変数の重なりのない単純な関数分解を用いた組合せ回路の改善方法

澤田 宏，山下 茂，名古屋 彰

NTT コミュニケーション科学研究所
〒 619-0237 京都府相楽郡精華町光台 2-4
Fax: 0774-93-5273
Tel: 0774-93-5285
E-mail: {sawada, ger, nagoya}@cslab.kecl.ntt.co.jp

変数の重なりのない単純な関数分解 $f(X,Y) = h(g(X),Y)$ は，関数分解の特別な形であり，1 出力関数に対する最適な組合せ回路の形を提供する．本稿では，多入力 1 出力の組合せ回路 F において，回路 F 全体の論理関数 f が上記の分解を持つ場合に，この分解を適用することで回路を改善する手法を提案する．分解後の関数 g と h を実現する新たな組合せ回路 G と H は，元の回路 F のいくつかの入力に，ある定数値を代入することで求めることができる．実験結果より，組合せ回路最適化プログラムの前処理として，本手法が特に有効であることがわかった．

論理関数，関数分解，変数の重なりのない単純な関数分解，
組合せ回路，多段論理回路

A Method for Improving Combinational Circuits using Simple Disjunctive Decompositions

Hiroshi SAWADA, Shigeru YAMASHITA and Akira NAGOYA

NTT Communication Science Laboratories
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237, JAPAN
Fax: 0774-93-5273
Tel: 0774-93-5285
E-mail: {sawada, ger, nagoya}@cslab.kecl.ntt.co.jp

Simple disjunctive decomposition $f(X,Y) = h(g(X),Y)$ is a special case of a functional decomposition, which offers an optimum circuit structure for a single-output logic function. This paper presents a method for improving combinational circuits by applying the above decomposition when exists in a function f represented by a given multi-input single-output circuit F. We can easily construct new circuits G and H that represent the new functions g and h, respectively, by assigning constant values to some inputs in the original circuit F. Experimental results show that the proposed method is very effective as a preprocessor for a combinational circuit optimization program.

logic function, functional decomposition, simple disjunctive decomposition,
combinational circuit, multi-level logic circuit

1 はじめに

論理関数 $f : \{0,1\}^n \rightarrow \{0,1\}$ を小さな複数の論理関数に分解することは，組合せ回路の自動合成・最適化において重要な要素技術の 1 つである．一般に，論理関数を $f(X \cup Y) = h(g_1(X), \ldots, g_t(X), Y)$ の形に分解することを関数分解 (functional decomposition) と呼ぶ．これに関する研究は 1960 年前後に始まる [1, 2, 3] が，分解を調べるための計算複雑度が大きく，当時は比較的小規模な論理関数にしか適用できなかった．一方，論理関数を積和形論理式で表現して，kernel 抽出と weak-division により分解を行う手法 [4] が，1980 年代に提案された．この手法は，比較的小さな計算複雑度の割りに良い分解を求めることができるため，今でも幅広く用いられている．しかし近年，LUT (Look-Up Table) 型 FPGA (Field Programmable Gate Array) 向けの論理合成手法として，関数分解が再び注目を集め，幅広く用いられるようになった．その理由は，変数集合 X の大きさを LUT の入力数に一致させることで，部分関数 g_i が 1 個の LUT で実現できるからである．さらに，OBDD (Ordered Binary Decision Diagram) [5] による効率的な論理関数処理により，適用できる論理関数の規模が拡大した [6, 7, 8].

関数分解の中でも，変数集合 X と Y が共通な変数を持たず，関数 g_i の数が 1 つ ($t = 1$ の場合) である特別なものは，simple disjunctive decomposition と呼ばれる．本稿では，これを**変数の重なりのない単純な関数分解**と呼ぶことにする．この関数分解は，変数集合を重なりのない 2 つの集合に分割し，それぞれの変数集合を持つ 2 つの論理ブロックを結ぶ結線がただ 1 本であるという点で，1 出力論理関数に対して最適な組合せ回路の形を提供する．また，その形が特殊であることから，変数集合 X と Y の選び方を工夫することができ，一般の関数分解よりも効率良く実行できる．最近，多くの研究者がこの分解に注目し，様々な効率的な分解手法を提案した [9, 10, 11, 12, 13, 14]．それらの成果により，OBDD で表現可能な規模の論理関数までなら，適用可能であると言えるほどになった．

