計算機アーキテクチャ　98－3
設　計　自　動　化　65－3
（1993. 1. 21）

配置・配線を保存する論理回路の再設計について

藤田昌宏
富士通研究所
川崎市中原区上小田中1015

久木元裕治
カリフォルニア大学バークレー校
Berkeley, CA 94720, USA

あらまし

フィールドプログラマブルゲートアレイ（FPGA）は、近年、集積度・スピードが向上し、システムのプロトタイプ作成などでさかんに利用されるようになっている。FPGAはプロトタイプ作成に利用されるため、設計変更も頻繁に発生すると考えられるが、仕様の一部を修正した場合でも、論理合成した結果の回路は大きく変わることが多く、結果的に回路の遅延が大きく変化してしまうことも多い。本論文では、表参照型FPGAを特長を活かした再設計手法を示す。仕様変更が発生した場合でも、回路構造を変えるのではなく、回路中の万能セルの実現する論理関数のみを適切に変化されることにより、新仕様に回路を合わせるようにする。このようにすることで、回路はネットリストとしては変化せず、既存の配置・配線をそのまま利用することができるため、遅延の変化も生じない。本稿では、問題をBoolean Relationを使って定式化する効果的な再設計手法を示す。

和文キーワード　再設計、特性関数、Boolean Relation、フィールドプログラマブルゲートアレイ

A Redesign Method Preserving Placement and Routing of Logic Circuits

Masahiro Fujita
FUJTISU LABS. LTD.
Kawasaki 211, Japan

Yuji Kukimoto
University of California, Berkeley
Berkeley, CA 94720, USA

Abstract

Field programmable gate array (FPGA) makes rapid prototyping an easier task, and is useful in many applications due to its growing speed and capacity. Since FPGA is especially useful for prototyping, there may be many design modifications when developing a chip. However, even if one modifies only small part of designs, the circuit can be drastically changed due to logic synthesis process. This implies that the delay of a circuit can be completely different after design modifications, which may cause serious problems to designers. In this paper, we present a rectification method for look-up table type FPGA's. Instead of changing the netlist of a circuit, we only modify functionality realized by look-up tables and keep the netlist equal so that there will be no change on the delay of the circuit. We formalize the problem using characteristic functions and present a redesign method based on Boolean relation techniques.

英文 key words　redesign, characteristic function, Boolean Relation, FPGA, rectification

1. はじめに

各セルが、入力数が一定数以下であれば、任意の論理関数を実現できるテーブル参照型のフィールドプログラマブルゲートアレイは、その設計の容易さから、近年重要が飛躍的に増大している。これに伴い、フィールドプログラマブルゲートアレイのためのCAD、特に論理合成に関して、近年、極めて活発に研究が進めら、数多くの論文が発表されている[5,8,9,10,11,13,16,18,19,20,25]。

フィールドプログラマブルゲートアレイはシステムのプロトタイプの際に使用されることも多いが、そのような場合には、設計変更は頻繁に行なわれると考えられる。設計変更により回路が実現しなければならない論理関数が変化した場合には、通常、回路を修正し、配置・配線仕直すため、配置・配線パターンが大きく変化することが多い。

フィールドプログラマブルゲートアレイでは、遅延はセル自体よりも配線の影響が大きい。このため、結果的に回路の遅延も大きく変わる可能性が高い。これでは、設計者が回路の性能を予測することが難しく、設計期間の増大につながる恐れがある。

そこで、ここでは、回路の論理仕様が変更された場合でも、配置・配線結果を全く変えず、変更するのは各セルが実現する論理関数のみに限定する。こうすることにより、配置・配線が変わらないので、修正後の回路の遅延と元の回路の遅延は一致し、設計者も容易に性能を予測することができる。

