システムＬＳＩ設計技術　106－2
（2002. 5．23）

再収れん構造に着目したFPGA用ブーリアンマッチングの高速化手法について

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　基本ブロックが数個の LUT(Look Up Table) の組合わせからなる FPGA のテクノロジマッピングを行うためには，マッピング対象回路の構造だけではなく，その論理関数も考慮したブーリアンマッチングを行う必要がある．しかし，通常はブーリアンマッチングを用いてマッチが存在するかどうかを判定する候補の数は莫大であり，多大な計算時間を必要とする．本稿では，回路の構造を考慮して，ブーリアンマッチングを適用する前にマッチが存在する可能性のない候補を除外する高速化手法についてのべる．

キーワード　FPGA，テクノロジマッピング，ブーリアンマッチング

On Acceleration of Boolean Matching for FPGAs Utilizing Structural Information

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　Boolean matching, which considers Boolean functions of the subcircuits to be matched, is required for technology mapping of FPGAs having basic blocks that consis of a couple of LUTs(Look Up Tables). In general, however, there are too many candidates for matching, so that it requires numeraous computation time. This paper describes an acceleration method for Boolean matching for FPGA technology mapping, which utilizes structural relations of the subcircuits to be matched.

Key words　FPGA, technology mapping, Boolean matching

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

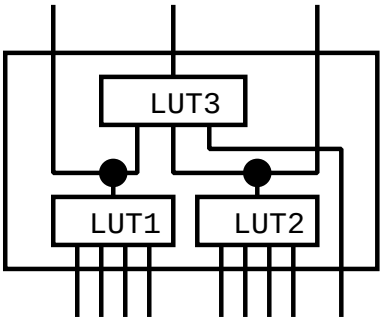

図 1　XC4000 の基本ブロック

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マッチング処理 (ブーリアンマッチング) が必要となる．この問題に対して，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アルゴリズム [7]〜[9] や，論理関数の直交分解 (disjoint functional decomposition) を利用したアルゴリズム [13]〜[15] が提案されている．

しかし，FPGA のテクノロジマッピングにこのブーリアンマッチングを用いる場合，マッチングの判定を行わなければならない候補の数が莫大なものとなり，実用化の妨げとなっている．本稿ではこの FPGA 用ブーリアンマッチングを用いてテクノロジマッピングを行う場合に，単独のブーリアンマッチングではなく，関連したたのマッチングの結果とマッピング対象の回路の構造情報を利用して全体の処理時間を短縮する工夫についての考察をのべる．具体的には，マッチングが存在しないと分かった部分回路を含む部分回路に対して，同様にマッチングが存在しない条件をもとめ，その情報により無駄なブーリアンマッチングの回数を削減しようというものである．以下，2 章で FPGA 用テクノロジマッピングの従来技術について述べ，3 章で効率化のための理論的解析および効率化手法についてのべる．

2. 従来の研究

2.1 用語の定義

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(注1)．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．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

(注1)：別の見方をすれば NAND ゲートの入力と出力の任意の位置にインバータを挿入できるものと考えることができる．

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．

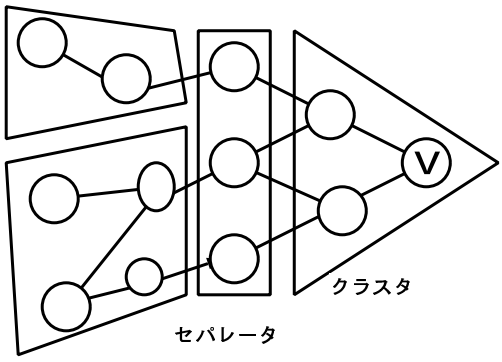

図 2　クラスタ

そのクラスタ関数が FPGA の 1 つの基本ブロックで実現できる場合，そのようなクラスタを**実現可能クラスタ (feasible cluster)** と呼ぶことにする．また，与えられたクラスタが実現可能かを判定する (さらにどのようにすれば実現できるかを求める) 問題を **FPGA のマッチング問題**と呼ぶ．

2.2　FPGA のテクノロジマッピング問題

以上の定義を用いると，FPGA のマッピング問題とは，与えられたサブジェクトグラフ $G(V,E)$ を実現可能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] (注2)．

図 1 に示した XC4000 用の基本ブロックで実現可能なクラスタを求める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調べるアルゴリズム [7]〜[9] や，論理関数の直交分解に基づくアルゴリズムが提案されている [13]〜[15]．本稿では，実現可能クラスタを求めるアルゴリズムに関しては任意のものを使用するものとして，全体のテクノロジマッピングの効率化に関する検討を行う．

