システムＬＳＩ設計技術　92－1
（１９９９．９．２０）

FPGAs用論理合成のための関数分解手法

喬健　　浅田 邦博

東京大学工学系研究科電子工学専攻

あらまし

本稿では、ルックアップテーブル(LookUp-Table)ベースのFPGAsの論理合成用関数分解の手法を紹介する。特に、面積最小（LUT数とかCLB数とかの最小）のための、関数分解にあたってのバウンド集合変数の抽出手法、及び部分依存の関数分解（Partially-dependent decomposition）、非離接性の関数分解（Non- disjunctive decomposition）のためののコンパットブルクラス（Compatible Class）のエンコーディング手法を提案する。ベンチマーク回路に対する初期の実験結果も報告し、大部の用例ではLUT数の減少が見られ、提案手法が有効であることが明らかになった。

Finding a good functional decomposition and its application to logic synthesis for LUT-based FPGAs

Jian QIAO　　Kunihiro ASADA

Dept. of Electronic Eng., Univ. of Tokyo

Abstract

This paper presents a logic synthesis approach for LUT-based FPGAs. Targeting area-minimization, our approach is based on functional decomposition. In the stage of bound-set variable's selection, we employ enumerative techniques in trying to find all simple decomposition or to find a good decomposition which has less compatible classes. In the stage of α function encoding, we try to get all possible non-disjuctive decoposition or partially-dependent decomposition to minimize the number of LUTs/CLBs. The preliminary experimental results from a set of MCNC and ISCS benchmarks show that the approach is quite promising.

1　はじめに

近年、ルックアップテーブル（LUT：Look-Up Table）ベースのFPGAsは人気のFPGA技術として、ASICのデザイン、ロジックエミュレーションなどによく使われる。この種類のFPGAはそのCLBs（Configurable Logic Blocks）が多出力で、幾つかのｋ入力のルックアップテーブルメモリからできるのである。例えば、Xilinx社のXC3000シリーズのFPGAs：一つのｋ入力のルックアップテーブル（LUT）はｋ入力の、任意のロジック関

数を実現でき、一つの2出力のCLBは全体の入力数がkに小なり、別々の入力の数がk-1に小なりの、二つのロジック関数を実現できる。本稿では、この多出力CLBsをターゲットとしてLUT-based FPGAsのための論理合成の手法を検討する。

最近、関数分解は関数の簡単化の手段として、LUT-based FPGAsのための論理合成によく応用される[2,4-8,10-15]。バウンド集合変数の抽出の問題について、[12]では権（Weight）関数が導入され、関数のSOP数式の解析を通じて各変数の権が計算されて、BS集合変数が選び出される。[4]では入力変数がデレイー（Delay）の順でBS集合変数に選ばれる。これらの手法は有効に使われるが、コンパットブルクラスの数の最小を目標とする近似手法なので、最適のバウンド集合を見逃す場合が常に存在する。一方、コンパットブルクラスのエンコーディングについて、[8]と[6]では、ベース関数の簡単化のためのエンコーディング手法が提案された。が、入力変数集合のサイズが大きい場合では、出来たベース関数が複雑になるので、この方法で最適の関数分解を見落すかもしれない。[5, 2, 7, 10]では、多出力CLBsのFPGAsを狙って、分解関数とベース関数がマージできるように、分解関数のサポート集合（Support Set）のサイズの最小化を目指すコンパットブルクラスのエンコーディング手法は提案された。ただし、分解関数の数を最小にするため、ストリクトエンコーディング（Strict Encoding）策を採用した。入力変数空間の分割に制限が置かれるので、もっと良い非離接関数分解を見逃す可能性がある。本稿では、バウンド集合変数の抽出については多段列挙の手法が考えられ、コンパットブルクラスのエンコーディングについてはベース関数再分解の容易さとか、分解関数のCLBs共有とかのエンコーディング手法が考えられる。

