設 計 自 動 化 85－16
（1997. 10. 29）

ブロック統合を用いたFPGA回路の最小化

幸田武範 上林彌彦

京都大学工学研究科
〒606-01 京都市左京区吉田本町
TEL:075-753-5395、FAX:075-753-4971
{kouda,yahiko}@kuis.kyoto-u.ac.jp

FPGAには種々の実現法はあるが、一般には k を実現できる関数の入力変数の最大値とすると、1つのブロックで実現できる関数は、(1)k 入力関数を1つと関連する $h(<k)$ 入力関数を実現できる場合（最大入力数論理）と、(2)独立な $h(<k)$ 入力関数を実現できる場合（非最大入力数論理）の2通りとなる。一般には、できる限り最大入力数論理で実現し、そうならない場合には複数の関数を1つの論理ブロックに対応させることにより、論理ブロック数を減少させることができる。本稿では、許容関数を用いることにより上記の(1)や(2)の性質をうまく利用して、より高水準のブロック統合を行い、さらに論理ブロック数を減少させる手法を示す。特に、従来ほとんど設計に利用されてこなかった(1)の性質を考慮した関数分解による手法を提案している。また、論理関数の変更には強力なエラー補償手続きを用いている。最後に、本手法をMCNCベンチマーク回路に適用した結果を示し、本手法の有用性を検証した。

論理最適化、FPGAs、トランスダクション法、CSPFs、SPFDs、エラー補償

Minimization of FPGA Circuits Utilizing Block Integration

Takenori Kouda　　Yahiko Kambayashi

Department of Information Science, Kyoto University, Japan
Kyoto-shi Kamigyou-ku, 606-01, Japan
TEL:075-753-5395、FAX:075-753-4971
{kouda,yahiko}@kuis.kyoto-u.ac.jp

Although there are several implementation methods for FPGA blocks, logic functions realized by typical FPGA blocks are as follows, where k is the maximum number of input variables of the functions realized by one FPGA block. (1) Max-input logic: Realization of one k-input function and related $h(<k)$-input function(s). (2) Non-max-input logic: Realization of two (or more) independent $h(<k)$-iput functions. A basic method to realize FPGA circuit with minimum number of blocks is as follows. (a) Decompose the given functions to functions with k or few input variables. (b) For each k-input function assign one FPGA block (case(1)). For two (or more) $h(<k)$-input functions assign one FPGA block (case(2)). Existing FPGA blocks can be merged using (1) and (2), which is called block integration. A powerful block integration is realized by permissible functions. Especially to utilize the multiple output capability of case(1) which is not used in previous FPGA design, a specific logic decomposition is used in our methods. Error compensation method is used to change logic function because of its powerful conversion capability. The effectiveness of the proposed methods are shown by MCNC benchmark.

Logic Optimization, FPGAs, Transduction Method, CSPFs, SPFDs, Error Compensation

1 はじめに

現在よく用いられているFPGA(Field Programmable Gate Arrays)[6]では、1つのブロックの中に複数個の論理セルが入っており、その組み合わせでブロックの論理が実現されている。従って、1つのブロックで実現される論理関数の入力数の最大値をkとすると、ブロックで実現可能な論理としては、次のようなものがある。

(a) 最大入力数論理：k入力関数を1つと関連する$h(h < k)$入力関数を実現

(b) 非最大入力数論理：独立な$h(h < k)$入力関数を複数個実現

本稿では、このような性質を使って、1つのFPGAブロックによって複数個の関数を実現することにより、FPGAブロック数を最小化する設計手法について検討する。基本的には、トランスダクション法に基づいて、すでに設計されている回路のブロック数を減少させることによって設計を行う。論理回路が与えられた場合、k入力以下の論理関数単位による関数に変換し、k入力関数を1つのブロックで実現し、複数個の$h(< k)$入力関数を1つのブロックで実現するのが基本的な設計法となる。複数の論理関数を1つのブロックで実現するためには、組み合わせによっては配線が長くなることがあり、適切な組合わせを選ぶ必要がある。この場合に、できる限りk入力関数を増加させると結線数や段数の点で有利である。

許容関数[1][2][10]の概念を用いると、ブロックで実現している関数を変更でき、これによって複数のブロックを一つに統合できる場合がある。特に上記(a)の場合、k入力関数と同時に実現される$h(< k)$入力関数は独立でないため、従来手法ではこの性質を有効に利用してこなかったと言える。k入力関数H'が、1変数関数とh入力関数G'の2つを入力とする2入力関数に分解できる時には、H'とG'を同時に1つのブロックで実現できることになる。提案手法においては、このような特殊な関数分解を用いることで、高水準なブロック統合を実現している。その際に、回路内に矛盾を生じさせることなく処理を行うために、強力な回路変形能力を持っているエラー補償手続き[12]を利用している。尚、ここでは、現時点でのFPGAの主な利用法であるプロトタイプ回路作成時に要求される高速性を持ちつつブロック数を最小化することを目的としているため、ブロック数減少に結び付かないようなセル数の最小化は行わないことで、計算時間を短縮させている。

