設 計 自 動 化 83－5
（1997. 2. 14）

エラー補償に基づく表参照型ＦＰＧＡ回路設計手法

幸田 武範　　上林 彌彦

京都大学大学院工学研究科

あらまし：最近の制御回路においては、センサーから次々に送られてくるデータの値ごとに異なる機能を要求される場合が多い。この機能はソフトウエア的な手法によって実現することも可能であるが、制御回路そのものを得られた値に従い自動的に変更する方が制御回路の速度や信頼性といった面でも望ましい。このような動的に変化する回路を実現するには、ユーザの手によって実現する論理を自在に変更可能な表参照型FPGAを用いればよい。本稿では、このような目的でFPGA回路を設計する際に有用なエラー補償手続きと呼ばれる手続きを三つ提案する。これらの手続きは、回路内に生じたエラーを補償し、回路出力を正しいものにするものである。さらにエラー補償手続きを応用したFPGA回路設計手法として、FPGA回路内の結線削除と生じたエラーの修正を繰り返すことでFPGA回路を最小化する手法についても提案を行っている。最後に、本提案手法をMCNCベンチマーク回路に適用することでその有用性を示す。

Logic Design Methods for LUT-Based FPGAs Using Error Compensation

T. Kouda　　Y. Kambayashi

Department of Information Science, Graduate School of Engineering,
Kyoto University

Abstract: In many kinds of control circuits, it is required to modify logic function realized by these circuits according to the values obtained by sensors. Logic function corresponding to different sensor values are rather similar. We believe that LUT-based FPGAs that can be realized any logic easily by users can be effectively used to realize such circuits. In this paper, we propose three design procedures for FPGA circuits for such purpose. They called "Error Compensation Procedures" work to compensate errors which are occurred in each output of LUTs. Logic Design Methods proposed here are logic optimization methods utilizing these error compensation procedures. By repetition of removing one connection from a circuit and compensating errors occurred in a corresponding circuit, we can generate another FPGA circuit realizing the original functions with less cost Computer experiments on MCNC multi-level benchmark show the effectiveness of the proposed procedures.

1 はじめに

FPGA(Field Programmable Gate Arrays)[6]は近年論理設計の分野で注目を集めている素子で、ユーザの手により自由に実現する論理を変更できるという特徴を持つ。この特徴を生かし回路の試作や検証などの分野において、FPGAは広く使われてきた。しかし動作速度が遅く、設計コストも高いために従来のゲートアレイの代用品としての価値はまだまだ低かった。ところが、近年のFPGA技術の進歩により製品に組み込んで用いることも現実的なものとなってきている。さらに、FPGAを使って構成された回路は従来までの回路と違い、製品化後に実現論理を変更することも可能であるという特徴を持っている。例えば、各地に取り付けられたセンサーから得られたデータ値ごとに異なる機能を実現する制御回路を設計する場合、従来まではソフトウエア的な手法により要求を実現してきた。しかし、FPGA回路を用いれば、そのデータ値ごとに異なる回路を動的に生成し、要求を満たすことが可能となる。このような動的に変化する回路を実現できれば、演算内のループなどの処理に時間のかかる部分の検出やハードウエア化による高速演算計算機の実現、遠隔地の機器のリモート管理やハードウエアバージョンアップなども夢ではない。これらのことを実現するためには、FPGA回路を高速に設計する手法の考案が不可欠である。現在、デバイスレベルでのFPGAの高速化や高機能化、回路設計装置（FPGAプログラミング装置）の高速化は企業単位で行われている。そこで我々は、デバイスレベルでなくもう一つ上の回路設計レベルでの設計手法について着目した。

従来、データ値などを元に制御回路等に対して求められる回路は、大幅な再設計よりは小規模の仕様変更によって実現されることが多い。大幅な再設計は多くの時間と資源を必要とする。これはFPGA回路においても同様で、小規模な仕様変更で元回路から要求回路が実現できることは回路設計においても重要なことであり、動的に変化するFPGA回路を実現するためにも不可欠なことである。