2 関数分解

関数分解は一つの複雑な関数$f(\mathbf{X})$を、ターゲット技術から見て比較に簡単な関数$\vec{\alpha}(\mathbf{x}_b)$と$g(\vec{\alpha}(\mathbf{x}_b), \mathbf{x}_f)$で数式（1）のように表すことである[1, 3, 9]。

$$f(\mathbf{X}) = g(\vec{\alpha}(\mathbf{x}_b), \mathbf{x}_f) \qquad (1)$$

その中、$\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$、$\mathbf{x}_b = \{x_1, x_2, \ldots, x_s\}$、$\mathbf{x}_f = \{x_{s-i+1}, \ldots, x_n\}$ $(i \geq 1)$、$\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_t)$。$\mathbf{X}$（$\mathbf{X} \in \{0,1\}^n$）は入力変数集合、$\mathbf{x}_b$（$\mathbf{x}_b \in \{0,1\}^s$）はバウンド変数集合(Bound Set)、$\mathbf{x}_f$（$\mathbf{x}_f \in \{0,1\}^{n-s}$）はフリー変数集合（Free Set）と呼ばれる。そして、関数$\vec{\alpha}$（$\vec{\alpha} \in \{0,1\}^t$）は分解関数（Decomposition Function）、関数gはベース関数（Base Function）と呼ばれる。特に、$\mathbf{x}_b \bigcap \mathbf{x}_f = \phi$の場合では、離接分解（Disjunctive Decomposition）、$\mathbf{x}_b \bigcap \mathbf{x}_f \neq \phi$の場合では、非離接分解（Non-disjunctive Decomposition）、$\| \vec{\alpha} \| = 1$の場合では、簡単分解（Simple Decomposition）、$\| \vec{\alpha} \| \geq 1$の場合では、複雑分解（Complex Deecomposition）と呼ばれる。

定義1：任意の二つの相違BSノード$\mathbf{x}_i$と$\mathbf{x}_j$ $(\mathbf{x}_i, \mathbf{x}_j \in \{0,1\}^s, i \neq j)$に対し、もし任意の$\mathbf{x}_f$ $(\mathbf{x}_f \in \{0,1\}^{n-s})$に対しても、数式$f(\mathbf{x}_i, \mathbf{x}_f) = f(\mathbf{x}_j, \mathbf{x}_f)$が成り立つと、この二つのBSノード$\mathbf{x}_i$と$\mathbf{x}_j$がコンパットブル（Compatible）であると呼ばれ、$\mathbf{x}_i \sim \mathbf{x}_j$と表れる。お互いにコンパットブルであるBSノードから組成する集合はコンパットブルクラス（Compatible Class）と呼ばれる。

関数分解（数式（1））の存在の必要充分条件：任意の二つのBSノード$\mathbf{x}_i$と$\mathbf{x}_j$に対し、もし$\mathbf{x}_i$と$\mathbf{x}_j$はコンパットブルではなければ（$\mathbf{x}_i \asymp \mathbf{x}_j$と記する）、必ず$\vec{\alpha}(\mathbf{x}_i) \neq \vec{\alpha}(\mathbf{x}_j)$が成り立つと、数式（1）のような関数分解は常に存在する[1]。

上述から見て、関数分解は所定のBS集合変数$\mathbf{x}_b$が与えられたら、適当な分解関数$\vec{\alpha}$とベース関数gを構築して、数式（1）を成り立たせることである。一般的に、関数分解にとって二つの課題がとても重要である。

- 所定の関数$f(\mathbf{X})$に対し、どうやてBS

変数の集合 $\mathbf{x}_b$ を選択していいか。つまり、どうやて $\mathbf{X}$ を $\mathbf{x}_b$ と $\mathbf{x}_f$ に分割するか。

- 所定の関数 $f(\mathbf{X})$ に対し、BS 集合変数 $\mathbf{x}_b$ が選択された後、どうやてコンパットブルクラスのエンコーディングをするか。つまり、どうやて毎個のコンパットブルクラス $c_i(i = 1, 2, \ldots, M)$ に t 桁の $\vec{\alpha}$ コードを配るとよいか。

