システムＬＳＩ設計技術　104－17
(2002. 1. 24)

ブーリアンマッチングを利用したFPGAの深さ最小化マッピング手法について

松永　裕介†

† 九州大学大学院システム情報科学研究院
情報工学部門
〒816-8580 福岡県春日市春日公園 6-1

E-mail: †matsunaga@slrc.kyushu-u.ac.jp

あらまし　LUT 型の FPGA は一つの基本ブロックで定められた入力数 (通常 4 または 5) 以下の任意の論理関数を実現できるという特徴を持つ．そのため，従来は対象回路の論理関数を考慮せずに構造のみに注目したテクノロジマッピング手法が用いられてきた．ところが，実際の FPGA の基本ブロックの中には Xilinx 社の XC4000 のように 5 入力以下の任意の論理関数だけでなく，6 入力以上の一部の論理関数を実現できるものが存在する．そのような特殊な場合のマッピングを考慮するためには，マッピング対象の回路の論理関数を考慮したブーリアンマッチングを行う必要がある．本稿では関数分解に基づくブーリアンマッチングを利用して効率よく LUT 型 FPGA 用の深さ最小の回路を求めるテクノロジマッピングアルゴリズムについて述べる．

キーワード　FPGA，テクノロジマッピング，遅延最小化，関数分解，二分決定グラフ

On Delay Minimum Mapping Algorithm for FPGAs Using Boolean Matching

Yusuke MATSUNAGA†

† Department of Computer Science and Communication Engineering
Graduate School of Information Science and Electrical Engineering
Kyushu University
6-1 Kasuga Koen, Kasuga, Fukuoka, 816-8580, Japan

E-mail: †matsunaga@slrc.kyushu-u.ac.jp

Abstract　A basic block of LUT-based FPGA has a capability that it can implement any logic function whose number of inputs does not exceed a designated limit (ex. 4 or 5). Utilizing this property, many conventional technology mapping algorithms only consider circuit's structure and ignore circuit's functionality. In the case of Xilinx XC4000 series, however, it can implement a part of functions with more than 6 inputs, as well as all the functions with no more than 5 inputs. Treating such a special case, Boolean matching that considers circuit's functionality is required. In this paper, a Boolean matching algorithm for LUT-based FPGAs that is based on disjoint functional decomposition technique is described, and an efficient heuristic for delay minimum technology mapping algorithm is also shown.

Key words　FPGA, technology mapping, delay minimization, functional decomposition, binary decision diagram

1. はじめに

LUT(look up table) 型のFPGA(field programmable array) は一つの基本ブロックで定められた入力数 (通常 4 または 5) 以下の任意の論理関数を実現できるという特徴を持つ．そのため，従来は対象回路の論理関数を考慮せずに構造のみに注目したテクノロジマッピング手法が用いられてきた．ところが，実際の FPGA の基本ブロックの中には Xilinx 社の XC4000 シリーズの基本ブロックのように 5 入力以下の任意の論理関数だけでなく，6 入力以上の一部の論理関数を実現できるものが存在する．図 1 に示すように Xilinx 社の XC4000 シリーズの基本ブロック (PLB:Programmable Logic Block) は，2 つの 4-LUT と 1 つの 3-LUT から構成される．この 1 つの PLB

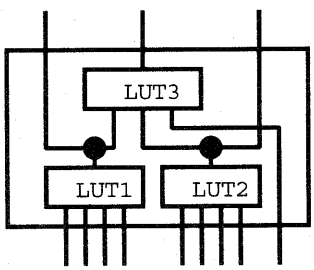

図 1　XC4000 の基本ブロック

では，

（1） 2 つの任意の 4 入力 1 出力関数
（2） 1 つの任意の 5 入力 1 出力関数
（3） 最大 9 入力の特定の 1 出力論理関数

を実現することが可能である．このうち，1 と 2 は実現できるかどうかの判定として入力数を調べるだけで行なえるので論理関数を考慮する必要がない．しかし，最後の 3 の場合には与えられた論理関数が 1 つの PLB で実現可能なのかどうかを判定するためには論理関数を考慮した処理が必要となる．