そこで我々は、FPGA回路に対する仕様変更の際に可能な限り元回路の特徴を生かし、要求された回路を実現するための手続きを提案する。この手続きは、元回路で実現されている論理に対して要求回路にて実現されている論理との差異をエラーとみなし、このエラーを修正する様に働くものである。本稿ではこの種の手続きをエラー補償手続きと呼び、以下の3種類の手続きを提案する。尚、本稿では種々のFPGAの中でも最も実現関数の自由度の高い表参照型FPGAを対象としている。

- **内部論理補償手続き (BLM)**：LUT（表参照型FPGAを構成する論理ブロック）で実現されている内部関数を変更し、エラーを修正する。
- **入力補償手続き (IC)**：LUTの内部関数の変更のみでエラーが修正不可能であった場合に、該当LUTの入力関数を変更しエラーを補償する。入力関数の変更は新なエラーとして入力側のLUTに伝搬されていく。
- **接続変換手続き (SC)**：エラーの生じているLUTの入力端子につながる結線を他のLUTからの結線と変更してエラーを修正する。このようなLUTが存在しない場合は、接続候補となっているLUTにて回路出力に影響を与えない範囲で内部論理を変更し接続を行う。

さらに、回路内の結線を削除し故意に回路内にエラーを発生させ、そのエラーをエラー補償手続きを用いて修正することにより、回路サイズや段数を最小化するFPGA回路最適化手法についても提案を行う。本最適化手法に関しては、MCNCベンチマーク回路に対する最適化実験を行ったので、その結果も示す。

2 基本的事項

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。しかしアーキテクチャに注目することで、大まかではあるが以下の4種類のグループに分類可能である。

- 表参照型FPGA
- マルチプレクサ型FPGA
- 階層型FPGA
- ゲート敷き詰め型FPGA

本研究は、これらの中でも最も論理関数の表現能力の高い表参照型FPGAを対象としている。

表参照型FPGAは、LUT (Look Up Table)と呼ばれる任意のk入力関数を実現可能な論理ブロックとLUT間の内部結線によって構成されるゲートアレイである。

一般にk=4～6で、LUTと内部結線はS-RAMを用いてプログラムされる。XILINX社などから発売されるXC4000などがこのタイプのFPGAである。表参照型FPGAの長所としては、他の型のFPGAと比較して実現可能論理関数の幅が広いことがあげられるが、逆に短所としては動作速度と素子面積が大きいことがあげられる。そのため、他型のFPGAよりも論理最適化手法等の設計支援手法に対する要求は大きいものと考えられる。

尚、本稿では表参照型FPGAを対象と考えているが、実際の製品ではなく図1に示された任意の5入力関数を実現可能なLUTと内部結線から構成される表参照型FPGAモデルを対象としている。

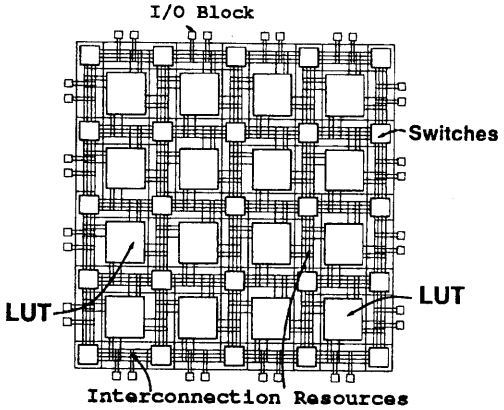

図 1: 表参照型FPGAモデル

2.2 論理関数の表現形式

仮に回路入力を$x_1,\cdots,x_n$ (n: 回路入力数)とした場合、論理関数fを表現するのに回路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の全ての組み合わせに対する真理値表の値のベクトル表現を用いてfを表現している。例えば、$f=\overline{x_1}\cdot x_2 + x_2\cdot\overline{x_3}$については真理値表は表1となり、それぞれ$x_1$=(00001111), x_2=(00110011), x_3=(01010101), f=(00110010)と表現される。

また、fはその真理値表内の値に不完全指定の

表 1: An example of function f

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

要素 * (don't care) を含むこともあり、その際はf=(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。

3 エラー補償手続き

本節では3種類のエラー補償手続きの提案を行う。これらは全てFPGA回路内に生じたエラーについて可能な限り補償を行い、与えられた回路出力を実現しようと試みるものである。

3.1 内部論理補償手続き (BLM)

本手続きの特徴は、回路内に生じたエラーをLUT間の結線を削除又は追加するなどの物理的な変更を行わずにLUT内で実現している論理関数のみを変更して修正することである。