以下、2章では基本的事項を説明し、3章ではエラー補償手続き、4章ではブロック統合と関数分解について述べる。さらに5章ではブロック統合を用いたFPGA回路面積最小化手法についての提案を行い、6章ではベンチマーク回路に対する実験結果及び考察を述べる。最後に7章で結論を述べる。

2 FPGAモデルと諸概念

2.1 FPGAモデル

現在、種々のFPGAが多数の企業によって開発・販売されている。これらのFPGAは実現する論理を後でプログラム可能であるという点についてのみ共通であり、その他の部分は大きく異なることが多い。しかし現在FPGAと呼ばれるものの大半が、高い論理表現力をもつ表参照型FPGAであることから、本稿でも表参照型FPGAを対象と見なすことにする。

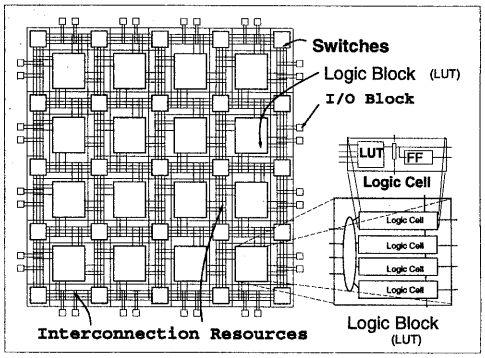

図1: 表参照型FPGA

一般に表参照型FPGAは、図1に示すように、LUTと呼ばれるロジックジェネレーターとフリップフロップからなる論理セルを複数結合した論理ブロックと、それらの間を張り巡らされた内部結線によって構成されている。各論理セルで実現できる論理関数の最大入力数hの値と一つの論理ブロック内の論理セル数、論理ブロックの実現できる入力数などはFPGA製造メーカーによりまちまちではあるが、シェア上位2社であるXILINX,Alteraの製品について見てみた場合、共に$h = 4$で、論理ブロック内の論理セル数はXILINXが4個（XC5000シリーズ）、Alteraが8個（FLEX10Kシリーズ、FLEX8000シリーズ）なっている。尚、XILINX社のXC4000シリーズとそれ以前のモデルの場合、論理セルという形式は採用していないものの、近似的に論理セル2つから論理ブロックが構成されていると言っても差し支えない。どのシリーズも、各々の論理セルで互いに独立なh入力以下の関数を実現する機能とともに、2つ以上の論理セルを内部で結合し、論理ブロック全体としてhより大き入力数kの論理関数を実現できる機能を持っている。

本稿で述べるFPGA回路面積最小化手法は、これらのFPGAの論理ブロックの特徴を利用したものであるが、本稿では話の簡単化のために、これらのFPGAの中でも代表的な存在であるXILINX社のXC4000シリーズを対象として考えることにする。しかし前述のように、他のFPGAもXC4000シリーズと同様の構造・機能を持っているため、そ

れらへの一般化は容易である。

図2は、XC4000シリーズの論理ブロックの構造を簡単に示したものである。

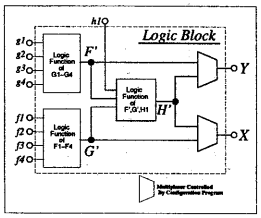

図 2: XC4000シリーズの論理ブロック

本シリーズの論理ブロックは、h=4である3つのLUTを利用し、2つの独立な4入力関数F',G'か、一つの5入力関数H'を実現できる。さらにブロック出力X,Yは、マルチプレクサによりG'又はH', F'又はH'を出力する仕様となっている。

本稿で提案するブロック統合で用いる関数分解は、5入力関数を実現する際に本来ならば必要とされるF'とG'のいずれか一方を使わずに5変数関数を表現することで、使用しなかったLUTを他の4変数以下の関数を実現できるようにし、論理ブロックの利用効率を向上させるものである。

2.2 論理関数の表現形式

一般に、論理ブロック内で実現される論理は、論理ブロックの入力関数の積和形で表現されている。そのため、本稿で提案されている種々の処理を理解するためには、回路入出力とブロック入出力の関係を明確にしておく必要がある。