Cong と Hwang はこの判定を行なうブーリアンマッチングアルゴリズムを提案した [7]〜[9]．しかし，彼らのアルゴリズムは全ての変数の分割をしらみつぶしに列挙して PLB の構造に合った関数分解が存在するか調べるどちらかというとナイーブなもので効率的とは言いがたい．これに対し，筆者は論理関数の直交分解 (disjoint functional decomposition) アルゴリズムを利用した効率の良いブーリアンマッチングアルゴリズムを提案した [13],[14]．このブーリアンマッチングアルゴリズムは，二分決定グラフ (BDD: binary decision diagram) を用いて高速に論理関数の直交分解を求めることのできるアルゴリズム [11],[12] を変数分割を求めるために用いている．これは，論理関数の直交分解の形は唯一に定まるという性質を利用したもので，無駄な変数の分割を列挙する手間を省いている．

しかし，いくらブーリアンマッチングが高速でも，通常のテクノロジマッピングの場合にはマッチングを試す対象の関数の数が莫大になり使い物にならないという問題点がある．さいわいなことに，遅延最小 (本稿では基本ブロックの段数を FPGA 回路の遅延時間の尺度として用いる) の回路を求めるためにはすべてのマッチングを試す必要がないので，このブーリアンマッチングアルゴリズムを実際に利用してテクノロジマッピングを行うことが可能である．本稿では，その概要と実験結果について述べる．以下，2 章で FPGA 用テクノロジマッピングの従来技術および関数分解の基礎理論について述べ，3 章でブーリアンマッチングアルゴリズムの説明を行う．4 章では遅延最小化テクノロジマッピングに応用について述べ，5 章では実験結果を示すとともに考察を行う．

2. 従来の研究

2.1 FPGA 用テクノロジマッピング

前述のように LUT 型 FPGA(以下では単に FPGA と表記する) の基本ブロックは k 入力以下 (通常 $k = 4$ か $k = 5$) の任意の論理関数を実現できるので，マッピング対象の回路を k 入力以下のゲート (節点) に分解し，それらのいくつかを組み合わせて入力数が k 以下の固まりを作ることができれば，その中の論理関数がなんであっても，それを基本ブロックに対応させることができる．そこで，処理の高速化のために多くのマッピングアルゴリズムでは回路の論理関数を考慮せずに，回路の構造をグラフとして捉えて，入力数が k 以下の部分グラフ (super node とか cluster とも呼ばれる) を構成することで基本ブロックにマッピングする手法を用いている [3]〜[6]．このなかで，Cong と Ding の提案した FlowMap アルゴリズム [6] は回路の遅延時間を基本ブロックの段数で評価した場合の最小遅延解を回路サイズの多項式時間で求めることのできるアルゴリズムであり，また，k 入力の部分グラフを求める処理にネットワークの最大フロー／最小カットアルゴリズムを用いた興味深いものである．

ここでは，遅延最小解を求めるマッピングアルゴリズムに必要な用語および概念の定義を行う．

マッピング対象の回路の各ゲートを 2 入力ゲートに

分解したものをサブジェクトグラフ (subject graph) と呼ぶ．分解の仕方は一通りではないし，分解の仕方によってマッピング結果が変わりうるが，マッピング前にどのような分解が適切なのかは予測不可能なので，ここでは任意の分解を用いるものとする．この2入力ゲートはNANDやNORのみでなく任意の2入力論理関数を実現することができるものとする．この2入力ゲートの実現している論理関数を局所関数 (local function) と呼ぶことにする(注1)．以降，マッピング対象の回路とはサブジェクトグラフのことを指すものとする．また，2入力ゲートの代わりにサブジェクトグラフ上の節点という表現を用いる．節点 v のファンインとは節点 v の入力となっている節点の集合であり，$FI(v)$ と表す．節点 v のファンアウトとは節点 v の出力となっている節点の集合であり，$FO(v)$ と表す．節点 v の推移的ファンインとは節点 v の入力からたどることのできる節点の集合であり，$TFI(v)$ と表す．$TFI(v)$ は再帰的に次式のようにも表される．