ここで扱う問題、つまり、仕様の一部変更があった時に設計を新しい使用に合わせる再設計に関しては、従来、いくつかの研究があるのみである。新しい仕様のための設計を既存の設計をできるだけ再利用する形で行う手法[12,14]や回路を追加することで新しい仕様に合わせる手法[24]などが研究されている。しかし、これらの手法では、回路構造が変化するため、回路遅延も大きく変化する可能性があり、本稿で扱う問題には不向きである。

本稿は、以下のような構成になっている。2章で、例題を示すことにより、ここで取り扱おうとしている問題を明確にする。3章で、使用する用語を説明し、4章で、問題を定式化する。5章で、我々の手法を示し、6章では実験結果を示す。最後の7章は結論である。

2. 例題

今、仕様として、下に示す論理関数が与えられたとする。

o = ab + bc + ac

s = a'b'c + a'bc' + ab'c' + abc

（本稿では、英小文字１字または、英小文字１字に数字を付けたもので変数を表わす。AND記号は省略し、ORは「＋」で表わす。否定は、否定と取る式の後に「'」を付ける）

ここでは、２入力以下のすべての論理関数を実現できる万能セルを使用して回路を構成するとする。上の仕様を満たす回路の設計例を図１に示す。

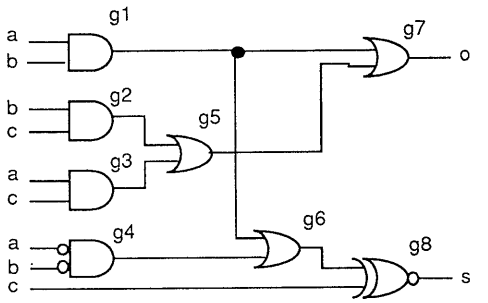

図１　元の仕様から生成された回路例

次に、仕様が変更され、出力oが以下の論理を実現しなければならないとする（出力sはもとの仕様のまま）。

o = a' + b' + c'

ここでは、配置・配線を一切修正しないという条件で考えているため、図１の回路の接続関係を変化させることはできず、できるのは、図１の各万能セルが実現する論理関数を変更することのみである。上の新しい仕様を満たすには、実はゲートg7を図２のようにORゲートからNANDゲートに変えれば良い。この場合には、回路中の１つのゲートを変更するだけで新しい仕様に合わせることができている。

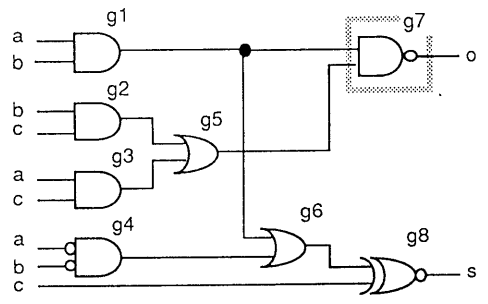

図２　新しい仕様に合わせた回路１

さらに、仕様が変更され、oとsが以下のように修正されたとする。

o = a' + b' + c'

s = a + b + c

この場合には、回路中の1つのゲートの変更のみでは対応することができず、ゲート g1, g5, g6, g8の4つを同時に図3のように修正することで、上の新しい仕様に合わせることができる。この際、ゲートg6は単に片方の入力を直接出力しているだけとなっている（これは、各ゲートが万能セルであるということから実現できる）。

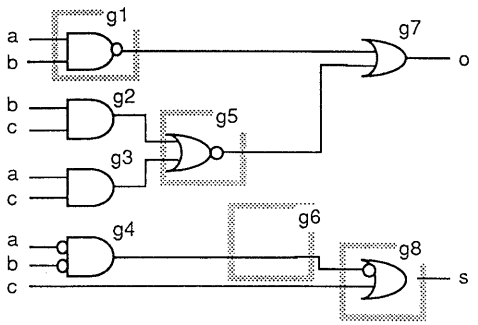

図3　新しい仕様に合わせた回路2

以上のように、回路の接続を一切変更せず、万能セルの論理関数を変更するのみで、与えられた新しい仕様に合わせることがここで扱う問題であり、仕様の与えられ方によっては、複数のセルを同時に変更する必要がある。