さてここで，多入力 1 出力の組合せ回路 F において，全体の論理関数 f が変数の重なりのない単純な関数分解 $f(X,Y) = h(g(X),Y)$ を持つ場合を考える．この関数分解は，1 出力論理関数に対して最適な組合せ回路の形を提供するため，適用すべきであると考える．しかしその際に，分解後の関数 g と h を実現する組合せ回路 G と H を，どのようにして構成すれば良いかということが問題となる．関数 g や h が，さらにこの分解を持つ場合は適用していけば良いが，多くの場合，あるところからはこの特殊な分解が存在しなくなる．そのような場合には，一般的な分解形を与える他の手法で分解しなければならないが，これには多くの時間がかかり，結果として元の組合せ回路よりも悪い回路が合成される可能性がある．

この問題に対し本稿では，変数の重なりのない単純な関数分解が存在する場合に，元の回路 F のいくつかの入力に定数値を代入して，分解後の関数 g と h を実現する回路 G と H を構成する手法を提案する．これにより，元の回路に存在する性質の良い構造を保存しながら，この関数分解を適用することができる．言い換えれば，この関数分解を用いて，与えられた組合せ回路を改善することが可能となる．

以下では，まず 2 章で変数の重なりのない単純な関数分解について説明し，3 章で提案手法の詳細を述べる．4 章では実験結果とそれに対する考察を示し，5 章で結論を述べる．

2 変数の重なりのない単純な関数分解

論理関数 $f(X,Y)$ を以下の形式で表現できるとき，変数の重なりのない単純な関数分解が存在するという．

$$f(X,Y) = h(g(X),Y) \tag{1}$$

ここで，X と Y は，$X \cap Y = \emptyset$ を満たす変数の集合であり，g: $\{0,1\}^{|X|} \rightarrow \{0,1\}$ と h: $\{0,1\}^{|Y|+1} \rightarrow \{0,1\}$ は論理関数である．集合 X は **bound set** と呼ばれる．

Ashenhurst [1] は，この形の分解が存在するかどうかを調べるために，分解表 (decomposition chart) というものを用いた．これは，列と行にそれぞれ X と Y を対応させた真理値表である．もし，その列に現れるベクトルが 2 種類であれば，分解が存在することになる．なぜなら，2 種類のベクトルのどちらが選ばれたかという情報

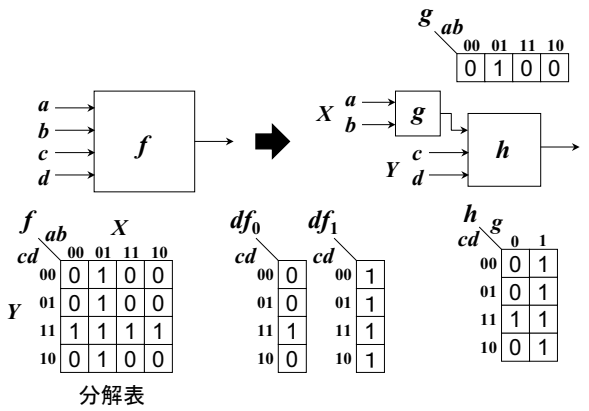

図 1: 変数の重なりのない単純な関数分解

は，1 ビットに符号化することができるからである．本稿では，関数 f と変数集合 X に対し，$SDF(f,X) = \{df \mid df = f(\epsilon, Y), \forall \epsilon \in \{0,1\}^{|X|}\}$ なるものを定義する (SDF: Set of Distinct Functions)．これは，分解表における異なる列ベクトルの集合に相当する．分解が存在する時の $SDF(f,X) = \{df_0, df_1\}$ を用いることで，式 (1) は，

$$f(X,Y) = g(X)' \cdot df_0(Y) + g(X) \cdot df_1(Y) \quad (2)$$

とも表現できる．なお，$g(X)'$ は $g(X)$ の否定を表す．

分解が存在する場合，関数 g は，式 (2) より，関数 f の df_0 と df_1 をそれぞれ 0 と 1 に置き換えることで得られる．また，関数 h は，$h = g' \cdot df_0 + g \cdot df_1$ で与えられる．