本手続の概略は以下の様なものとなる。

step1. 処理対象LUTを1つの部分回路と見なす。

step2. 入力関数x_i(i=1,$\cdots$)の順序付けを行う。

step3. 要求されている出力関数fに対して入力x_1についてのシャノン展開を行う。
$\rightarrow f = x_1\cdot f_{11} + \overline{x_1}\cdot f_{10}$.

step4. 生成された中間項(例:f_{11}, f_{10})について、0と*のみ、又は1と*のみであった項は展開終了。その他の項はさらに次の入力についてのシャノン展開を行う。未処理の入力がなくなるまでこの操作を繰り返す。

step5. すべての入力に対して処理が終了した際に、すべての項が展開の終了条件を満たしておれば手続き成功。それ以外は失敗。

BLM 手続きに成功した場合、該当 LUT において要求された出力関数はその入力関数の論理式で表現できることになる。よって、展開の結果より求められた入力の積和形で表現される論理式を新に LUT にプログラミングすることにより、該当 LUT に生じたエラーを補償することが可能である。

図 2に BLM 手続き成功例を示す。

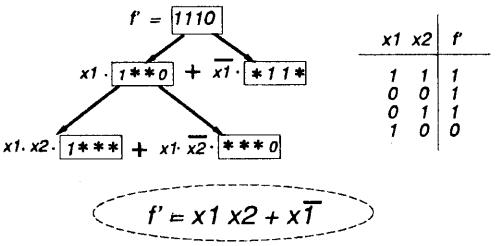

図 2: BLM 手続き成功例

この例では、全ての項が最終的に展開を成功している。その結果、要求された関数 f' を入力 x_1, x_2 の積和形 $x1 \cdot x2 + \overline{x1}$ で表現できることが示されている。

次の図 3は逆に手続きに失敗している例である。

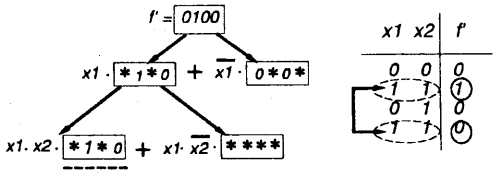

図 3: BLM 手続きによる展開と内部論理計算の例

この例では、最終段の入力 x_2 に関する展開の結果が終了条件を満たしていない。このような場合は手続き失敗であり、与えられた入力関数では要求されている出力を実現できないことになる。この様な場合、さらに強力なエラー補償能力を持つ後述の二つのエラー補償手続きを適用し、与えられた出力を実現しようと試みる。

尚、BLM 手続きにおける展開操作は BDD のグラフ構造と類似しているので、実装の際には BDD の構造を利用することにより高速に処理を行うことが可能である。

3.2 入力補償手続き (IC)

一般に BLM 手続きは不完全指定関数を用いて特定の論理関数を表現する問題と等価である。よって、図 4に示すように各入力の組み合わせについて出力が 0 になるものを OFF-set、出力が 1 になるものを ON-set として入力の組み合わせを分類する。 仮に (1) に示す様に ON-set と OFF-set の両

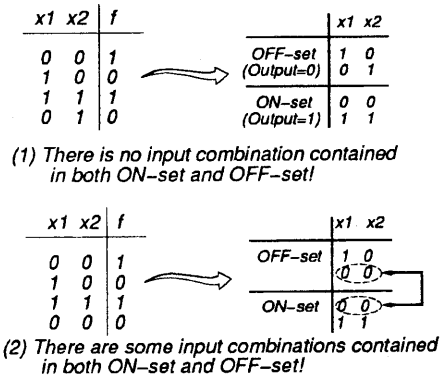

図 4: OFF-set と ON-set

方に含まれる入力の組が存在しない場合は BLM 手続きによって要求関数を実現可能である。しかし、図中の (2) の様に ON-set と OFF-set の両方に含まれる入力の組が存在する場合は BLM 手続きでは要求関数を実現不可能である。このような場合に IC 手続きを用いる。