3. 用語の定義

ここでは、以下で使用する用語を定義する。尚、Boolean Relationの詳しい説明については文献[2,22,23]を見られたい。

定義1：

Boolean Relationとは、2つの集合B^m, B^nの間の関係Rである（ただし、Bは{0,1}であり、0と1からなる2値を示す）。

定義2：

Boolean Relation Rがwell-definedであるとは、B^mのすべての要素xに対し、B^nのある要素yが対応し、(x,y)がRを満たすことである。

定義3：

Boolean Relation Rが与えられた時、B^mからB^nへの論理関数fがRを満たすとは、B^mのすべての要素xに対し、(x,f(x))がRを満たすことである。

定義4：

Boolean Relation Rの特性関数Rとは、$B^m \times B^n$からBへの1出力論理関数であり、R(x,y)=1の時またその時に限って(x,y)がRを満たすことである。

smooth演算　$S_x f$とは、$S_x f = f_x + f_{x'}$のことである。

また、consensus演算$C_x f$とは、$C_x f = f_x f_{x'}$のことである。

4. 問題の定式化

以上の定義を用いて問題を定式化すると以下のようになる。

問題：

すべて万能セルからなる組み合わせ回路Nが与えられたとする。この回路が実現するB_mからB_nへの論理関数をfとする（回路の入力数をm、出力数をnとする）。ここでは、配置・配線は既に終了しているとする。gを新たな仕様に対応するB^mからB^nへの論理関数とし、fとは等しくないとする。Gを回路N内のすべての万能セルの集合とし、G_sをそのサブセットとする。この仮定の元で、G_s内の万能セルの論理関数のみを変更することによって、回路Nが実現する論理関数をfからgに変えることが、ここで取り扱う問題である（ただし、いつも解があるとは限らず、修正できない場合もある）。

5. 提案する手法

ここでは、以下のようにして、上の問題を解いていく。

(1) 論理関数を変更する万能セルの集合G_sを決める。

(2) G_sに許されるBoolean Relation Rを求める。

(3) 入力変数の制限を考慮して、Boolean Relation R に制限を加え、R_cとする。

(4) 入力変数の組み合わせの制限を考慮して、R_cの解となる関数を求める。もし求まれば、それがG_s内の万能セルが新たに実現すべき論理関数である。

5.1 論理関数を変更する万能セルの集合G_sの決定

ここで決定した集合G_s内の万能セルのみの論理を変更でき、それ以外の万能セルの論理が変更できないとする。これは、設計者から与えられる場合もある。設計者からの指定が無いときは、原理的には全ての万能セルの論理を変更できるが、変更する万能セルの数が多いほど、問題が難しくなるので、ここで、有効だと思われる万能セルの集合を決める。ここに示す以外にも様々な方法が考えられる。

まず、回路を入力からの段数で区切っていき、同じ段数のものを１つのクラスタと考え、それをG_sとする。ある段数のクラスタをG_sとして、本手法を適用し、失敗した場合には、他の段数のクラスタをG_sとして再度、本手法を適用することを繰り返す。

5.2 G_sに許されるBoolean Relation の計算

ここでは、文献[4]の手法によって、Boolean Relation を計算する。文献[21,17]でも同様の手法が述べられている。また、実装は、２分決定グラフ[3,1]で行なうことで、より大きなBoolean Relationを扱えるようにする。