以後、回路入力$X_1,\cdots,X_n$ (n: 回路入力数), 回路出力$Z_1,\cdots,Z_m$ (m: 回路出力数)に対し、ブロック入力$x1,\cdots,x_i (i < 5)$とブロック出力zの関係は、図3のようになるものとみなし、特に指定がない限り、実際の論理ブロック内の論理セルの数などは考慮せず、一つのブロックで一つの関数を実現するものとする。

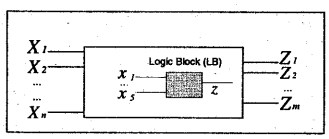

図 3: 回路入出力とブロック入出力

一般に、論理関数を表現するのに回路入力又はブロック入力の積和形からなる論理式を用いることが多いが、本稿では説明を簡単化するために回路入力に対する真理値表(表1)値のベクトル表現を用いて表現することにする。よって、それぞれx_1=(00010101), x_2=(00101101), z=(11010111)と表現される。尚、n=3,i=2、ブロックの内部論理を$z=x_1+\overline{x_1}\cdot\overline{x_2}$とした。

表 1: 回路入力に対する真理値表

X_1	X_2	X_3	x_1	x_2	z
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	1	1	1

X_1,X_2,X_3: 回路入力
x_1,x_2: ブロック入力
z: ブロック出力

また、FPGA内の関数は、その真理値表内の値に不完全指定の要素 $*$ (don't care) を含むこともあり、その際は(0 $*$ 11 $*$ 010)という形で表現する。尚、設計手法実装の際には保持と計算に多くのメモリ量を必要とする真理値表表現は用いておらず、BDDによる表現を用いている。

2.3 CSPFs

CSPFsは論理最適化手法の一手法であるトランスダクション法[2]で、回路の冗長性を表す概念である。ここでは、この概念について、簡単な説明を行う。

あるLUTの出力fをf'に変更しても回路出力に変化がない場合、f'を許容関数(PF: Permissible Function)という。また、許容関数は1つだけでなく複数考えられ、それらの最大集合をMSPF (Maximum set of PF)と呼ぶ。さらに、MSPFの部分集合で回路内の全LUTに対して同時に変更可能な許容関数のみからなる集合をCSPFs(Compatible Set of PFs)と呼ぶ。CSPFsは回路内の冗長性を表す概念として用いられる。

例えば、図4に示す論理ブロックの入力のCSPFは、論理ブロックをAND,ORなどの基本論理素子からなる仮想的な組み合わせ回路とみなし、ブロック出力から各ブロック入力に向かって、各素子のCSPFsを計算していくことで得ることができる。計算方法の詳細は、文献[1][2]に示されている。

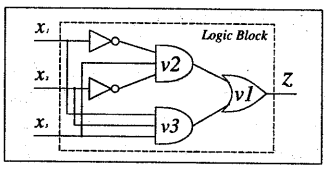

図 4: 論理ブロックの一例

2.4 SPFDs

CSPFsは一般的な組み合わせ回路の冗長性を表現する手法としては非常に有用であるが、FPGAの場合は内部論理を変更できるため、現段階の内部論理から得られた仮想的な組み合わせ回路を用い

表 2: ブロック入力 x_3 の SPFDs

x_1	x_2	x_3	$CSPF(z)$
0	0	0	0
1	1	1	1
0	0	1	1
1	1	0	0
0	1	1	0
1	0	1	0
1	1	1	1
1	0	1	*

→

x_1	x_2	$CSPF(z)$	$SPFD(x_3)$
0	**0**	**0**	$\overline{A}$
1	1	1	B
0	**0**	**1**	A
1	1	0	$\overline{B}$
0	1	0	*
1	0	0	*
1	1	1	B
1	0	*	*

x_1, x_2, x_3: ブロック入力、$CSPF(z)$: ブロック出力の CSPFs、$SPFD(x_3)$: x_3 の SPFDs

て求めた CSPF では、回路の冗長性を十分に表現することはできない。そこで NTT の山下らは、論理ブロックの入出力の依存関係に注目し、回路出力に影響を与えないという条件下でのブロック入力の変更可能範囲を効率よく表現できる SPFDs(Set of Pairs of Functions to be Distinguished) の提案している。本節では、SPFDs の計算法について直感的な説明を行う。尚、計算方法の詳細は文献 [10] に示されている。