IC 手続きでは、図 5に示すように入力関数を変更することで、OFF-set と ON-set の両方に含まれる入力の組をなくし展開を可能にしている。

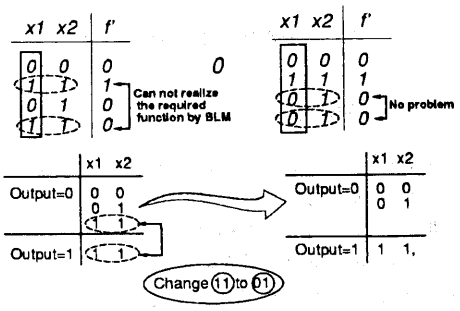

図 5: IC における入力関数変更の例

図 6に IC 手続きの実行例を示す。

入力関数の変更は、処理対象となる LUT の入力端子につながる他の LUT の出力関数を変更するこ

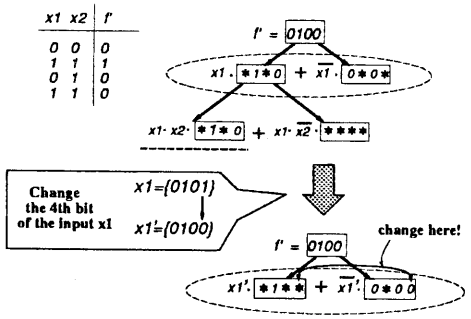

図 6: IC 手続きの実行例

と等価なので容易なものではなく、最悪の場合でも回路出力を変更してしまうことを避けなければならない。

そこでIC手続きにおいては、標準的な論理最適化手法の一つであるトランスダクション法 [1][2] における CSPF の概念を利用することで、これらの問題が生じさせないようにしている。

CSPF：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るので、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 CSPF(Compatible Set of PF) と呼ぶ。一般に、CSPF は回路内の冗長性を表す概念として用いられる。

IC 手続きにおいては、入力候補として計算された x_i' が処理対象 LUT の入力端子につながる LUT の関数の CSPF に含まれるかどうか調べる。もし含まれていた場合は、x_i を x_i' に変更しても回路出力に影響を与えないので入力関数の変更を試みる。そのために、まず入力側 LUT で要求する入力関数を出力として実現可能かどうかを調べる。具体的には、入力端子につながる LUT 内の内部論理の変更のみで新しい関数が実現可能かどうか調べ、不可能であった場合には、その LUT に対して要求される関数と現在の関数の差をエラーとした上で IC 手続きを実行し、要求された関数の実現を試みる。これらの処理はエラーの伝搬が回路入力に達し、新しい入力関数が実現不可能であることが示されるまで再帰的に行われる。

これらの処理の結果、新しい入力関数が実現可能であることが示されたら、入力を変更の上で BLM 手続きと同様の展開を行い新しい論理式を計算する。

本手続きも BLM 手続きと同様に回路内の結線を変更せずに LUT の内部論理のみを変更することによって実現できるため、変更が少なく再利用性の高い動的に変化する回路の設計に適した手法である。

3.3 接続変換手続き (SC)

接続変換 (SC) 手続きは、処理対象 LUT の入力結線を切断することにより FPGA 回路内に生じたエラーを、元の結線の代りに他の LUT の出力を繋ぐことによって修正する手法である。

エラー補償の成功率を向上させるため、本手続きでは前述の 2 種類のエラー補償手続きと同様に各 LUT の内部関数・入力関数の変更も併用する。

SC 手続きの概略は以下の通りである。

1. ある LUT l_a の切断前の入力関数 x_i と同様、または x_i の CSPF $G(x_i)$ に含まれる出力関数を持つ LUT l_b が存在する場合、x_i の代わりに l_b の出力からの結線をつなぎ手続き終了。
2. 1. を満たす LUT が存在しない場合、
   - 2.1. l_i より入力側の LUT について順序付けを行い、その順序に従って LUT l_b を選択する。
   - 2.2. l_a の CSPF$G(l_a)$ と l_b の CSPF $G(l_b)$ について完全指定部分の要素が共通であった場合、$G(l_a) \cdot G(l_b)$ を l_b の新な出力関数 x_i' とし IC 手続きを l_b に対して適用し x_i' が実現可能かどうか調べる。実現可能であることも示されたら、x_i の代わりに l_b の出力からの結線をつなぎ l_a でのエラー補償は終了。
   - 2.3. 2.2 に該当する LUT が存在しなかったら、2.1 に戻り次の LUT について処理を行う。
3. 接続対象となる LUT が存在しなくなったら手続き終了。