$$TFI(v) = \cup_{u \in FI(v)} TFI(u)$$

節点 v の推移的ファンアウトとは節点 v の出力からたどることのできる節点の集合であり，$TFO(v)$ と表す．$TFO(v)$ は再帰的に次式のようにも表される．

$$TFO(v) = \cup_{u \in FO(v)} TFO(u)$$

グラフ $G(V,E)$(V は節点の集合，E は枝の集合) に含まれる任意の節点対 ($v_i \in V, v_j \in V$) に対して，その節点を結ぶ経路が存在する時，グラフ G を連結グラフと呼ぶ．グラフ G から節点の部分集合 $S \subset V$ とそれに接続する枝の部分集合を取り除いた結果，グラフ G が連結でなくなる時，そのような節点の部分集合 S をセパレータと呼ぶ．

サブジェクトグラフ $G(V,E)$ 上の節点 v に対して，節点 v とその推移的ファンインのみからなる部分グラフ G_v を考える．この部分グラフ G_v 上のセパレータ S が与えられたとき，G_v から S および S に接続している枝を取り除くと G_v は2つ以上の非連結な部分グラフに分解される．これらの部分グラフのうち v を含むものを v を根とするクラスタ (cluster) と呼ぶことにする (図2)．このようにクラスタは根の節点 v とセパレータ S によって定義されるので，以降，$c(v,S)$ でクラスタを表記することとする．クラスタ $c(v,S)$ の度数 (degree) を $|S|$ と定義する．すなわち，クラスタの入力側に隣接しているセパレータの節点数である．図2のようにクラスタの複数の枝がセパレータの一つの節点に接続していることもあるのでクラスタの度数とクラスタの入力側の枝数とは必ずしも一致しないことに注意．ただし，本稿では以降，クラスタの入力数という表記でクラスタの度数を表すものとする．クラスタ $c(v,S)$ に対して，v の出力の論理関数を S の各節点を入力として表したものをクラスタ関数 (cluster function) と呼ぶ(注2)．

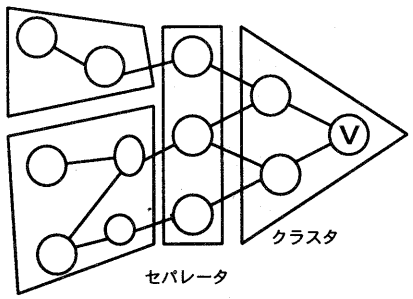

図2 クラスタ

以上の定義から k 入力LUTが度数 k 以下のクラスタにマッチすることは明らかである．そこで，FPGAのマッピング問題とはサブジェクトグラフ $G(V,E)$ を適当なクラスタの集合で被覆する問題と見なすことができる．ただし，ここでいう被覆とは一般的な被覆と少し意味が異なる．まず，サブジェクトグラフ上の全ての節点がいずれかのクラスタに含まれていなければならない，という通常の被覆条件は当然，満たされなければならないが，さらに，あるクラスタ $c(v,S)$ が解に含まれるならば，$u_i \in S_1$ であるような節点 u_i を根とするクラスタ $c(u_i, S_{u_i})$ も解に含まれなければならないという条件も満たされなければならない．このような条件の被覆問題はDAG covering問題と呼ばれ，一般的には厳密に解くことが難しい問題と言われている [2] が，目的関数が遅延時間の場合には動的計画法を用いて入力側から局所最適解を求めれば最終的に全体の最適解が得られることが知られている [6]．

一方，XC4000用のブーリアンマッチングアルゴリズムとしては個々のLUTの入力となる変数の分割をしらみつぶしに与えて図1の形のマッチングが存在するか調べるアルゴリズムが提案されている [7]〜[9]．しかし，この手法はそれほど効率がよいとは思

(注1)：もちろん，ブーリアンマッチングを行うのでなければこの局所関数は用いられない．

(注2)：局所関数と同様にブーリアンマッチングでのみ用いられる．

えない．実際，文献[9]の実験では計算時間を短縮するために，6入力以上の関数に対するマッチングで，全部で19種類あるマッチングパタンのうち6種類しか試していない．これに対して，筆者は論理関数の直交分解に基づくブーリアンマッチングアルゴリズムを提案している[13],[14]．これは，論理関数の直交分解の形が一意に定まる性質を利用したもので，直交していない(複数のLUTに入力している)入力変数を列挙するだけで残りの変数を自動的に求めることができるため，効率よく，ブーリアンマッチングを行うことができる．