例えば，図 1に示す関数 f に対し，bound set X を $\{a,b\}$ とすると，$SDF(f,\{a,b\}) = \{df_0, df_1\}$ が得られ，その大きさは 2 であるから変数の重なりのない単純な関数分解が存在する．関数 g は df_0 と df_1 をそれぞれ 0 と 1 に置き換えることで得られ，関数 h は $h = g' \cdot df_0 + g \cdot df_1$ で得られる．それぞれの得られた結果は図 1に示す通りである．

以上のような流れで，ある bound set X に対して変数の重なりのない単純な関数分解が存在するかどうかを調べ，存在する場合は新たな関数 g と h を求めることができる．しかし，bound set X の候補は，f の変数の数に対して指数関数的に増えるため，分解を与える bound set を効率的に見つける工夫が必要となる．近年，そのための様々な手法が提案された [9, 10, 11, 12, 13, 14]．本稿では以下，これらの手法等で，関数 f に対する上記の分解が見つかっているものとする．その上で，関数 g や h を実現する組合せ回路を如何にして構成するかということについて議論する．

3 分解後の回路の構成法

本章では，組合せ回路 F が表現している論理関数 f が，変数の重なりのない単純な関数分解 $f(X,Y) = h(g(X),Y)$ を持つ時に，論理関数 g と h を実現する新たな回路 G と H を構成するアルゴリズムを示す．その特長は，元の回路 F のいくつかの変数に定数値を代入して，回路の不必要な部分を取り除くだけで，新たな回路 G と H を構成できることにある．

3.1 関数 $g(X)$ を実現する回路 G の構成

まず初めに，回路 G を求めるアルゴリズムを示す．これは，式 (2) $f(X,Y) = g(X)' \cdot df_0(Y) + g(X) \cdot df_1(Y)$ に基づき，$df_0(\delta) \neq df_1(\delta)$ を満たす定数値 δ を，元の回路 F の変数集合 Y に代入することによる．

1. $d_P(Y) = df_0(Y)' \cdot df_1(Y)$ と $d_N(Y) = df_0(Y) \cdot df_1(Y)'$ を計算する．
2. $d_P(\delta_P) = 1$ あるいは $d_N(\delta_N) = 1$ を満たす最小項 $\delta_P, \delta_N \in \{0,1\}^{|Y|}$ をそれぞれ選ぶ．$df_0(Y)$ と $df_1(Y)$ が異なる論理関数であるため，少なくとも δ_P か δ_N のどちらかは存在する．両方が存在する場合も有り得る．
3. もし δ_P が存在すれば，最小項 δ_P を回路 F の入力に代入することで新たな回路 G_P を作る．
4. もし δ_N が存在すれば，最小項 δ_N を回路 F の入力に代入し，出力を否定することで新たな回路 G_N を作る．
5. 論理値が常に定数であるような結線を G_P や G_N から削除して，それらのうちのコストの小さい方の回路を G とする．

上記の手続きで得られる回路 G は論理関数 $g(X)$ を表現している．証明は以下の通りである．δ_P は $df_0(\delta_P) = 0$ と $df_1(\delta_P) = 1$ を満たすことから，式 (2) より，G_P は $g(X)' \cdot 0 + g(X) \cdot 1 = g(X)$ を表現する．また，δ_N は $df_0(\delta_N) = 1$ と $df_1(\delta_N) = 0$ を満たすことから，式 (2) より，G_N は $g(X)' \cdot 1 + g(X) \cdot 0 = g(X)'$ の否定を表現する．