3 バウンド集合変数の抽出

BS 変数集合の選択は直接に関数分解のコンパットブルクラスの数を決定する。コンパットブルクラスの数は本回の関数分解に必要する $\vec{\alpha}$ 分解関数の数に緊密な関係があるので、BS 変数集合の選択問題は関数分解にとって重要な問題であると思われる。この問題に対し、本稿では列挙手法（enumerative technique）をベースにする BS 集合変数の抽出手法が提案された。

本手法の方針は、各 BS 変数集合による関数分解を評価し、その中から最適の BS 集合変数を確実に捜し出す。BS 変数集合の評価のコストはコンパットブルクラスの数と $\vec{\alpha}$ 分解関数のサイズ（t の数）の統合で、小さいほど良いと考えられた。しかしながら、列挙手法は所要の計算時間が入力変数集合 $\mathbf{X}$ のサイズ n と BS 変数集合のサイズ s に比例しているので、n と s が大きい場合では、列挙法は計算の面で非現実的である。

この問題に対し、計算時間を大いに減少するため、列挙しようとする変数空間から無駄な BS 変数集合の列挙をカットするとよい。そして、BS 集合のサイズ n を k(LUT の入力数) に制限し、入力変数の集合 n を多段に分割してから毎段で列挙法を使うという BS 変数集合の抽出手法が考えられた。

まず、無だな計算（BS 集合の列挙）を削減するため、関数の特性（対称と準対称）を検討する。論文 [11] と [14] では、関数の対称（Symmetry）と準対称（Near Symmetry）の検出手法が提案された。本研究はこの対称性の検出手法を利用し、分解しようとする関数の対称特性を検討する。お互いに対称である変数は必ず同じグループとか BS 変数集合とかにし、それらの変数の間での列挙捜索をやめて、お互いに対称でない変数の間だけで列挙捜索をし、最後に BS 変数集合を求める。

本稿の BS 変数集合の抽出のアプローチ：

- ステップ 1：入力変数が全体的に対称である場合では、列挙捜索をしないで、番号の先にある k 個の変数を BS 集合の変数にして、ステップ 3 に行く。

- 準対称の場合では、お互いに対称である入力変数があれば、その変数を同じグループに分割する。対称である変数がない場合では、入力変数の全体を番号順で l（例えば、$l = 10$）に小なりの幾つかのグループに分割する。

- ステップ 2：各グループに対し、最適 BS 変数集合を列挙捜索で捜し出す。もし一つのグループの変数全体は対称ならば、列挙捜索をやめて番号の先にある k 個の変数を BS 変数にし、ステップ 3 に行く。もしグループの変数が部分的に対称ならば、お互いに対称である変数の間での列挙捜索をやめて、対称でない変数の間だけで列挙捜索をする。列挙捜索のコストは $\vec{\alpha}$ 関数のサイズ $\| \vec{\alpha} \|$ とコンパットブルクラスの数である。

- ステップ 3：選ばれた BS 変数集合での関数分解へ行く。

4 コンパットブルクラスのエンコーディング手法

所定の関数 $f(\mathbf{X})$ に対し、BS 変数集合 $\mathbf{x}_b$ を選択した後、所要の分解関数 $\vec{\alpha}$ の数も決まって（Strict Encoding より）関数分解の品質も大部決まった。関数分解に大いに影響を与えるもう一つの要素はコンパットブルクラスのエンコーディングのことである。つまり、出来た分解関数 $\vec{\alpha}$ とベース g との簡単なため、どうやて t 個の $\vec{\alpha}$ 分解関数でコン

パットブルクラス $C = \{c_1, c_2, \ldots, c_m\}$ を符合化して良いかのことである。本稿では、2出力 CLBs の FPGAs をターゲットとして、ベース関数の最小と分解関数 $\vec{\alpha}$ の共有のため、二種類のコンパットブルクラスのエンコーディング手法を統合的に考え、最適の関数分解を図る。