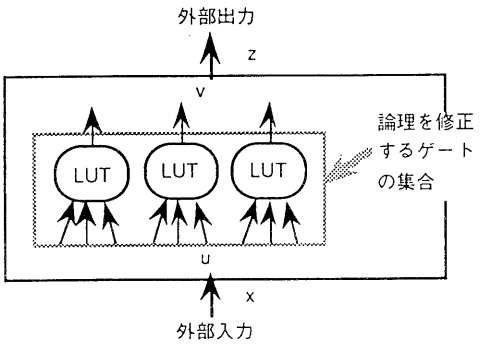

図４　取り扱う多段論理回路の構成

図４に多段論理回路の構成を示す。図中、LUTは万能セルを示している。破線で囲った部分がG_sに対応し、これらが論理を変更しようとしている万能セルである。ここで、$x = \{x_1, ..., x_m\}$は回路全体の入力とし、$z = \{z_1,...,z_n\}$は回路全体の出力とする。また、論理を変更しようとしている部分回路への入力を$u = \{u_1,...,u_p\}$とし、部分回路からの出力を$v = \{v_1,...,v_q\}$とする。

$f_i(x)$で、u_iが実現する論理関数を外部入力で表現したものとする。FI(x,u)は、外部入力xがuをどのように制御できるかを示す特性関数である。

$$FI(x,u) = \prod_{i=1}^{p} (u_i = f_i(x))$$

$g_j(x,v)$で、z_jが実現する論理関数を外部入力xと内部ノードvによって表わしたものとする。FO(x,b,z)は、zからxとvをどのように観測できるかを示す特性関数である。

$$FO(x,v,z) = \prod_{j=1}^{n} (z_j = g_j(x,v))$$

定理１[4]：

Spec(x,z)を新しい仕様を表わす特性関数とする。回路全体の論理関数がSpec(x,z)となるためには、論理を変更する部分回路の出力論理が以下のBoolean Relation R(u,v)を満たさなければならない。

$$R(u,v) = C_x(FI(x,u)' + S_x FO(x,v,z)Spec(x,z))$$

図１の回路に対し、ゲートg1, g2, g3, g4の出力をvとし、uとして外部入力a, b, cを使った場合に、定理１から求めたBoolean Relation R(u,v) = R(a,b,c,g1,g2,g3,g4)を図５に示す。図５のa, b, c が全て０の場合を見ると（最初の行）、許されるg1, g2, g3, g4の値は0010, 0100, 0110の３種類であることが分かる。これらの値は通常のドントケア値（「-」で表記）を含む論理仕様では、表現できないことに注意されたい。

a b c	g1g2g3g4
0 0 0	0010, 0100, 0110,
0 0 1	0010, 0100, 0110,
0 1 0	0011, 0101, 0111, 1- - -
0 1 1	0010, 0100, 0110,
1 0 0	0011, 0101, 0111, 1- - -
1 0 1	0010, 0100, 0110,
1 1 0	0011, 0101, 0111, 1- - -
1 1 1	0000

図5　図1の回路から抽出したBoolean Relation

上のBoolean Relationは、必要条件である。十分条件となるには、回路の接続が一切変更できないので、各部分回路の万能セルについて、入力に関する制限も満たさなければならない。この制限のことを以下では、単に入力制限と呼ぶ。定理1により計算されたBoolean Relationを「元のBoolean Relation」と呼び、Rで表わす。また、入力制限も考慮したBoolean Relationを「制限されたBoolean Relation」と呼び、R_cで表わす。

```
   restrict_relation(R)
   {
1    do {
2      R_old = R;
3      foreach fanin u_i ∈ u
4        V_s = {v_j ∈ v | u_i is an alement of
                          v_j's support};
5        R = R·C_ui S_vs R;
6      } while (R ≠ R_old);
7      return(R);
   }
```

図6　制限されたBoolean Relationを求めるアルゴリズム

図6に元のBoolean Relation Rから制限されたBoolean Relation R_cを計算するアルゴリズムを示す。2行目から5行目のループになっているところが処理の中心である。まず、4行目でノードu_iに接続されているノードをV_sとして列挙する。次にV_sに含まれる変数に関し、smooth演算を施すことにより、V_sに含まれる変数を除去してBoolean Relationを求める。これにu_iに関しconsensus演算を施し、元のBoolean Relationとの積を計算することを各u_iについて実行する。その結果1つ前の結果と異なる場合には、以上の処理を再度行なう。この処理は、求めているBoolean Relationが処理毎に変化しないか小さくなるかのいずれかのため、必ず終了する。