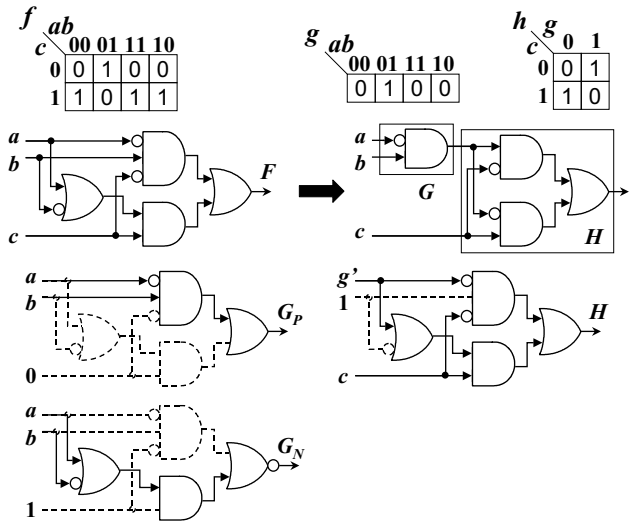

図 2: 新たな回路の構成

例 1 図2に例を示す．回路 $F = a'bc' + (a + b')c$ が与えられたとする．その論理関数 f では，$SDF(f,\{a,b\}) = \{c,c'\}$ であるから， bound set $\{a,b\}$ に対して変数の重なりのない単純な関数分解が存在する．以下では， $df_0 = c$， $df_1 = c'$ として，本アルゴリズムにより，回路 F から回路 G を構成する．まず初めに， $d_P = df_0' \cdot df_1 = c'$ を計算し， $d_P(\delta_P) = 1$ を満たす最小項 $\delta_P = 0(= c')$ を選ぶ．そして， δ_P を F に代入することで $G_P = a'b$ を得る．同様にして， $d_N = df_0 \cdot df_1' = c$ を計算し， $d_N(\delta_N) = 1$ を満たす最小項 $\delta_N = 1(= c)$ を選ぶ．そして， δ_N を F に代入して出力を反転することで $G_N = (a + b')'$ を得る． G_P と G_N の実現コストは同じであるので，どちらを選んでも良いが，ここでは G_P を G とする． □

3.2 関数 $h(g,Y)$ を実現する回路 H の構成

次に，回路 H を求めるアルゴリズムを示す．これは，式 (1) $f(X,Y) = h(g(X),Y)$ に基づき，bound set X から適当に変数 x を選び， $g_{x'}(\delta) \neq g_x(\delta)$ を満たす定数値 δ を回路 F のいくつかの入力 $X - \{x\}$ に代入することによる．

1. X から任意の変数 x を選ぶ．
2. $d(X - \{x\}) = g_{x'}(X - \{x\}) \oplus g_x(X - \{x\})$ を計算する．
3. $d(\delta) = 1$ を満たす最小項 $\delta \in \{0,1\}^{|X|-1}$ を選ぶ．変数 x が冗長な変数でない限り，$d(\delta) = g_{x'}(\delta) \oplus g_x(\delta) = 1$ を満たす δ が必ず存在する．
4. 最小項 δ を回路 F の入力に代入することで回路 H_I を作る．
5. $d(\delta) = 1$ であることから， $g(\delta) = x$ か $g(\delta) = x'$ のどちらかが成り立つ．もし $g(\delta) = x$ であれば， H_I 内の変数 x を g で置き換えることで回路 H を作る．もし $g(\delta) = x'$ であれば， H_I 内の変数 x を g' で置き換えることで回路 H を作る．

上記の手続きで作られた回路 H は，論理関数 $h(g,Y)$ を表現する．証明は以下の通りである． $g(\delta) = x$ であるときは，式 (1) より， H_I は $h(x,Y)$ を表現し， H は $h(g,Y)$ を表現する． $g(\delta) = x'$ であるときは，式 (1) より， H_I は $h(x',Y)$ を表現し， H は $h((g')',Y) = h(g,Y)$ を表現する．

例 2 (例 1 からの続き) 例 1 に引き続き，ここでは回路 F から回路 H を構成する．図2に示すように， bound set は $\{a,b\}$ であり，ここから変数 a を選択する．関数 g は， $g = a'b$ であることから， $d(\{a,b\} - \{a\})$ は， $d = g_{a'} \oplus g_a = b \oplus 0 = b$ となり， $d(\delta) = 1$ を満たす最小項 $\delta = 1\ (= b)$ を選ぶ． δ を F に代入し， $H_I = a'c' + ac$ を作る．ここでは $g(\delta) = a'$ であるから， H_I 内の a を g' で置き換えて $H = gc' + g'c$ を得る． □

変数集合 X から選ぶ変数 x は，任意のものを選ぶことができる．組合せ回路 F において，出現回数が少ない変数，あるいは出力に近い変数を選ぶことで，規模の小さい，あるいは段数の少ない回路になることが期待できる．