前節の図 4の回路について、ブロック入力 x_3 の SPFDs を求める。まず、図 2に示すように真理値表内から x_3 を取り除く。その結果、太字のブロック入力組 (x_1,x_2)=(0,0) に対応するブロック出力の CSPF の値が 0,1 の二値を取ることになり、x_1,x_2 だけでは決定できない。そこで、x_3 で上記の入力組に対応する $CSPF(z)$ の値が 1 になるものと 0 になるものを区別する必要があり、SPFDs はそれぞれ $CSPF(z)$=1 に対応する項が A、0 に対応する項が $\overline{A}$ となる。ここで、A の値は 0 又は 1 の任意の値で、$\overline{A}$ は A の値の逆値である。ブロック入力組 (x_1,x_2)=(1,1) についても同様で、$CSPF(z)$ の値が 1 である項を B、0 である項を $\overline{B}$ とする。

尚、下線付きの入力組 (x_1,x_2)=(1,0) のように、x_3 が存在しなくても $CSPF(z)$ の値一意に決定できる入力の組み合わせに対応する SPFDs の値は、* となる。また、$CSPF(z)$=* である入力組に対応する項は、いかなる場合であっても * となる。

3 エラー補償手続き

FPGA 向きエラー補償手続きは文献 [12] に示された手続きで、図 5のように、配置不能結線が存在した場合や回路出力・ブロック出力で実現する関数の仕様が変更された時に生じるエラーを、内部論理の変更や入力結線の削除・追加、入力関数の変更などの手段を用いて修正するものである。

エラー補償手続きは、エラー補償のの方法により以下の 3 種類に分類することができる。

1 内部論理補償手続き (BLM):
出力矛盾入力組が存在しない場合に、内部論理の変更のみでエラーを修正。

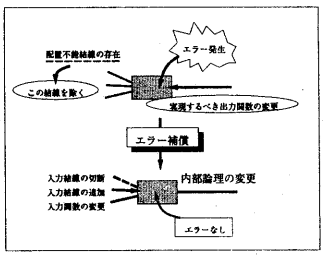

図 5: エラー補償手続き

2 入力補償手続き (IC):
BLM では補償不可能なエラーに対して、ブロック入力の論理修正を行い、エラーを修正。

3 接続変換手続き (SC):
BLM では補償不可能なエラーに対して、入力結線の交換と内部論理の変更を併用し、エラーを修正。

4 関数分解とブロック統合

4.1 ブロック統合 (Block Integration:BI)

一般的な FPGA の論理ブロックは、そのブロックで実現可能な論理関数の最大入力数を k とした場合、内部の論理セルごとに別々な論理関数を割り当て、ブロック全体として複数の $h(< k)$ 入力関数又は、各論理セルで実現した関数を結合用の LUT や内部結線・Wired 論理などによってブロック内部で演算することで実現される k 入力関数を実現することができる。本章では、FPGA の論理ブロックのもつ上記の特徴を考慮し、これらの論理関数を効率よく配置することで、回路の面積を最小化する手法について提案する。尚、ここでは、対象としては 2 章でも述べたように XILINX 社の XC4000 シリーズを考えるが、他のシリーズへの応用は容易である。

XC4000 シリーズの論理ブロックは図 2に示す構造になっており、近似的には h=4 の論理セルを 2 つもった構造と見なすことができる。そのため、ブロック出力としては、図 6(a) に示すようなお互いに独立な 4 入力関数を二つ出力することができる。また図 6(b) に示すように、各々の論理セルの出力