マッピングの目的関数が面積 (使用ブロック数) か遅延 (ブロック段数) かによって細部は異なるが，ブーリアンマッチングを用いた FPGA のテクノロジマッピングアルゴリズムの概略は以下のようになる．

（1） サブジェクトグラフの節点を入力側からのトポロジカル順に整列させる．

（2） 節点を 1 つ取り出し，その節点を根とするクラスタをすべて列挙する．

（3） 列挙されたクラスタのなかから実現可能クラスタを求める．

（4） 実現可能クラスタのなかでもっとも評価値の高いものを選ぶ．

（5） 処理されていない節点が残っていたら 2. へ戻る．

（6） 出力側の節点から選択されたクラスタをたどることでクラスタによる被覆を求める．

このうち，もっとも計算時間を要するのが 3 の実現可能クラスタの判定 (ブーリアンマッチング) である．文献 [15] では遅延最小解を求めることに特化することでこのブーリアンマッチングの回数を減らしているが，一般的には解の品質を維持しつつこの回数を

(注2)：実現可能クラスタの定義を与えられたセルライブラリに含まれるセルで実現可能と変えれば，以上の定義は一般的なセルベースのテクノロジマッピングのものとなる．

減らすことは難しい.

3. クラスタ相互の構造の関係を用いた高速化手法

前節でのべたようにブーリアンマッチングを用いたテクノロジマッピングでもっとも計算時間を要するのはクラスタが実現可能か判定する部分である．回路の構造にもよるが，多くの場合，ある節点を根とするクラスタは数十から数百となり，全体として試さなければならないブーリアンマッチングの回数は莫大なものとなる．ただし，それらは全く無関係なわけではなく，もともとは同じサブジェクトグラフから切り出されたクラスタであり，なかには共通の部分をもつクラスタも多数存在する．そこで，単独のブーリアンマッチングの高速化とは別に，複数のブーリアンマッチング処理を全体として高速化するための手法を提案する．これは以下のような直感的な考察に基づくものである (厳密な定理は後述).

“もしも，あるクラスタが実現可能でなければそのクラスタを部分グラフとして含むようなクラスタも実現不可能である．”

つまり，クラスタ相互の包含関係を記録しておいて，実現不可能なクラスタを部分クラスタに持つクラスタはブーリアンマッチングを試す候補から除外すれば無駄なブーリアンマッチングを起動する回数を減らすことができる.

3.1 ブーリアンマッチングの高速化に関する理論的考察

まず，いくつかの定義を行う.

［定義 1］ 同一の節点 v を根とする 2 つのクラスタ $c_1(v, S_1)$ と $c_2(v, S_2)$ に対して，c_1 に含まれるすべての節点，および枝が c_2 にも含まれるとき，c_2 は c_1 を**包含**するという．クラスタ c_1 を包含するクラスタ c_2 から，c_1 に含まれる節点と枝を取り除いた部分グラフを c_2 の c_1 による **残余グラフ**と呼び，$R(c_2, c_1)$ と表す.

この残余グラフ $R(c_2, c_1)$ が 1 つもしくは複数の木状グラフからなる場合，c_2 は c_1 を**単純に包含する**と言う．以降ではこの単純な包含関係を $c_2 \supset c_1$ と記すことにする． □

［補題 1］ クラスタ c_2 がクラスタ c_1 を単純に包含する場合，c_2 のクラスタ関数 F_{c_2} の変数のいくつかを 0 または 1 に固定することで，c_1 のクラスタ関数 F_{c_1} に一致させることができる.

証明: c_2 は c_1 の単純な包含なので，残余グラフ

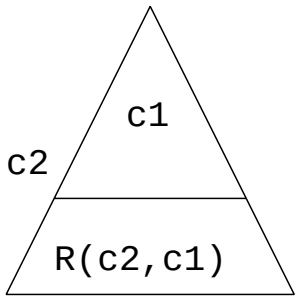

図 3 クラスタの包含，残余

$R(c_2, c_1)$ は 1 つもしくは複数の木で構成される．木状回路の場合，任意の入力から出力までの一本の経路を活性化するように残りの入力の値を 0 または 1 に決定することができるので，そのような値の割り当てを c_2 のクラスタ関数 F_{c_2} に対して行えば c_1 のクラスタ関数 F_{c_1} を得ることができる． □

補題 1 から次の定理が導かれる.

