the time spent in computing symbolic encoding of the restructuring choices, and the total runtime.

5.2 Physical domain optimization

The incremental nature of the restructuring technique also localizes circuit changes, avoiding needless perturbation of non-critical gates. In addition, restructuring optimizations in the physical domain have to meet a large number of constrainsts. Having a large number of restructuring options increases the likelihood of satisfying the constraints. This can prove invaluable in physical synthesis where restructuring decisions face additional constraints such as the amount of space available around a selected gate.

We evaluate performance of the restructuring technique invoking it in the inner loop of a timing-closure script of a high performance physical synthesis flow [7]. This flow performs placement-driven optimization of a design layout considering the complex constraints of its physical domain. Each invocation of our restructuring technique in the timing-closure loop alternates between area and delay minimization objectives in the timing-critical portion of the design. The fanin increase constraint in these transformations is set to 1. The loop terminates when improvement in the timing slack starts falling below a specified threshold.

In the experiment we consider a set of macro-blocks from an

Table 2: Restructuring results within a timing-closure loop of physical synthesis flow.

circuit	plain		with restructuring optimization		
	slack	area	slack	area	delay impr.
macro1	-97	9146	-66	9612	0.85
macro2	-102	15196	-73	9660	0.90
macro3	-84	40634	-28	35921	0.74
macro4	-47	16465	-24	13589	0.93
macro5	-27	12309	-12	11447	0.97
macro6	-67	24299	-44	24108	0.85
macro7	-98	62508	-90	60854	0.95
macro8	-17	4494	-9	3776	0.95
macro9	25	10038	27	9808	0.98

industrial design as our test examples. Results comparing effects of using our restructuring technique on these circuits within the timing-closure script are given in Table 2. Columns two and three of the table give slack and area results of running the timing-closure script without the restructuring technique. They are compared against results in columns four and five collected after running the script with the restructuring technique. The last column in the table gives ratio of the delay reduction on the critical path, as a result of applying our technique. Overall, 11% in the area reduction, and 10% in delay improvement can be observed as a result of our technique.

5.3 Structure-driven optimization

The proposed restructuring optimizations can be applied to analyze benefits of optimizing circuit topology at the early steps of synthesis targeting its routability. It has been shown [10, 13] that circuit topology can be estimated using structural metrics. Using this approach we evaluate the improvement in routability obtained by applying the circuit topology optimizations. We applied circuit topology optimizations on 6 designs prior to technology mapping. The designs were then processed through physical synthesis for timing closure. Routability measurements were made on the final design.

Circuit topology optimizations were driven by the metric discussed in [10]. This metric uses the concept of neighborhood populations. For a given cell c_i and distance k, the neighborhood population is a set of cells residing within k nodes away from c_i. For each k, the metric averages neighborhood population sizes across all cells in the design. The objective of our optimization was to minimize these average numbers, placing a stronger weight on the numbers corresponding to smaller values of k.

Results of processing the designs through circuit topology optimizations are given in Table 3. The second column gives peak congestion and total wire length obtained without the circuit topology optimizations. The three right-most columns present the same measurements with circuit topology optimizations. The column titled cong, gives the sum of the peak horizontal and vertical congestion in the design. Peak congestion in any direction, horizontal or vertical, is defined as the maximum number of wires crossing any given bisection of the placed design. The total wire length is sum of the Steiner lengths of all wires in the placed design.

The results show that peak congestion improves by 5% overall and total wire length improves 7%. The results illustrate the benefit of applying topological optimizations driven by a good structural metric. It is made possible by having an extensive variety of Boolean restructuring choices.

Table 3: Effect of structure-driven optimization on the circuit layout.

circuit	plain		with structural pre-processing		
	cong.	wire len.	cong	wire len.	# gates
design1	667	21552	651	21347	814
design2	157	32699	145	32390	321
design3	309	85920	296	85262	553
design4	240	55543	233	56122	564
design5	1707	1149931	1604	1099344	4924
design6	548	206661	510	190774	1671
Total	3628	1552306	3439	1431472	8847

6. CONCLUSIONS

In this paper we introduced new circuit restructuring technique based on the implicit enumeration of restructuring patterns. It is able to explore efficiently a vast number of restructuring possibilities with the imposed constraints on the rewiring topologies. The experimental results show that the implemented technique produces circuits with much improved area and timing properties without undue computational costs. The incremental ability of the approach allows better assessment of circuit implementations, and makes it particularly suited at the late stages of circuit optimization, as well as the early stages of logic synthesis.