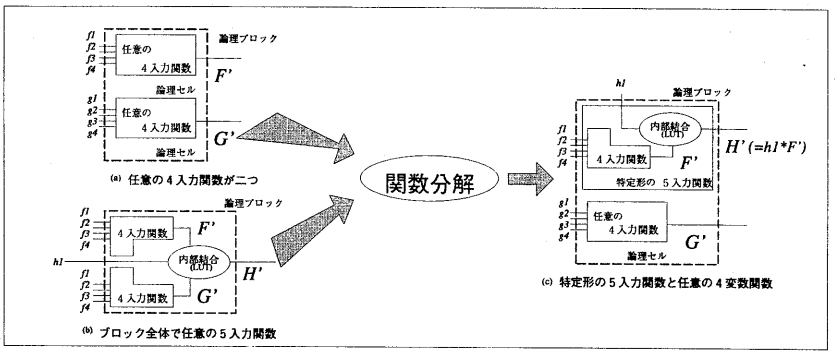

図 6: 関数分解とブロック統合

とは別に、スイッチによってこれらの2つの論理セル出力ともう一本のブロック入力の内部結合（本シリーズの場合はLUTを利用）によって、任意の5入力関数を一つ実現することも可能である。そのため多くのCADシステムは、配置配線の際の自由度が上がる上に、そのまま他社のFPGAにもデータの転用が容易な前者か、回路段数や結線数を小さく抑えることができる後者のどちらかを選択してマッピングを行っている。しかし、FPGA自体の機能としては、5入力関数を実現しているブロックにおいても、スイッチの設定次第で、内部で論理セルの実現する4入力関数も他のブロックの入力として、利用することができる。しかし、5入力関数の一部となっている4入力関数を他のブロックから利用することは容易ではないため、上記の特徴は従来のCADシステムではほとんど利用されていない。我々は、5入力関数を特定形の論理関数（図6(c)では$H'(=h1*F')$）に変形すれば、4変数関数（例えばG'）とはお互い独立に一つの論理ブロック内で実現可能という点に注目し、5入力関数に特殊な関数分解を適用することで、分解前と同様の論理を実現しつつ統合可能な論理関数の数を増やすことを実現した。

4.2 ブロック統合のための関数分解

一般に、5入力関数で1変数関数と4入力関数を入力とする2入力関数で表現できるものは、4入力以下の関数と統合可能である。図2及び図6内の記号H'（ブロック入力$f1 \sim f4$,$h1$からなる5入力関数），F'（ブロック入力$f1 \sim f4$からなる4入力関数），G'（ブロック入力$g1 \sim g4$からなる4入力関数）を用いて示した場合、具体的にはH'=$h1 \cdot F'$などが統合可能な5入力関数である。仮に、$h1$とF'の組み合わせのみを考えた場合でも、16種類の積和形が考えられるが、実際には$h1 \cdot F'$と$h1 = \cdot\overline{F'}$はF'が任意の論理関数であることから同一のものと見なすことができる。そのため、実際に考慮するべき5入力関数の条件としては、以下のようなものになる。

[統合可能な5入力関数]

条件1 $H' = h1 \cdot F'$を満すH'

条件2 $H' = \overline{h1} \cdot F'$を満す$H'$

条件3 $H' = h1 + F'$を満すH'

条件4 $H' = \overline{h1} + F'$を満すH'

条件5 $H' = h1 \cdot F' + \overline{h1} \cdot \overline{F'}$を満す$H'$

実際には、F'の代りにG'を用いた関数も考えられるが、上記のように見なしても一般性を失わないので、以後は上記の5種類のみについて考えることにする。

H'がこれらの条件のいずれかを満す場合、4入力関数G'を5入力関数H'と共に同一論理ブロック内で実現することができる。尚、この場合、これらの2関数に依存関係がある必要はない。実際にCADシステムが生成するFPGAデータにおいては、これらの条件を満すものはほとんど存在しない。そこで、我々は以下の三種類の方法で、5入力関数を条件を満す関数に変形し、統合可能な5入力関数の数を増加させている。

a. 入力端子の割り当て変更
　単純な入力結線の変更。$h1$に割り当てるブロック入力を現在の入力から残る他の入力に変更することで統合可能となる条件を満す入力端子の割り当てを求める。

b. F'の論理変更
　ブロック入力$f1 \sim f4$からなる4入力関数の論理を変更することで、$h1$,H'を変更せずとも上記の条件を満す形にするというもの。

c. $h1$につながる入力結線の交換
　5番目のブロック入力である$h1$につながる結線を他のブロック出力からの結線と交換する。

これらの処理は、5入力関数を最終的なブロック出力を変更することなく、4入力関数と1変数関数に分離するものである。

一般に、前述のb.を行うために必要な理想的なF'の値であるWFの算出は容易ではない。提案手法においては、以下の手順でWFの計算を行い、関数分解を実現している。

[F' の理想値 WF の算出手順]

1. 以下のいずれかを満す関数 WF を計算する。
   - 恒偽又は恒真
   - WF=$h1 \cdot H' + \overline{h1} \cdot \overline{H'}$
2. エラー補償手続きを用いて、4 入力関数を実現している論理セル（LUT）で関数 WF の実現を試みる。
3. エラー補償に成功すれば、F' と WF を交換し終了。失敗なら WF の実現は失敗。

また c. については、対象ブロックの入力からの深さ（入力レベル）が小さいものを対象にした手法と、$h1$ の SPFDs に含まれた関数を対象にした手法が考えられる。最小化能力を優先する場合は、全ての可能性を検証できる前者を、速度を優先する場合は交換しても回路出力に矛盾を起こすことのない関数をすばやく見つけることができる後者を選択する。