定理2：

与えられたBoolean Relationを与えられた入力制限の元で満たす関数fが存在するならば、その関数fは、対応する制限されたBoolean Relation R_cも満たす

（証明略）

定理3：

もし制限されたBoolean Relation R_cを満たす関数が1つしか無ければ、その関数は、入力制限も満たす。

（証明略）

定理4：

制限されたBoolean Relation R_cがwell definedであっても、入力制限を満たす関数が存在するとは限らない

（証明略）

この定理4から、通常のBoolean Relationでは、入力制限を正確には扱えないことが分かる。Boolean Relationでは、許される出力値から実際の出力値を選択する際には、各入力値に独立に選ぶことができるが、入力制限を考慮するには、入力値を考慮して出力値を選ぶ必要がある。特に、他の入力に対してどの出力値を選んだかによって、今の入力値での出力値の選び方が影響される必要があるが、これはBoolean Relationでは表現できない。

5.3 入力制限のもとでのBoolean Relationを満たす関数の抽出

制限されたBoolean Relation R_cが得られた時点で、まずそれがwell definedであるかを調べる。もし、well definedでなければ定理2から解が存在しないことが分かる。well definedの場合には、自動修正に成功する可能性がある（定理4から必ず成功するとは言えない）。

文献[6,7]に、入力制限を考慮して、ここで考えている問題と同様の問題を解く手法が与えられている。[6,7]では、順序回路の最適化問題をrecurrence equationと呼ば

れる問題として形式化しているが、結果的に本稿で扱っている問題と等価になっている。しかし本稿では、表参照型のフィールドプログラマブルゲートアレイを対象としているため、解として得られる論理関数の複雑さは関係なく、とにかく解が得られればそれで良いため、よりシンプルな手法で解くことが処理時間や扱える回路規模などの点で良い。

図7に我々の提案するアルゴリズムを示す。これは、既に求めた制限されたBoolean Relation R_cがwell definedの時に起動される。しかし、定理4から解が必ずしも存在するとは限らない。ここでは、単純な方法を用いており、まず、5行目に示すように、1つの出力のみについて入力制限を満たす論理関数を1つ決める。そして、その決まった論理関数を仮定して、他の出力のための制限されたBoolean Relation R_cを修正する(6行目)。この手続きは、全出力について論理関数が決定でき、解が得られるまで続けられる。3行目では、全出力に対する制限されたBoolean Relation R_cから今取り扱っている出力v_iのみのBoolean Relationを求めている。

この結果得られるBoolean Relationは1出力のみであるため、簡単にこのBoolean Relationを満たす関数のONセット、OFFセット、DCセット(ドントケアセット)を次のように計算することができる。

ONセット： $R(i)_{vi}\, R(i)'_{vi}$,

OFFセット： $R(i)'_{vi}\, R(i)_{vi}$,

DCセット： $R(i)_{vi}\, R(i)_{vi}$,

これらの3つの集合からv_iのための関数を1つ決めることは簡単である。

図7のアルゴリズムは、最悪の場合にはすべての場合を調べることになるため、バックトラック回数に制限を加え、それを越えた場合には、解を見つけられなかったとしてあきらめる。

実験的には、次章に示すように、かなりの場合に解を見つけることに成功しており、実用性は高いと言える。

```
restrict_function(Rc)
  {
1   foreach vi in V {
2     Us = {uj|uj is not in the support
                of vi };;
3     R(i) = CUsSV-{vi}Rc;
4     if (R(i) is well-defined)
5       choose a compatible function
                              fi(u) for vi;
6       Rc = Rc|vi = fi(u);
7     else
8       backtrack to the previous
                              choice point
9   }
10  if(a compatible function is found
                              for all vi's)
11    return({f1,...,fn});
12  else
13    return("no function found");
  }
```