4.1 ベース関数 g の最小

[8] と [6] では、ベース関数 g の再分解が容易なようにコンパットブルクラスのエンコーディング手法が提案された。関数分解の存在の必要充分条件によると、各コンパットブルクラスに異なる α コードを配らなければならない。コンパットブルクラスの数は M とすると、この M 個のコンパットブルクラスを異なる $\vec{\alpha}$ コードを配るため、t ($t \geq \lceil log_2 M \rceil$) 桁が必要である。分解関数 $\vec{\alpha}$ のサイズ $\| \vec{\alpha} \|$ の最小のため、t は $\lceil log_2 M \rceil$ を取ることは一般的である (Strict Encoding と呼ばれる)。それに対し、各コンパットブルクラスの ID 番号の t 桁の二進数を t 桁の $\vec{\alpha}$ コードとして、そのコンパットブルクラスに配るエンコーディング手法は採用される。特に t が $\lceil log_2 M \rceil$ に大なりの場合では、二進値の M 以上の $\vec{\alpha}$ コードが使われていないので、ベース関数 g の最小のための Don't Care セットに見られる。でも、このエンコーディング手法の欠点は 2 出力 CLBs の特徴が充分に考えられていないので、最適の関数分解を見落す可能性が常にある。

4.2 α 分解関数のサポットセットの最小

多出力 CLBs の FPGAs を考えると、CLBs の共有を目指して各分解関数 $\vec{\alpha}$ のサポートセット (Support Set) の最小は特別に意味がある。例えば、入力変数の数が k に小なりの二つの $\vec{\alpha}$ 分解関数を一つの 2 出力 CLB に共有に載せるし、入力変数の少ない $\vec{\alpha}$ 分解関数を他のノード関数 (例えば、ベース関数 g) の中にマージすることもできる (全体の入力数は k に小なりの場合)。そこで、本稿では、特に部分依存の関数分解と非離接関数分解は考えられる。

定義 2 [7]：もし二つの BS ノード (BS Vertex) $\mathbf{x}_p$ と $\mathbf{x}_q$ ($\mathbf{x}_p, \mathbf{x}_q \in \{0,1\}^s, s = \| \mathbf{x}_b \|$) は第 j 桁だけで違うなら、$\mathbf{x}_p$ が $\mathbf{x}_q$ と x_j でお互いに隣接 (Adjacent) であり、$\mathbf{x}_p$ と $\mathbf{x}_p$ が x_j ー隣接対と呼ばれる。

定義 2 から見てわかるように、ある分解関数 α_i をある入力変数 x_j に依存させないために、必ずその α 関数のオンーセット (On-Set) に含まれる BS ノードをそれぞれ、x_j ー隣接対になるようにしなければならない。そこで、所定の関数 $f(\mathbf{X})$ と BS 変数集合 $\mathbf{x}_b$ の下で、ある x_j に依存しない分解関数を生成するため、BS 変数空間を BS ノードが皆、x_j ー隣接対になるように分割しなければならない。つまり、コンパットブルクラスをベースにして BS 変数空間を、含まれるあらゆる BS ノードが x_j ー隣接対になる、二つの部分に分けられるように s_t 個まで再分割をする。そうすると、その一つの部分を α 関数のオンーセットにし、もう一つの部分を α 関数のオフセットにしてから、その α 関数は x_j に依存しないようになる。

特に本稿では、潜在の部分依存の関数分解を捜し出すため、プライブルエンコーディング (Pliable Encoding) 策も考えられた。図 1 の例に示すように、ストリクトエンコーディング (Strict Encoding) 策を採用すれば (つまり、t を $t = \lceil log_2 M \rceil = 2$ とする)、BS 変数空間の $2^t = 4$ 分割 (図 1 (c)) から部分依存の分解関数 α が見られなくて、所定の関数を実現するには三つの CLBs が必要である (図 1 (a))。でも、プライブルエンコーディング策をとると、例えば、t を $t = \lceil log2M \rceil + 1 = 3$ にしたら、BS 変数空間を (図 1 (d)) のように分けて、非離接分解ができるようになる (図 1 (b))。そして、α_1 (x_5 に依存しない) と α_2 (x_4 に依存しない) は部分依存の分解関数で、α_3 は x_4 にしか依存しないので、結局二つの 2 出力 CLBs だけで所定の関数が実現できる。