［定理 1］ 実現可能でないクラスタ c_1 を単純に包含するいかなるのクラスタもまた実現可能ではない.

証明: クラスタ c_2 が c_1 を単純に包含し，かつ実現可能と仮定する．この場合，補題 1 から c_2 の入力のいくつかを 0 もしくは 1 に固定することによって c_1 のクラスタ関数と同一の論理関数を実現することができる．クラスタ c_2 が実現可能とすると，c_2 のクラスタ関数 F_{c_2} を実現するような基本ブロックの設定が存在することになる．そのような設定に対して，いくつかの入力を 0 もしくは 1 に固定することで c_1 のクラスタ関数 F_{c_1} を得ることができる．つまり，クラスタ c_1 は実現可能ということになり矛盾が生ずる．よって，クラスタ c_2 が c_1 を単純に包含し，かつ実現可能という仮定が誤りだとわかる． □

この定理により，あるクラスタが実現不可能と判定されたらそのクラスタを単純に包含するクラスタのブーリアンマッチングは試さなくて良いことがわかる.

3.2 単純な包含関係の保持

クラスタの包含関係(注3)は無駄なブーリアンマッチングを排除するためには有益だが，すべてのクラスタの間の包含関係を保持しようとすると，最悪 $O(n^2)$(n はクラスタ数) の記憶領域，計算量が必要となる．実際には，このようにすべての関係を保持する必要はなく，平均的には $O(n)$ 程度の情報を保持しておけば十分である.

(注3)：以降では単純な包含関係のことを包含関係と記すことにする.

［定義 2］ クラスタ c に包含されるクラスタの集合を**内包クラスタ**と呼び，$\mathcal{I}(c)$ と表す．つまり，$I(c) = \{d|c \supset d\}$ である． □

［定義 3］ クラスタ c の内包クラスタ $\mathcal{I}(c)$ の要素のうち，他の要素には包含されないクラスタを**極大内包クラスタ**と呼び $\mathcal{P}(c)$ と表す．つまり，$\mathcal{P}(c) = \{d|d \in \mathcal{I}(c), \forall e \in \mathcal{I}(c), e \not\supset d\}$ である． □

［定理 2］ クラスタ c が実現可能かの判定に際して次の 2 つの命題は等価である．

（1） 全内包クラスタ $\mathcal{I}(c)$ の中で実現不可能なクラスタが含まれる．

（2） 極大内包クラスタ $\mathcal{P}(c)$ の中で実現不可能なクラスタが含まれる．

証明:

(1) ⇒ (2)

まず，$\mathcal{I}(c)$ 中の実現不可能なクラスタを d とする．定義により，d を包含する極大内包クラスタが必ず存在する．これを e とすると，定理 1 から e も実現不可能となる．

(2) ⇒ (1)

$\mathcal{P}(c)$ 中に実現不可能なクラスタが存在したとすると，そのクラスタは同時に $\mathcal{I}(c)$ の要素でもあるので (1) が成り立つ． □

定理 2 からブーリアンマッチングのフィルタリングのためには各クラスタに対して極大内包クラスタ $\mathcal{P}(c)$ の情報だけを保持しておけば良いことがわかる．

3.3 極大内包クラスタの計算

ここでは，節点 v を根とするクラスタの集合が与えられたときに，その各要素のクラスタに対する極大内包クラスタの計算方法についてのべる．

まず，クラスタ c とその極大内包クラスタ $\mathcal{P}(c)$ の間には次のような関係がある．

［補題 2］ クラスタ c の節点数を N_c とし，c の極大内包クラスタ $d \in \mathcal{P}(c)$ の節点数を N_d とすると，$N_c = N_d + 1$ が常になり立つ．

つまり，節点 v に対するクラスタをその節点数に応じて分類し，自分よりも節点数が 1 少ないクラスタとの間に単純な包含関係が成り立つかどうか調べれば良い．

4. おわりに

本稿では，LUT 型 FPGA 用のブーリアンマッチングを行う際の候補を絞り込む手法として，クラスタの包含関係に基づくフィルタリングアルゴリズムを提案した．これは LUT 型 FPGA 用ブーリアンマッチングに強く依存した性質を利用したもので，ブーリアンマッチングを行う前にクラスタ間の構造を調べておけば容易に判定を行うことができる．今後，実際のテクノロジマッピングのアルゴリズムに組み込んで有効性を検証するとともに他の効率化手法を検討していく予定である．

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.
[15] 松永 裕介, “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”, 信学技法 VLD2001–138, pp. 45-52, Jan. 2002.