図7　R_cから論理関数を抽出するアルゴリズム

6. 実験結果

我々は、ここで提案した手法をUCBの論理合成システムSISの上に実装した。表1に実験結果を示す。MCNCの論理合成用ベンチマーク回路の1つであるmisex1に対し、まず、script.ruggedという回路合成手続きで最適化し、それをXILINXの表参照型フィールドプログラマブルゲートアレイ用の回路に変換した。これを初期回路とし、新しい仕様は、初期回路の論理をランダムに変更することにより作成した。これらの新しい仕様を7種類作成し、それぞれ、misex1_1, ..., misex1_7とした。

circuit	script.ruggedで最適化した回路			script.ruggedで最適化しない回路		
	result	time(sec)	修正されたLUTの数	result	time(sec)	修正されたLUTの数
misex1_1	failure	6.3	-	success	1.3	6
misex1_2	failure	6.5	-	failure	6.3	-
misex1_3	success	4.7	5	success	3.7	5
misex1_4	success	4.4	5	success	3.8	5
misex1_5	failure	5.9	-	success	1.3	6
misex1_6	failure	7.2	-	success	3.7	5
misex1_7	failure	5.7	-	failure	5.5	-

表1　実験結果

そして、初期回路中の万能セルの論理関数を修正することのみで、これらの新しい仕様を満たすように自動修正できるか否かを実験した。これらの7種類の新しい仕様の内、成功したのは、2回のみであった。表1の右側に示された実験結果は、初期回路として、script.ruggedを使用せずそのままXILINXの表参照型フィールドプログラマブルゲートアレイ用の回路に変換したものを初期回路として場合である。この場合には、7回の内、5回で成功している。

script.ruggedは回路を強力に最適化し、冗長性を削除する処理であり、この実験結果から、元の初期回路が十分最適化されていると、仕様が変化した場合には、修正できる範囲が小さくなってしまうことが分かる。

実行例として、図1の回路に対し、図3と同じ仕様変更を行った場合の結果を図8に示す（修正に成功している）。論理を修正するLUTの決め方が回路の段数順のため、図8は図3とは異なった形になっている。

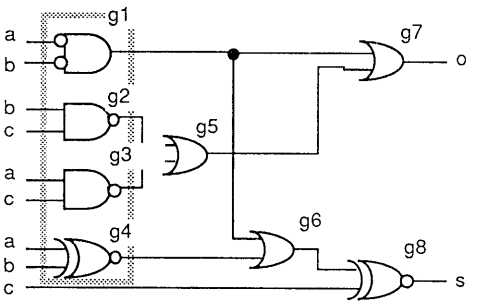

図8　図1の回路変更の例3

7. 結論

表参照型フィールドプログラマブルゲートアレイに対し、万能セルの論理関数を修正するのみで、回路構造を一切変更しないという条件の元で、新しい仕様に合うように元の設計を自動的に修正する手法について述べた。回路構造は一切変化しないため、配置・配線後の遅延は元の回路から変化せず、元の回路遅延をそのまま利用して新しい回路を使用することができる。ここでは、この問題をBoolean Relationに制限を加えたもので定式化し、それを解くアルゴリズムを示した。現在のアルゴリズムは、解が存在する場合でも必ずしも解を見つけられない可能性があるため、今後は、解が存在する場合には必ず見つける手法について研究していく予定である。

参考文献

[1] K. S. Brace, R. L. Rudell, and R. E. Bryant. Efficient implementation of a BDD package. In *Proceedings of 27th ACM/IEEE Design Automation Conference*, pages 40–45, June 1990.

[2] R. K. Brayton and F. Somenzi. Boolean relations and the incompletely specification of logic networks. In *Proceedings of International Conference on Very Large Scale Integration*, pages 231–240, August 1989.

[3] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[4] E. Cerny and M. A. Marin. An approach to unified methodology of combinational switching circuits. *IEEE Transactions on Computers*, C-26(8):745–756, August 1977.