4.3 冗長結線削除

5 入力関数を統合可能な形にする方法は、関数分解だけではない。テクノロジーマッパーの生成する論理ブロックの中には、冗長な変数を持つものも少なくない。

5 入力関数を実現する論理ブロックの各入力について、SPFDs を求め、回路入力に対する真理値表の全ての項が $*$ であった入力については、冗長であると考えられるため、回路内から削除することが可能である。また、すべての項が $*$ でなかった場合でも、当該入力を削除の上で矛盾が生じた部分をエラーとし、エラー補償手続きを適用することで、4 入力関数化を試みる。エラー補償に成功した場合、当該入力を回路内から削除可能となる。それらの結果、当該ブロックで実現していた関数は 4 入力関数となり、他の関数と統合可能になる。

5 FPGA 回路面積最小化手法

本章では、前章で提案したブロック統合に基づく FPGA 回路最小化手法について述べる。本手法は、関数分解や冗長結線の削除などを用いて、5 入力関数を他の 4 変数関数と統合可能な特定形に変形するものである。以下に本手法の流れを示す。尚、記号等については、前章でもちいたものと同一とし、4 入力関数 F',G' の入力が $f1 \sim f4$,$g1 \sim g4$、5 入力関数 H' は $f1 \sim f4$,$h1$ からなる論理関数とする。

[FPGA 回路 最小化手法の流れ]

Step.1 回路内の 5 入力関数の順序付け

Step.2 順序に従い、未処理の 5 入力関数がある限り、*Step.2.1* へ

Step.2.1 エラー補償手続き（冗長入力の除去）
→成功：*Step.2* へ。次の関数について。
→失敗：次の *Step* へ。

Step.2.2 関数分解（a. 入力割り当ての変更）
→成功：*Step.2* へ。次の関数について。
→失敗：次の *Step* へ。

Step.2.3 関数分解（b. 4 入力関数 F' の変形）
→成功：*Step.2* へ。次の関数について。
→失敗：次の *Step* へ。

Step.2.4 関数分解（c. 入力 $h1$ の交換）
→成功：*Step.2* へ。次の関数について。
→失敗：次の *Step* へ。

Step.3 当該関数での処理に失敗。*Step.2* へ。

6 実験結果

前章で示した FPGA 回路最小化手法を C 言語を用いて作成し、提案手法の有用性を検証した。実装の際には、論理関数を計算機で効率よく表現するために、現 NTT の湊真一氏による SBDD パッケージ [3] を用いた。実験は SUN 社の Ultra 1 (167Mhz) 上で行なった。

初期回路としては、MCNC ベンチマーク回路 [4] に対してテクノロジーマッパ SIS[5] を用い、下記のコマンド列により生成される入力数 5 の論理ブロックにマッピングした回路を用いている.

[初期回路生成のための SIS コマンド列]

```
sis1> eliminate 2
sis2> gkx -ac
sis3> simplify -d
sis4> xl_part_coll -m -g 2
sis5> xl_cover -e 30 -u 200
sis6> xl_coll_ck -k
```

表 3に、提案手法適用の結果と他手法適用結果を示す。

表中の各欄内の数字は、それぞれ順に論理セル数 / 論理ブロック数 / 結線数 / 回路段数 又は論理セル数 / 論理ブロック数 / 結線数 / 回路段数 /CPU 時間 (sec.) を表している。CPU 時間の項が'0' であるものは、全ての処理が 1 秒以内に終了したケースを意味する。

「提案手法（速度優先）」の列は、仮想分解における結線交換の際に、SPFDs を用いる速度優先手法に対する実験結果を示す。また「提案手法（面積優先）」の列は、仮想分解における結線交換の際にすべての接続可能性を調べることで、時間がかかっても面積を小さくすることができる面積優先手法に対する実験結果を示す。

提案手法との比較のために、本実験の初期回路生成法に入力数 4 の制限を加えることで生成した 4 入力ブロックにマッピングした回路データ（表中では「比較手法 (4)」と表現）と、文献 [12] に示されたエラー補償手続きを用いた FPGA 回路最適化手法を適用した時の回路データ（表中では「比較手法 (opt)」と表現）を同じ表内に並べて示してある。