本稿のコンパットブルクラスエンコーディ

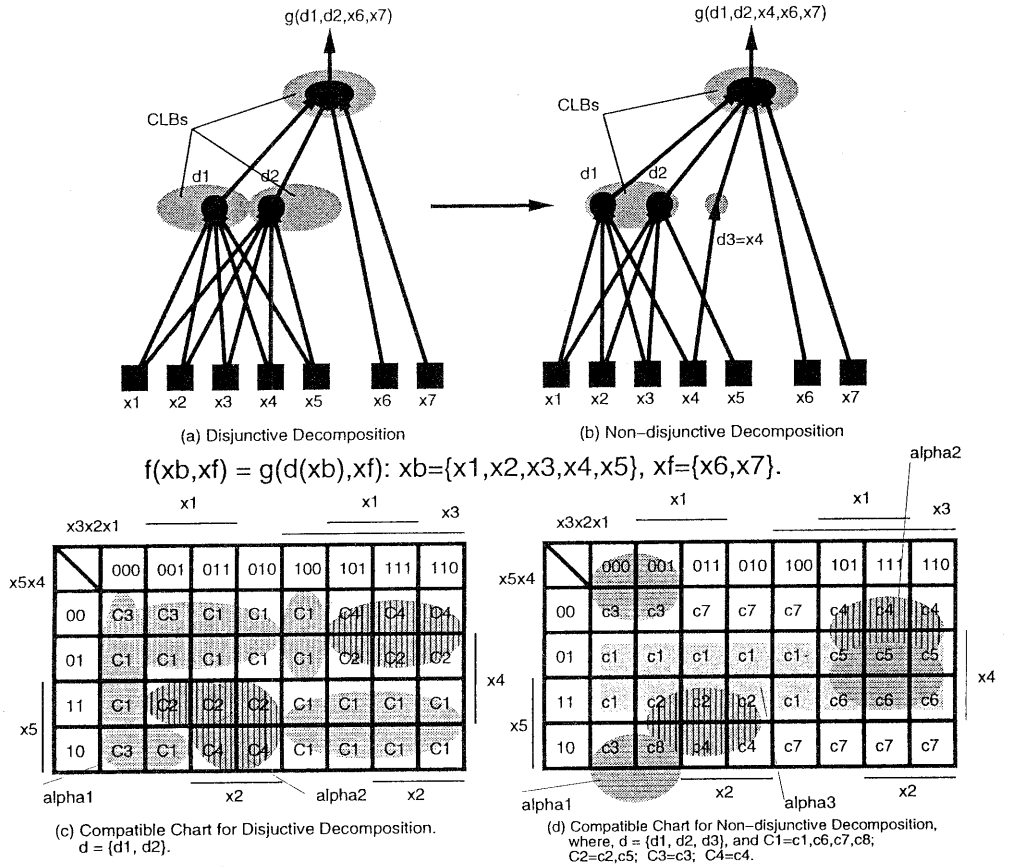

図 1: コンパットブルクラスのエンコーディング

ングのアプローチ:

- ステップ 1 : 所定のバウンド集合の下でコンパットブルクラスを計算する。
- ステップ 2 : ストリクトエンコーディング策で、BS 変数空間を分割し、部分依存の関数分解を捜索する。部分依存の関数分解が存在すれば、ステップ 5 に行って $\vec{\alpha}$ と g を計算する。
- ステップ 3 : プライブルエンコーディング策で BS 変数空間を分割し、非離接関数分解を捜索する。非離接関数分解が存在すれば、ステップ 5 に行って、$\vec{\alpha}$ と g を計算する。
- ステップ 4 ：ベース関数の再分解が容易のため、コンパットブルクラスのエンコーディングをする（部分依存分解と非離接分解とが存在しない場合）。
- ステップ 5 ：次回の関数分解。