4 実験結果

提案した手法を用いて，与えられた組合せ回路を改善する実験を行った．手順は以下の通りである．本手法は，多入力 1 出力の組合せ回路に対して適用可能であるため，まず組合せ回路全体を maximum fanout free cone (MFFC) [15] の集合に分割する．これにより，多入力多出力の回路全体が，できるだけ少ない数の多入力 1 出力の組合せ回路に分割されることになる．そして，それぞれの MFFC F に対して，論理関数 f を計算し， [11] に示す手法により変数の重なりのない単純な関数分解が存在するかどうか調べる．存在した場合には3章で述べた手法により，新たな回路

表 1: 実験結果

回路名前	SDD + SIS #lit	比	時間	SIS だけ #lit	時間
alu2	356	0.99	3.0+ 52.1	361	49.0
apex6	736	0.99	1.7+ 10.0	743	7.5
apex7	248	1.01	0.6+ 2.2	245	2.1
b9	126	1.03	0.7+ 1.3	122	1.1
c8	127	0.91	0.6+ 0.8	139	1.0
cordic	62	0.97	0.9+ 3.4	64	0.5
dalu	938	0.96	4.9+ 69.1	979	128.5
des	3454	0.99	4.2+226.7	3472	218.7
f51m	71	0.78	0.4+ 0.4	91	1.2
frg1	136	1.00	0.9+ 22.1	136	5.2
frg2	731	0.83	3.8+ 20.0	886	35.0
i8	987	0.97	5.6+ 26.7	1015	36.4
k2	1120	0.99	3.1+ 93.5	1135	95.4
lal	104	0.99	0.7+ 1.1	105	1.3
pm1	48	0.96	0.4+ 0.3	50	0.4
rot	672	1.00	1.6+ 12.8	672	12.6
sct	80	1.01	0.6+ 0.9	79	1.3
t481	36	0.04	14.9+ 0.2	881	137.4
term1	137	0.81	2.4+ 4.7	170	8.2
ttt2	189	0.86	0.8+ 1.8	219	2.7
x1	297	1.00	2.5+ 3.8	298	3.3
x3	759	0.97	2.7+ 9.7	785	10.0
x4	389	1.01	1.4+ 4.2	385	4.4
z4ml	36	0.88	0.4+ 0.2	41	0.5
Total	11839	0.91	58.9+568.0	13073	763.7

G と H を構築して F と置き換える．そして，新たな回路 G と H に対しても，再帰的に改善を試み，変数の重なりのない単純な関数分解が存在しないところで処理を終わる．本手法の最適化能力は，各 MFFC を改善するだけであるため，複数の MFFC 間での論理の共有や，内部結線の自由度を利用した簡単化などは行われない．したがって，これらの最適化を行うために，本手法の後に SIS [16] を用いた．

表 1は，“回路” に示すいくつかの組合せ回路 [17] に対する実験結果である．“SIS だけ” は，SIS の “script.rugged” スクリプトによる結果を示す．“SDD + SIS” は，本稿で提案した我々の手法を適用した後に “script.rugged” を実行した結果を示す．“#lit” には，各ノードの factored form のリテラル数の総和を示している．これが，組合せ回路の規模に相当する．“比” は，“SDD + SIS” の “SIS だけ” に対する “#lit” の比を示している．“時間” は，Sun Ultra 2 Model 2200 上での実行時間を秒単位で表している．“SDD + SIS” の “+” の左側の数字は，[11] に示す手法で分解の存在を調べるための実行時間と 3章で述べた手法による実行時間の合計を，右側の数字は “script.rugged” による実行時間を示している．

提案手法を前処理として用いることで，全体として，9% の改善が見られた．そのための実行時間は概して少ない．回路 “f51m” や “z4ml” では，出力側に 2 入力の XOR を用いた変数の重なりのない単純な関数分解が多く見られた．SIS に見られるような，積和形論理式における kernel 抽出を基本とする分解法では，このような分解は見つけにくいように思われる．回路 “t481” に関しては，大きな改善が見られた．これは全体が，2 入力の XOR や AND ゲートで構成される tree 状の回路で実現できるものである．その初期回路は非常に複雑なものであるが，本手法は変数の重なりのない単純な関数分解のみに着目しているため，このような最適な回路の形を見つけることができたと言える．