2.2 論理関数の分解

まず，いくつかの用語および概念の定義を行う．

論理関数$F(X)$に対して，ある入力変数$x \in X$を0または1に固定した関数$F|_{x=0}$,$F|_{x=1}$をFのxによる**コファクター(cofactor)**と呼ぶ．$F|_{x=0}$,$F|_{x=1}$は$F_{\overline{x}}$,F_xと表されることもある．また，コファクターをとる変数xが自明な場合にはF_0,F_1とも表される．また，同様に，論理関数Fに対して複数の入力変数の値を固定したものもコファクターと呼ぶ．論理関数Fに対してxによる2つのコファクターが異なるとき($F_{\overline{x}} \neq F_x$)，$F$は変数$x$に**依存**しているという．論理関数$F(X)$の**サポート(support)**とは$F$が依存している変数の集合である．

論理関数$F(X)$を次のような2つの関数G, Hを用いて表すことができるとき，これを関数Fの**関数分解(functional decomposition)**と呼ぶ．

$$F(X) = G(X_1, H(X_2))$$

論理関数Fが式(1)の形の分解を持つとき，関数HのサポートX_2を**束縛集合(bound set)**と呼ぶ．また，関数GのサポートからHを除いた物($=X_1$)を**自由集合(free set)**と呼ぶ．

2つの(変数)集合AおよびBが共通な要素を持たないとき，すなわち，$A \cap B = \phi$のとき，2つの集合は**直交**すると言い，$A \perp B$と表記する．

式(1)の関数Hが二値の論理関数の場合を単純な分解(simple decomposition)呼ぶ．Hが多値の場合，または，複数の二値論理関数のベクタの場合を複雑な分解(complex decomposition)と呼ぶ．次に，変数集合X_1とX_2が互いに素な場合($X_1 \perp X_2$)を直交な分解(disjoint decomposition)と呼び，そうでない場合を直交でない分解(non-disjoint decomposition)と呼ぶ．

一般に関数分解は，分解された子関数G, Hに対して再帰的に適用できるので式(1)よりも複雑な形になりうるが，本稿では図1の形の関数分解を持つかどうか判定することを目的としているので，以降では式(1)の形を関数分解の標準的な形として扱うものとする．

$$F(X) = C(A(X_A), B(X_B), X_C) \tag{1}$$

明らかなように関数$A(X_A)$がLUT1に，関数$B(X_B)$がLUT2に，関数$C(a, b, X_C)$がLUT3に対応している．また変数集合X_A, X_B, X_Cは以下の制約を満たす．

$$X = X_A \cup X_B \cup X_C$$
$$|X_A| \leqq 4$$
$$|X_B| \leqq 4$$
$$|X_C| \leqq 1$$

関数分解を求めるアルゴリズムに関しては比較的古くから研究が行われている[1]が，近年，二分決定グラフを用いて単純な直交分解を効率よく求めるアルゴリズムが提案された[11],[12]．しかし，直交でない分解に関してはあまり効率の良い手法は知られていない．筆者は文献[13]において，多重度(束縛集合と自由集合の両方に含まれる変数の数)が小さい場合には適用可能なアルゴリズムを提案している．

3. LUT型FPGA用ブーリアンマッチングアルゴリズム

ここではLUT型FPGA用ブーリアンマッチングアルゴリズムについて簡単に述べる．詳細は文献[14]を参照のこと．

まず，マッチングのための場合わけを行う．図1の9つの入力は互いに独立であるが，このうちのいくつかをブリッジ接続することも可能である．ここではこのブリッジ接続によって場合わけを行なう．場合わけは2種類の独立した分類法からなる．

ひとつめは，LUT3の入力X_Cが他の変数集合と交わっているかどうかで次のように分類するものである．

Type-A: $X_A \perp X_C$かつ$X_B \perp X_C$．つまりLUT3の入力は他とブリッジしない．

Type-B: $X_A \perp X_C$かつ$X_B \perp X_C$．つまりLUT2の入力とLUT3の入力がブリッジ．

Type-C: $X_A \perp X_C$かつ$X_B \perp X_C$．つまりLUT1，LUT2両方の入力とLUT3の入力がブリッジ．

ふたつめは，LUT1の入力X_AとLUT2の入力X_Bが交わっているかどうかによる分類である．

Type-0: $|X_A \cap X_B| = 0$.