7. REFERENCES

[1] L. Benini, P. Vuillod, and G. De Micheli. Iterative re-mapping for logic circuits. *IEEE TCAD IC*, CAD-17(10):948–964, October 1998.

[2] D. Brand. Verification of large synthesized designs. In *Proc. ICCAD*, pages 534–537, November 1993.

[3] R. K. Brayton and C. McMullen. The decomposition and factorization of Boolean expressions. In *Proc. IEEE Int. Symp. Circ. and Syst.*, pages 29-54, May 1982.

[4] R. K. Brayton, G. Hachtel, and A. Sangiovanni-Vincentelli. Multi-level logic synthesis. in *Proc IEEE*, vol. 78, no. 2, pages 264-300, February 1990.

[5] F. M. Brown. *Boolean Reasoning*. Kluwer Academic Publishers, Boston, 1990.

[6] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE TC*, C-35(6):677–691, August 1986.

[7] Y. Chan, P. Kudva, L. Lacey, G. Northrop and T. Rosser. Physical Synthesis Methodology for High Performance Microprocessors. In *Proc 40th DAC*. pages 696-701, June 2003.

[8] S.-C. Chang, M. Marek-Sadowska, and K.-T. Cheng, Perturb and Simplify: Multi-level Boolean Network Optimizer. *IEEE TCAD IC,* CAD-15(12):1494-1504, December 1996.

[9] S.-C. Chang, L. Van Ginneken, and M. Marek-Sadowska. Fast Boolean optimization by rewiring. In *Proc. ICCAD*, pages 262-269, November 1996.

[10] H. T. Heineken, W. Maly, P. Nag, C. Ouyang and W. A. Pleskacz. CAD at the design-manufacturing interface. In *Proc. 34th DAC*, June 1997.

[11] V. N. Kravets. Constructive Multi-level Synthesis by Way of Functional Properties. Ph.D. Thesis, University of Michigan, Ann Arbor, 2001.

[12] V. N. Kravets and K. A. Sakallah. Resynthesis of Multi-level Circuits Under Tight Constraints Using Symbolic Optimization. In *Proc. ICCAD*, pages 687-693., November 2002.

[13] P. Kudva, A. Sullivan, and W. Dougherty. Metrics for structural logic synthesis. In *Proc. ICCAD*, pages 551-556, November 2002.

[14] W. Kunz, D. Stoffel and P. Menon. Logic Optimization and Equivalence Checking by Implication Analysis. *IEEE TCAD IC*, CAD-16(3):266-281, March 1997.

[15] E. Lehman, Y. Watanabe, J. Grodstein, and H. Harkness. Logic decomposition during technology mapping. In *Proc. ICCAD*, pages 264–271, November 1995.

[16] A. Mishchenko, X. Wang and T. Kam. A New Enhanced Constructive Decomposition and Mapping Algorithm. In *Proc 40th DAC*, pages 143148-701, June 2003.

[17] J. P. Roth and R. Karp. Minimization over boolean graphs. *IBM J. Res. and Develop.*, 6(2):227–238, April 1962.

[18] H. Savoj. *Don't Cares in Multi-Level Network Optimization*. Ph.D. thesis, University of California, Berkeley, 1992

[19] E. M. Sentovich et al. SIS: A system for sequential circuit synthesis. Technical Report UCB/ERL M92/41, UC Berkeley, May 1992.

[20] S. Sinha and R. K. Brayton. Implementation and use of SPFDs in optimization of Boolean networks. In *Proc. ICCAD*, November 1998.

[21] H. J. Touati, H. Savoj, and R. K. Brayton. Delay optimization of combinational logic circuits and partial collapsing. In *Proc. 28th DAC*, pages 188–191, June 1991.