表 3: 提案手法を適用した結果と他手法との比較

回路名	初期回路 (5) Ce/B/Co/L	提案手法 (速度優先) Ce/B/Co/L/T	提案手法 (面積優先) Ce/B/Co/L/T	比較手法 (4) Ce/B/Co/L	比較手法 (opt) Ce/B/Co/L/T
C1908	82/103/429/13	60/103/429/13/15	53/103/418/13/4576	76/151/549/13	73/98/395/13/3504
C432	50/66/275/17	33/66/273/16/14	33/66/272/16/1254	37/73/268/18	49/66/266/16/1721
alu2	93/109/482/19	65/108/467/19/4	62/108/450/19/49	71/142/510/23	80/101/423/18/274
alu4	158/207/858/24	113/207/837/23/10	112/207/826/23/194	133/266/946/24	143/199/795/23/534
apex7	49/73/292/6	36/72/280/6/0	35/70/272/6/19	42/84/299/9	40/67/249/6/48
b9	30/38/159/4	23/38/159/4/1	19/38/155/4/0	22/43/150/4	30/38/159/4/24
c8	22/30/123/3	16/30/123/3/0	16/30/123/3/10	27/53/183/6	22/30/123/3/9
cht	23/37/157/2	19/37/157/2/0	19/37/157/2/4	19/38/149/2	23/37/157/2/7
cmb	10/12/50/5	7/12/50/5/0	7/12/50/5/1	8/15/54/3	10/12/50/5/1
cordic	14/17/76/8	10/17/76/8/1	10/17/76/8/7	11/21/73/7	13/16/71/8/19
count	25/31/141/5	19/31/141/5/2	16/31/136/5/7	20/39/125/9	23/31/131/5/33
cu	14/19/75/4	10/19/75/4/0	10/19/75/4/1	12/23/82/4	14/19/75/4/2
dalu	266/328/1380/17	185/327/1346/17/50	184/326/1325/17/2538	220/440/1602/20	235/301/1219/15/2427
decod	16/16/80/1	16/16/80/1/0	16/16/80/1/0	9/18/68/2	16/16/80/1/1
exp2	85/105/449/5	56/102/428/5/3	52/102/414/5/103	59/117/387/6	78/101/418/5/126
f51m	21/24/106/5	17/24/104/5/0	16/24/102/5/2	28/55/199/7	19/24/102/4/15
frg1	28/33/148/9	23/33/148/9/57	23/33/148/9/1623	30/60/225/21	28/33/148/9/603
i1	11/18/50/3	9/18/50/3/0	9/18/50/3/1	10/20/56/4	9/18/50/3/1
i2	53/56/267/10	50/56/267/10/1	50/56/267/10/143	37/74/291/9	53/56/267/10/118
i3	29/38/164/7	20/38/164/7/0	20/38/164/7/33	25/50/176/9	29/38/164/7/39
i4	47/50/240/6	44/50/240/6/8	44/50/240/6/294	39/78/272/8	47/50/240/6/328
i5	65/66/324/5	55/66/324/5/1	33/66/324/5/371	33/66/198/9	50/66/264/5/445
i6	53/67/307/1	53/67/307/1/12	53/67/307/1/70	53/106/386/2	53/67/307/1/70
i7	102/103/511/2	100/103/511/2/29	100/103/511/2/308	84/167/631/2	102/103/511/2/254
i8	213/279/1236/8	179/279/1233/8/54	179/279/1230/8/1662	192/384/1425/10	210/279/1218/8/825
i9	135/138/679/5	131/138/679/5/16	124/138/672/5/272	104/207/759/7	131/138/657/5/1564
lal	27/36/142/4	18/35/137/4/0	18/35/129/4/3	22/44/156/4	19/30/103/3/15
pcle	11/16/65/3	10/16/65/3/0	10/16/65/3/1	10/20/64/5	11/16/65/3/3
pcler8	19/26/94/4	13/26/94/4/0	13/26/93/4/4	15/29/90/6	19/26/94/4/6
pm1	11/18/62/2	8/18/62/2/0	8/18/62/2/1	11/21/64/4	11/18/62/2/2
sct	13/21/77/3	11/21/77/3/0	11/21/75/3/1	13/26/89/4	12/20/72/3/5
ttt2	45/53/237/4	28/50/217/4/0	27/50/210/4/11	39/77/275/7	33/43/187/4/29
unreg	24/32/128/2	16/32/128/2/0	16/32/128/2/4	16/32/112/2	24/32/128/2/5
vda	200/246/1043/8	156/246/1043/8/2	141/246/1028/8/68	184/367/1322/8	199/246/1038/8/264
x2	10/13/56/3	7/13/55/3/0	7/13/55/3/0	8/16/56/4	10/13/53/3/4
x3	181/205/938/6	145/202/915/6/6	119/198/873/6/1964	139/277/989/8	162/189/845/5/2826
z4ml	4/5/19/2	3/5/19/2/0	3/5/19/2/0	5/9/29/3	4/5/19/2/0

Ce: 論理セル数 B: 論理ブロック数 Co: 内部結線数 L: 回路レベル (段数) T:CPU 時間 (sec.)
※ CPU 時間の項が'0' と表記してあるところは、計算するのに 1 秒かからなかった個所である。