Type-1: $|X_A \cap X_B| = 1$.
Type-2: $|X_A \cap X_B| = 2$.
Type-3: $|X_A \cap X_B| = 3$.

上記の2種類の分類法は互いに独立であり直交しているので任意の組合わせが可能である(例えばA-0, C-3など: 各々Type-AとType-0, Type-CとType-3の組合わせを表すものとする). しかし, このうちのいくつかは冗長なため除外することができる. 具体的には, 例えばC-3は独立な入力を4つしか持たない. 4入力の論理関数なら1つのLUTで実現可能なためわざわざこのように複雑な場合を考慮する必要はない. 同様に, ブリッジ接続の結果残った入力数が5以下の場合には自明に実現可能なため考慮する必要がない. このことを考慮すると, 考慮すべきケースは以下の9種類となる.

- A-0, A-1, A-2, A-3
- B-0, B-1, B-2
- C-0, C-1

A-0の場合, 各々LUTの入力は互いに共通部分を持たないので, このLUTの接続構造と同じ形の単純な直交分解が存在するはずである. そこで, 文献[12]のアルゴリズムを用いて論理関数の直交分解を行い, その形がA-0にマッチするかを調べれば良い. 他のケースの場合, ブリッジしている入力があるために簡単ではないが, ブリッジしている入力を0および1に固定した関数を考えてみると, 残りの入力は互いに共通部分を持たないので, 関数の直交分解を行ってマッチングを調べることができる. 問題はどの入力がブリッジ入力かわからないということであるが, これは単純なしらみつぶしで調べる. XC4000シリーズのパタンの場合, 最大でも3つの変数しかブリッジしないのでしらみつぶしの組み合わせも高々30通りである([14]). また, 直交分解を求めるアルゴリズムの計算複雑度は論理関数を表す二分決定グラフの節点数の二乗であるが, 今回扱っている論理関数は最大で9入力であり節点数は多くて数百なので問題ではない.

4. 遅延最小化テクノロジマッピングアルゴリズム

論理を考慮しない場合は, 遅延最小解を求めるアルゴリズムとしてはCongとDingのFlowMapアルゴリズムを用いれば良いが, ブーリアンマッチングを行う場合にはクラスタの度数だけではなく, クラスタ関数も考慮しなければならないので, FlowMapのようにネットワークの最大フロー問題を応用することはできない. そこで, 各節点vに対して, そのvを根とするすべてのクラスタを列挙し, そのクラスタ関数がXC4000シリーズのPLBで実現可能かどうかブーリアンマッチングで調べる必要があるが, すべてのクラスタを列挙することは容易ではない. さいわいなことに, 遅延最小解を求めるためにはすべてのクラスタに対してマッチングを試す必要はなく, 最小解を得ることができるマッチを一つ求めてしまえば残りのクラスタに対してはマッチングを行わなくても良い.

サブジェクトグラフvに対してレベル$l(v)$を以下のように定義する. vが外部入力節点の場合$l(v) = 0$となる. vを根とするクラスタ$c(v, S)$のレベル$cl(v, S)$を次式のように定義する.

$$cl(v, S) = \max_{u \in S} l(u) + 1$$

vが内部節点の場合は$l(v)$はvを根とするクラスタのレベルの最小値となる. つまり, サブジェクトグラフの各節点に対して, 外部入力側から最小値を与えるクラスタを選択してゆけば, 回路全体のレベルが最小の解を求めることができる.

ここで, 節点vの2つのファンインをu_1とu_2とすると, 次式が成り立つ[6].

$$\max(l(u_1), l(u_2)) \leq l(v) \quad (2)$$

$$\max(l(u_1), l(u_2)) + 1 \geq l(v) \quad (3)$$

(2)式: $l(v)$を与えるクラスタ$c(v, S)$からvを取り除き, $u_1(u_2)$の推移的ファンインのみを残したものは$u_1(u_2)$を根とするクラスタになっており, かつそのレベルは$l(v)$と等しいか小さいはずである. それが$l(u_1)(l(u_2))$よりも小さいはずはない.