5 実験結果

全部の提案手法はバークレイで開発された $SIS-1.2$ プログラムの中に C 言語で実装しているところである。BS 変数集合の最適抽出のアルゴリズムはもう実装された。評価のため、$MCNC$ 集合と $ISCAS$ 集合との部分ベンチマーク回路に対し、提案手法で論理合成をした。

表 1: 部分ベンチマーク回路の実験結果

Benchmarks	*usual*	*ISODEC-S*	*new*	*#PI/#PO*
	#LUT/#Level/sec	#LUT/#Level/sec	#LUT/#Level/sec	
9sym	8/3/0.1	8/3/0.4	6/3/69.	9/1
alu2	55/6/1.1	52/6/7.3	110/9/57.0/	10/6
alu4	316/11/9.6	231/11/210.3	92/12/883.4	14/8
apex2	328/16/113.3	165/11/161.9	107/12/3738.0	39/3
apex6	213/6/8.5	207/6/8.3	169/8/25.6	135/99
apex7	76/5/7.5	71/5/7.9	55/6/127.9	49/37
b9	57/5/0.6	41/4/0.9	36/4/6.8	41/21
clip	22/3/0.4	22/3/0.4	25/3/83.5	9/5
duke2	258/8/6.3	213/8/21.5	118/7/35.3	22/29
example2	156/5/2.2	139/4/2.4	100/4/80.0	85/66
frg1	116/11/12.1	40/9/55/7	30/9/12.8	28/3
i7	139/2/0.3	103/2/0.6	108/3/104.0	199/67
misex2	41/4/0.4	40/4/0.4	35/4/21.6	25/18
misex3c	200/10/4.3	152/8/29.1	135/10/126.	14/14
rd84	12/2/0.2	12/2/0.5	13/3/221.0	8/4
sao2	25/4/0.5	22/3/7.0	27/3/73.0	10/4
too_large	328/16/111.3	165/11/163.4	111/110/83.0	38/3
vda	495/7/10.5	351/8/54.4	187/9/120.0	17/39
vg2	79/9/3.5	59/9/5.7	23/6/18.2	25/8
t481	-	-	5/3/418.0	16/1

5－LUT 回路を合成した結果を表 1 に示し、提案手法は *new* と表示し、運行時間が Sun SPARC station2 での時間である。比較のため、SIS-1.2 からの結果及び最近発表された合成手法の結果も表に示す。表中の # LUT、# Level および *sec* は、それぞれ合成結果の回路内の 5－LUT の数、合成回路の段数及び所要の時間を表す（コラム 2 とコラム 3 のデータは [7] より、DEC AlphaStation 250 4/266 からの結果である。違うコンピューターからのデータので、計算時間のコラムは参考だけに使う）。

最適化の段階では、単出力ネットワークに対し、以下の SIS スクリプトで初期ネットワーク回路を生成してきたのである。

collapse

$simplify - d - mnocomp$

多出力ネットワークに対し、SIS に提供されたスタンダードスクリプトで初期ネットワークを生成してきたのである。

テクノロジマッピングの段階では、初期のネットワークから以下のスクリプトでターゲットの FPGAs にマッピングしたのである。

$xl\_partition - tm$

xl_k_decomp /＊各提案手法＊/

$xl\_partition - tm$

xl_cover

表 1 の結果から見て、我々の提案手法は半分以上でいい結果を取れた。バウンド集合変数の列挙抽出とコンパットブルクラスのエンコーディングの最適捜索のため、非常に多くの計算時間を要することが予想される。大部のベンチマーク回路が有効に合成できる。ただし、加算器 alu2 などの回路例に対し、提案手法があまり向いていないようである。

6 まとめ及び今後の課題

本稿では、多出力 CLBs の FPGAs の論理合成用の関数分解の手法を提案した。提案手法は、関数の対称特性をベースとして入力変数の集合を適当なサイズに分割した後、列挙法によって分解関数 $\vec{\alpha}$ のサイズ $\| \vec{\alpha} \|$ の最小又はコンパットブルクラスの数の最小の