[5] J. Cong, A. Kahng, K.-C. Chen, and P. Trajmar. Graph based FPGA technology mapping for delay optimization. In *Proceedings of ACM Workshop on Field-Programmable Gate Arrays*, February 1992.

[6] M. Damiani and G. De Micheli. Recurrence equations and the optimization of synchronous logic circuits. In *Proceedings of 29th ACM/IEEE Design Automation Conference*, pages 556–561, June 1992.

[7] M. Damiani and G. De Micheli. Synthesis and optimization of synchronous logic circuits from recurrence equations. In *Proceedings of the European Conference on Design Automation(EDAC92)*, pages 226–231, March 1992.

[8] D. Filo, J. C.-Y. Yang, F. Mailhot, and G. De Micheli. Technology mapping for a two-output RAM-based field programmable gate array. In *Proceedings of European Conference on Design Automation*, pages 534–538, February 1991.

[9] R. Francis, J. Rose, and Z. Vranesic. Chortle-crf: Fast technology mapping for lookup table-based FPGAs. In *Proceedings of 28th ACM/IEEE Design Automation Conference*, pages 227–233, June 1991.

[10] R. J. Francis, J. Rose, and K. Chung. Chortle: A technology mapping program for lookup table-based field programmable gate arrays. In *Proceedings of 27th ACM/IEEE Design Automation Conference*, pages 613–619, June 1990.

[11] R. J. Francis, J. Rose, and Z. Vranesic. Technology mapping of lookup table-based FPGAs for performance. In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 568–571, November 1991.

[12] M. Fujita, T. Kakuda, and Y. Matsunaga. Redesign and automatic error correction of combinational circuits. In *Proceedings of the IFIP TC10/WG10.5 Workshop on Logic and Architecture Synthesis*, pages 253–262. North Holland, May 1990.

[13] M. Fujita and Y. Matsunaga. Multi-level logic minimization based on minimal support and its application to the minimization of look-up table type FPGAs. In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 560–563, November 1991.

[14] M. Fujita, Y. Tamiya, Y. Kukimoto, and K.C. Chen. Application of Boolean unification to combinational logic synthesis. In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 510–513, November 1991.

[15] L. Guerra and Y. Watanabe. Local transformations on multi-output nodes using Boolean relations. EE290ls project report, UC Berkeley, May 1992.

[16] K. Karplus. Xmap: a technology mapper for table-lookup field-programmable gate arrays. In *Proceedings of 28th ACM/IEEE Design Automation Conference*, pages 240–243, June 1991.

[17] Y. Kukimoto and M. Fujita. Reduction of critical path delay by optimizing Boolean relations. In *Proceedings of ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems: Tau92*, March 1992.

[18] R. Murgai, Y. Nishizaki, N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli. Logic synthesis for programmable gate arrays. In *Proceedings of 27th ACM/IEEE Design Automation Conference*, pages 620–625, June 1990.

[19] R. Murgai, N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli. Improved logic synthesis algorithms for table look up architectures. In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 564–567, November 1991.

[20] R. Murgai, N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli. Performance directed synthesis for table look up programmable gate arrays. In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 572–575, November 1991.

[21] H. Savoj and R. K. Brayton. Observability relations and observability don't cares. In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 518–521, November 1991.

[22] F. Somenzi and R. K. Brayton. An exact minimizer for Boolean relations. In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 316–319, November 1989.

[23] Y. Watanabe and R. K. Brayton. Heuristic minimization of Boolean relations. In *Proceedings of MCNC International Workshop on Logic Synthesis*, May 1991.

[24] Y. Watanabe and R. K. Brayton. Incremental synthesis for engineering changes. In *Proceedings of MCNC International Workshop on Logic Synthesis*, May 1991.

[25] N.-S. Woo. A heuristic method for FPGA technology mapping based on the edge visibility. In *Proceedings of 28th ACM/IEEE Design Automation Conference*, pages 248–251, June 1991.