(3)式: vに対して$S = \{u_1, u_2\}$という節点集合は必ずセパレータになっており, このセパレータから作り出されたクラスタのレベルは$\max(l(u_1), l(u_2)) + 1$であるから$l(v)$がこの値よりも大きいはずはない.

(2),(3)式からわかることは, 節点vのレベルは自分の入力となっている節点のレベルと等しいか高々1大きいだけだということである. そこで, まず, 入力の節点のレベルと等しいレベルを持つクラスタがマッチすることがわかれば残りのクラスタのマッチを試す必要はない.

ただし, マッチが求まったからと言ってクラスタの列挙まで省略することはできない. これは節点vのクラスタ集合を作り出すのに, 節点u_1と節点u_2のクラスタ集合をマージしているからである. つまり, u_1の最初のクラスタがXC4000のPLBにマッ

チすることがわかったとしてもそのあとで v のクラスタに対するマッチングは試す必要があり，さらにそのときには u_1 のマッチングを試さなかった残りのクラスタを部分として含むマッチが存在する可能性があるからである．

以上のことから，遅延最小化テクノロジマッピングアルゴリズムは以下のようになる．

(1): 外部入力に対して空のクラスタをマッチさせる．レベルは0としておく．

(2): 外部入力側から節点 v を一つ取り出し以下の処理を行う．

(2-a): 入力節点 u_1 と u_2 のクラスタをマージして，入力数が9入力以下のもののみを残す．

(2-b): そのクラスタ集合をレベルにしたがって2つに分類する．一つは $\max(l(u_1), l(u_2))$ (clist1 と呼ぶ) もう一つは $\max(l(u_1), l(u_2)) + 1$ (clist2 と呼ぶ) となっているはずである．

(2-c): clist1 の各クラスタに対してのみブーリアンマッチングを試す．ひとつでもマッチが見つかったら処理を終わる．

$$l(v) = \max(l(u_1), l(u_2))$$

とする．

(2-d): マッチが見つからなかったら

$$l(v) = \max(l(u_1), l(u_2)) + 1$$

とする．clist1 は不要なクラスタなので削除する．clist2 のなかには $\{u_1, u_2\}$ をセパレータとする自明なマッチが含まれているはずなのでブーリアンマッチングは試さない．

3: 外部出力側から最小レベルを与えるクラスタを選択してゆく．

CongとHwangのアルゴリズムではブーリアンマッチングが高速でない (と推測される) のですべてのパタンのマッチングは試していない [9] が，提案手法のブーリアンマッチングは高速なのでマッチングを試すパタンを制限する必要はない．しかし，回路によってはクラスタ数が膨大になってしまうので，これを間引くために2つの入力のクラスタ集合をマージするときに各々のクラスタの度数に制限を設けている (たとえば5入力以下のクラスタ同士だけをマージする)．

5. 実験結果

以上のアルゴリズムをC++を用いて実装し，ベンチマーク回路に適用して実験を行った．実験の概要は以下の通りである．

- MCNC'89 の多段回路ベンチマークに対して sis の rugged.script を適用し，回路を簡単化する．
- 回路を2入力ゲートに分解する．これをサブジェクトグラフとする．
- XC4000 シリーズの基本ブロックにマップさせる．今回はレベル (ブロック段数) 最小化のみを目的関数にした．

前述のように全ての回路に対して9入力のクラスタをすべて列挙することはできなかったので，2つの入力のクラスタ集合をマージする際に，その度数が5以下，6以下，7以下の3種類の制限値でクラスタ集合を間引きした．これは1つの入力のクラスタの度数が5, 6, 7以下ということであって，マージされた結果は9入力以下という制限のままであることに留意されたい．実験結果を表1に示す．