関数分解に対応するバウンド集合変数を選び出し、それから2出力のCLBsを目指して、α関数の最大共有及びg関数の最小を図るための、バウンド集合変数の空間の再分割を通じ、コンパットブルクラスのエンコーディングをして、潜在の部分依存分解と非離接分解を見逃さないように最適の関数分解を実現する。

全部のアルゴリズムはまだ実装されていないが、実際にベンチマーク回路を合成した初期結果から見ると、提案手法はその性質上多くの実行時間が必要であるが、実用的であり、合成の結果が比較的によいということがわかった。今後、提案手法全体の実験結果を求めていきたい。そして、適当な非離接関数分解を検出するための、時間的に有効なアルゴリズムも今後の課題である。

参考文献

[1] R. L. Ashenhurst, "The Decomposition of Switching Functions," Proc. Int'l Symp. on Theory of Switching Functions, 1959.

[2] Jason Cong, and Yean-Yow Hwang, "Partially Dependent Functional Decomposition with Applications in FPGA Synthesis and Mapping," in Proc. ACM/SIGDA Int'l Symp. on FPGAs, pp.35-42, Feb. 1997.

[3] H. A. Curtis, "A Generalized Tree Circuit," Journal of the ACM, Vol.8(4), pp.484-496, 1961.

[4] Jason Cong, Y. Ding, "Beyond the combinatioral limit in depth minimization for LUT-based FPGA designs," in proc. IEEE Int'l Conf. CAD, pp.110-114, Nov. 1993.

[5] Juinn-Dar Huang, Jing-Yang Jou, and Wen-Zen shen, "Compatible Class Encoding in Roth-Karp Decomposition for Two-Output LUT Architecture," in Proc. ICCAD, pp.359-363, Nov. 1995.

[6] Jie-Hong R. Jiang, Jing-Yang Jou, and Juinn-Dar Huang, "Compatible Class Encoding in Hyper-Function Decomposition for FPGA Synthesis," in 35th ACM/IEEE Design Automation Conference, pp.712-717, June 1998.

[7] Christian Legl, Bernd Wurth, and Klaus Eckl, "Computing Support-Minimal Subfunctions During Functional Decomposition," in IEEE Trans. on VLSI, Vol.6, No.3, pp.354-363, Sep. 1998.

[8] Rajeev Murgai, Robert K. Bryton, and Alberto Sangiovanni-Vincentelli, "Optimum Functional Decomposition Usiing Encoding," in Proc. 31th ACM/IEEE Design Automation Conference, pp.408-414, June 1994.

[9] J. P. Roth, and R. M. Karp, "Minimization Over Boolean Graphs," IBM Journal of Research and Development, pp.227-238, April 1962.

[10] Hiroshi Sawada, Takayuki Suyama, and Akira Nagoya, "Logic Synthesis for Look-Up Table based FPGAs using Functional Decomposition and Support Minimization," in Proc. ICCAD, pp.353-358, Nov. 1995.

[11] Christoph Scholl, Dirk Moller, Paul Molitor, and Rolf Drechsler, "BDD Minimization Using Symmetries," in IEEE Trans. Computer-Aided Design, Vol.18, No.2, pp.81-99, Feb. 1999.

[12] W. Z. Shen, J. D. Huang, and S. M. Chao, "Lambda Set Selection in Roth-Karp Decomposition for LUT-Based FPGA Technology Mapping," in 32nd ACM/IEEE Design Automation Conference, pp.65-69, June 1995.

[13] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephen, R. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," U. C. Berkeley Technical Report UCB/ERL M92/41, May 1992.

[14] Feng Wang, and Donald L. Dietmeyer, "Exploiting Near Symmetry in multilevel Logic Synthesis," in IEEE Trans. Computer-Aided Design, Vol.17, No.9, pp.772-781, Sep. 1998.

[15] B. Wurth, K. Eckl and K. Antreich, "Functional Multiple-Output Decomposition: Theory and Implicit Algorithm," in 32nd ACM/IEEE Design Automation Conference, pp.54-59, June 1995.