尚、「比較(opt)」におけるCPU時間は、最適化にかかった時間である。

これらの実験結果から、提案手法は5入力ブロックにマッピングした時の特徴である小さい段数と少ない結線数と共に、4入力ブロックにマッピングした時の少ないブロック数を実現できることが示された。特に、速度優先手法においては、他の最適化手法や面積優先手法には結線数を始めいくつかの項目について劣るものの、かなり高速にある程度の結果を得られることがわかった。本手法が、多くの計算時間を必要とするが効果の高い製品回路の最適化ではなく、速度を要求されるプロトタイプ作成用途での利用を目的としていることからも、速度優先手法でも十分要求を満たせれるものと思われる。また、面積優先手法についても、ほとんどの回路について、回路最適化手法と同等かそれの数分の一の時間で、同等の面積最小化を実現できることが示された。

7 結論 及び 今後の課題

本稿で提案した手法は、ブロックで最大入力数論理を実現した場合には利用されることのない当該論理を構成する内部の論理関数を利用することでブロック数を減少する手法である。本手法においては、最大入力数論理に対して、入力端子の割り当て変更や内部関数の論理変更などの特殊な関数分解を適用することで、独立な他の論理関数を容易に統合できるようにしている。また同時に、エラー補償手続きを用いた冗長入力の削除も併用し、可能な限り多くの関数のブロック統合を実現している。その結果、回路最適化手法との比較しても、高速に十分なブロック数減少効果を得られることがわかった。また実験結果から、仮想分解の際の結線交換の際に、速度優先か面積優先かを選択することで、速度を優先したいプロトタイプ用途にも面積の最小化を優先する製品回路生成の際の利用にも十分有用であることが示された。

今後は、統合不可能な5入力関数内の2つの4入力関数をそのまま利用する手法について、さらなる検討していきたいと考えている。

謝辞

有益な助言をしていただいたイリノイ大学の室賀教授、及び上林研究室の皆様に感謝します。

参考文献

[1] Y.Kambayashi, H.C.Lai, J,N,Culliney, S.Muroga, "NOR Network Transduction Based on Error-Com pensation (Principles of NOR Network Transduction Programs NETTRA-E1,NETTRA-E2 and NET TRA-E3)",Report No. UIUCDCS-R-75-737,Dept. of Comp.Sci. Univ.of Illinois(June 1975).

[2] S.Muroga,Y.Kambayashi,H.C.Lai,J.N.Culliney, "The Transduction Method-Design of Logic Networks Based on Permissible Functions",IEEE Trans.Comput., pp.356-359,Nov.1989

[3] S.Minato,N.Ishiura,S.Yajima, "Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation", Proceedings of 27th Design Automation Conference(1990), 52-57.

[4] S.Yang,"Logic Synthsis and Optimization Benchmarks User Guide Version 3.0", in 1991 MCNC International Workshop on Logic Synthsis (1991).

[5] M.E.Sentovich,K.Singh and etal,"SIS: A system for sequential circuit synthesis",Memorandom No. UCB/ERL M92/41 (1992).

[6] S.D.Brown,R.J.Francis,J.Rose,Z.G.Vranesic, "FIELD PROGRAMMABLE GATE ARRAYS",Kluwer Academic Publishers,1992

[7] M.Fujita,Y.Kukimoto,"Patching Method for Look-Up Table Type FPGA's", Proceedings of Int. Conf. Comput. Aided Design,pp.54-61,Nov.1992.

[8] P.Sawkar,D.Thomas."Area and Delay mapping for Table-Look-Up Based Field Programmable Gate Arrays", Proceedings of 29th Design Automation Conference,pp368-373,Jun.1992.

[9] S.C.Chang,K.T.Cheng,N.S.Woo,M.M.Sadowska, "Layout Driven Logic Synthesis for FPGAs",Proceedings of 31st Design Automation Conference,pp308-312,Jun.1994.

[10] S.Yamashita,H.Sawada,A.Nagoya, "A New Method tp Express Functional Permissibilities for LUT based FPGAs and its Applications", In Proc. of IEEE/ACM ICCAD'96, pp.254-pp.261, Nov.1996

[11] 山下茂, 上林彌彦, 室賀三郎," 許容関数に基づいた表参照型FPGAの最適化手法", 電子情報通信学会論文誌,D-1,Vol.J78-D-1,No11,pp.878-885,Nov.1995

[12] 幸田武範, 上林彌彦," エラー補償に基づく表参照型FPGA回路設計手法", 情報処理学会研究報告, 97-DA-83, 83-5, pp.33-40, Feb.1997