各々のコラムのなかの 'D' および 'T' はそれぞれレベル (基本ブロックの段数) および計算時間 (単位は秒) を表している．使用計算機は Pentium-III (1GHz)，FreeBSD-4.4 である．また 'T' の括弧内の数字はブーリアンマッチングに要した時間を記している．'G', 'B[5]', 'B[6]', 'B[7]' はそれぞれ以下のように適用したマッチングの手法を表している．

G: 単純にクラスタの度数が5以下の場合のみマッチする．

B[5]: ブーリアンマッチングを行う．クラスタ列挙のためのマージの際に入力の節点のクラスタのうち度数が5以下のクラスタのみを用いる．

B[6]: ブーリアンマッチングを行う．クラスタ列挙のためのマージの際に入力の節点のクラスタのうち度数が6以下のクラスタのみを用いる．

B[7]: ブーリアンマッチングを行う．クラスタ列挙のためのマージの際に入力の節点のクラスタのうち度数が7以下のクラスタのみを用いる．

XC4000 シリーズの基本ブロックに対するマッピングにブーリアンマッチングを適用した結果は文献 [9] にも示されており，この結果でも同様に大幅な段数削減が達成できることを確認できる．さらに文献 [9] の実験ではマージの際の制限値を5にして，さらに一部のブーリアンマッチングのみを調べているので提案手法を用いた結果よりも悪いものが見られた．実験に用いた初期回路の分割が同一ではないため厳密な比較はできないが，たとえば文献 [9] では C499 が 4(3)，C880 が 7(5) となっている (括弧内は本手法の結果)．また，文献 [9] には C6288 や C7552 などの比較的規模の大きな例に対する結果は載っていない．本手法では10段以上の回路に対しての削減率が著し

表 1　マッピング結果

回路名	G		B[5]			B[6]			B[7]		
	D	T	D	T		D	T		D	T	
9symml	6	0.4	5	1.2	(0.3)	4	3.0	(1.2)	4	3.8	(0.7)
C1355	4	1.0	4	13.1	(0.8)	4	46.2	(5.8)	3	232.6	(13.1)
C1908	10	0.7	8	5.0	(1.1)	7	12.9	(3.2)	6	31.1	(5.3)
C2670	13	0.7	8	6.1	(0.4)	7	15.5	(2.2)	6	36.1	(4.8)
C3540	13	1.9	10	21.4	(6.8)	9	50.5	(14.4)	8	112.5	(17.8)
C432	15	0.2	12	1.4	(0.3)	11	3.5	(1.3)	11	5.5	(0.9)
C499	4	0.8	4	13.0	(0.8)	4	45.8	(5.8)	3	232.1	(13.1)
C5315	7	2.0	7	22.6	(4.6)	5	67.5	(14.7)	5	233.3	(19.5)
C6288	22	8.3	21	82.8	(13.8)	17	378.6	(81.5)	13	2022.8	(167.2)
C7552	11	5.4	9	85.7	(5.4)	8	233.5	(10.3)	8	953.1	(22.3)
C880	9	0.3	7	1.6	(0.3)	6	3.6	(0.5)	5	7.6	(0.6)
apex6	5	0.4	4	1.4	(0.4)	4	3.4	(1.6)	4	5.9	(2.1)
apex7	5	0.1	4	0.3	(0.0)	3	0.3	(0.1)	3	1.1	(0.7)
b9	3	0.1	3	0.2	(0.1)	2	0.3	(0.1)	2	0.7	(0.3)
des	7	3.2	6	31.3	(8.6)	5	76.6	(25.8)	4	245.2	(113.6)
f51m	4	0.1	4	0.4	(0.0)	2	1.7	(0.0)	2	8.6	(0.0)
rot	8	0.4	7	1.7	(0.3)	6	4.5	(1.4)	5	8.7	(3.0)
z4ml	3	0.0	3	0.2	(0.0)	2	0.5	(0.0)	1	1.3	(0.0)
計	149	26.0	126	289.4	(44.0)	106	947.9	(169.9)	93	4142.0	(385.0)