実験結果より，本手法は，少ない実行時間のオーバヘッドで，既存の組合せ回路最適化手法の不得意な部分を補うことができることがわかった．さらに，適用することによって生じる悪い副作用が少ないことも確認できた．これは，変数の重なりのない単純な関数分解が存在しない部分の回路を，できるだけ保存するような形で回路の変形を行っているためであると考えられる．

5 おわりに

本稿では，多入力 1 出力の組合せ回路 F において，全体の論理関数 f が変数の重なりのない単純な関数分解 $f = h(g(X), Y)$ を持つ場合に，新たな論理関数 g と h を実現する組合せ回路 G と H を，元の組合せ回路 F を利用して構成するアルゴリズムについて述べた．これにより，変数の重なりのない単純な関数分解を用いて，組合せ回路を改善することが可能となった．

本手法を，一般的な組合せ回路最適化プログラムの前処理として用いたところ，いくつかの回路において，回路規模の削減や処理時間の削減に効

果があることが確認できた．あまり効果がなかった回路に対しても，実行時間のオーバヘッドはわずかであるため，本手法を前処理として用いることの有効性を示すことができたと言える．今後は，本稿で示した組合せ回路の改善手法が，実際的な設計環境で有効に利用されるように努力していきたいと考えている．

参考文献

[1] R. L. Ashenhurst, "The Decomposition of Switching Functions," *Proc. an Int'l Symposium on the Theory of Switching*, pp. 74–116, Apr. 1957.

[2] H. A. Curtis, *A New Approach to the Design of Switching Circuits*. Van Nostrand, 1962.

[3] J. P. Roth and R. M. Karp, "Minimization Over Boolean Graphs," *IBM Journal*, pp. 227–238, Apr. 1962.

[4] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang, "MIS: A Multiple-Level Logic Optimization System," *IEEE Trans. CAD*, vol. CAD-6, pp. 1062–1081, Nov. 1987.

[5] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. Computers*, vol. C-35, pp. 667–691, Aug. 1986.

[6] S. Chang and M. Marek-Sadowska, "Technology Mapping via Transformations of Function Graphs," *Proc. Int'l Conf. Computer Design*, pp. 159–162, Oct. 1992.

[7] T. Sasao, "FPGA Design by Generalized Functional Decomposition," *Logic Synthesis and Optimization* (T. Sasao, ed.), pp. 233–258, Kluwer Academic Publishers, 1993.

[8] Y.-T. Lai, M. Pedram, and S. Vrudhula, "BDD Based Decomposition of Logic Functions with Application to FPGA Synthesis," *Proc. Design Automation Conf.*, pp. 642–647, June 1993.

[9] 笹尾 勤, "論理回路の Bi-decomposition について," 情処研報, 96–DA–82, pp. 9–16, Dec. 1996.

[10] 松永 裕介, "論理回路の bi-decomposition を行うボトムアップアルゴリズムについて," 信学技報, ICD96–200, pp. 25–32, Mar. 1997.

[11] 澤田 宏，山下 茂，名古屋 彰, "対称変数の検出による関数分解の高速化と多段論理合成への応用," 信学技報, CPSY97–84, pp. 119–126, Oct. 1997.

[12] V. Bertacco and M. Damiani, "The Disjunctive Decomposition of Logic Functions," *Proc. Int'l Conf. Computer-Aided Design*, pp. 78–82, Nov. 1997.

[13] T. Sasao and M. Matsuura, "DECOMPOS: An Integrated System for Functional Decomposition," *International Workshop on Logic Synthesis*, June 1998.

[14] 湊 真一， Giovanni De Micheli, "非冗長積和形生成とその因数分解による論理関数の simple disjunctive decomposition の抽出," *DA シンポジウム '98*, July 1998.

[15] J. Cong and Y. Ding, "On Area/Depth Trade-Off in LUT-Based FPGA Technology Mapping," *IEEE Trans. on VLSI systems*, vol. 2, pp. 137–148, June 1994.

[16] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," Memorandum UCB/ERL M92/41, Univ. of California, Berkeley, May 1992.

[17] S. Yang, *Logic Synthesis and Optimization Benchmarks User Guide Version 3.0*. MCNC, Jan. 1991.