く，実用上の効果は十分あると期待できる．また，計算時間の内訳をみると，回路規模の大きな例になるとほとんどの時間はクラスタの列挙に要する時間であることがわかる (B[7] の場合には全体の 1 割以下)．このことから文献 [14] のブーリアンマッチングアルゴリズムが十分実用的であると言える．全体の高速化のためにはもはやブーリアンマッチングの処理の高速化は重要ではなく，列挙すべきクラスタの数を減らすなどの工夫が必要であろう．ただし，B[5] の結果と B[7] の結果を見比べると明らかに探索範囲の大小で計算時間と段数のトレードオフが存在することがわかる．処理の高速化のために中途半端に列挙するクラスタ数を制限することはマッピング結果の品質に大きな影響を与えることになるのでヒューリスティックを考えるときには注意が必要である．

6. おわりに

本稿では，LUT 型 FPGA 用のブーリアンマッチングを用いた遅延最小化テクノロジマッピングアルゴリズムについて述べ，その性能が十分実用的であることを示した．このブーリアンマッチングアルゴリズムは論理関数の直交分解を行うアルゴリズムを利用したもので，高速にマッチング判定処理を行っている．ただし，最大 9 入力のクラスタを列挙するためには多大な時間がかかるため，このクラスタ列挙にかかる時間を削減するためのなんらかのヒューリスティックを開発する必要がある．本来はマッチする可能性のあるクラスタを列挙できなければマッピング結果に悪い影響をあたえるので以下に効率よくクラスタのフィルタリングを行えるかが鍵となると思われる．

文　献

[1] R.L. Ashenhurst, "The decomposition of switching functions", *in Proceedings of an international symposium on the theory of switching*, pp. 74-116, April 1957.

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

[3] R. J. Francis, J. Rose, and Z. Vranesic, "Chortle-crf: Fast technology mapping for lookup table-based FPGAs", *in Proceedings of the 28th ACM/IEEE Design Automation Conference*, pp. 613–619, June 1991.

[4] R. Murgai, N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Improved logic synthesis algorithms for table look up architectures", *in Proceedings of the International Conference on Computer-Aided Design*, pp. 564–567, Nov. 1991.

[5] K. C. Chen, J. Cong, Y. Ding, A. B. Kahng, and P. Trajmar, "DAG-map: Graph-based FPGA technology mapping for delay optimization", *IEEE Design and Test of Computer*, pp. 7–20, Sept. 1992.

[6] J. Cong and Y. Ding, "FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA design", *IEEE Transactions on Computer-Aided Design*, vol. 13, no. 1, pp. 1–12, Jun. 1994.

[7] J. Cong, and Y.-Y. Hwang, "Parially-Dependent Functional Decomposition with Applications in FPGA Synthesis and Mapping", *In Proceedings of the ACM 5th International Symposium on FPGA*, pp. 35 – 42, Feb. 1997.

[8] J. Cong, and Y.-Y. Hwang, "Boolean Matching for Complex PLBs in LUT-based FPGAs with Application to Architecture Evaluation", *In Proceedings of the ACM 6th International Symposium on*

FPGA, pp. 27 – 34, Feb. 1998.

[9] J. Cong, and Y.-Y. Hwang, "Boolean Matching for LUT-Based Logic Blocks with Applications to Architecture Evaluation and Technology Mapping", *IEEE Transactions on Computer-Aided Design*, vol. 20, no. 9, pp. 1077–1090, Sep. 2001.

[10] R.E. Bryant, "Graph-based algorithms for boolean function manipulation", *IEEE Transactions on Computer*, C-35(8), pp. 677–691, Aug. 1986.

[11] V. Bertacco and M. Damiani, "The disjunctive decomposition of logic functions", *In Proceedings of the International Conference on Computer-Aided Design (ICCAD'97)*, pp. 78-82, November 1997.

[12] Y. Matsunaga, "An Exact and Efficient Algorithms for Disjunctive Decomposition", In Proccedings of the Workshop on Synthesis And System Integration of Mixed Technologies (SASIMI'98), pp. 44 – 50, Oct. 1998.

[13] 松永 裕介, "直交でない関数分解の効率的な列挙手法", 情処研報 2000-SLDM-96-9, pp. 57–63, May 2000.

[14] 松永 裕介, "関数分解を用いた LUT 型 FPGA 用ブーリアンマッチングアルゴリズムについて", 信学技法 VLD2000-96, pp. 161–166, Nov. 2000.