本手続きは各 LUT の内部論理のみでなく結線も変更する必要があることで、再プログラムのために前述の 2 つの手続きと比較して多くの時間と資源を必要としてしまう点が短所であるが、エラー補償能力が他と比較して極めて高いことから実用的な手法であるといえる。

4 エラー補償手続きの応用

前述のエラー補償手続きはFPGAの特徴を生かしたものであった。これらの手続きを用いることで、回路再利用を考慮した動的回路生成手法や高速な回路変形手法を実現することが可能となるものと考えられる。本稿ではエラー補償手続きの応用例として、FPGA回路の最適化手法について述べる。本提案手法はFPGA回路のみならず、他のゲートアレイで構成される回路の最適化にも有用である。

4.1 FPGA回路最適化手法

本節では、エラー補償手続きをFPGA回路の最適化に応用した手法について述べる。本手法の概略は以下のようなものとなる。

step 1. LUTの順序付けを行なう。その後で決定した順序$r(l_i)$に従いLUT l_iを選択し、*step 2*へ。もし、選択されていないLUTが存在しなかった場合、このアルゴリズムを終了する。

step 2. LUT l_iについて、CSPF $G(l_i)$を計算する。

step 3. LUT l_iにLUT l_jが、結線c_{ij}で接続されているとする。この時、c_{ij}を切断し、*setp 4*へ。もし、切断可能な結線が存在しなかった場合はl_iでの処理を終了し、*step 1*に戻り次のLUTについて同様の処理を行なう。

step 4. 結線 c_{ji}の削除の結果、LUT $f(l_j)$の出力が変化した場合、その変化後の出力関数がl_iのCSPF $G(l_i)$に含まれていなかったら、出力に対するエラー補償が必要となる。そこでエラー補償手続きを用いてエラーの修正を試みる。もし、エラーが修正された場合は*step 2*に戻り、他の結線で同様の処理を試みる。エラーの修正が不可能だった時は、*step 5*へ。

step 5. 切断した結線 c_{ji}を元の状態に戻し、*step 3*に戻り他の結線で同様の処理を試みる。

本最適化手法は、上記の様に結線の切断とエラー補償を可能な限り繰り返すことで回路の最小化を行なう手法である。その際に、アルゴリズム内の*step 1*におけるLUTの順序付けを工夫することにより、生成回路に特徴を持たすことが可能である。

今回の提案手法実装の際には回路出力からの段数を元に順序付けを行なっており、初期回路と比較して段数や回路面積を特に優先して最適化された回路を生成可能であることが後述の実験結果より示されている。

仮にFPGA回路の動作速度の改善を優先したい場合、動作速度が遅くなる原因であるLUT間の結線での遅延、詳しくは結線を構成するためのS-RAMスイッチ上での遅延を減少させればよく、結線距離を評価し上記のアルゴリズムでの*step 1*におけるLUTの順序付けの際にその評価結果を反映させることで実現できると考えられる。

4.2 FPGA回路最適化実験

前節で示したFPGA回路最適化を行なうプログラムを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]を用い入力数5以下の論理ブロックにマッピングした回路を用いている。以下は初期回路生成のためのSISコマンドである。

```
sis1> xl_split  -n  5
sis2> xl_partition  -n  5
sis3> xlcover
```

表2に、エラー補償手続きとしてBLM手続きを用いた手法、IC手続きを用いた手法、SC手続きを用いた手法の3種類の最適化手法による実験結果の一部を示す。

表2内の太字の部分は、各手法を比較した際に最も結果の良かったものである。また、各欄は順にLUT数／内部結線数／計算に必要としたCPU時間(sec.)を表している。

表2の結果から得られたLUT数及び結線数の平均減少率を表3に示す。

これらの結果から、SC手続きを利用した最適化手法は大変強力な回路最小化能力を持っていることが示された。またBLM手続きやIC手続きは、最適化能力を有しているものの単独で用いるよりは他の手続きと併用することにより力を発揮するものと考えられる。尚、SC手続きが強力な最適化能力を

表 2: 各エラー補償手続きを用いたFPGA回路最適化の結果比較

回路名	初期回路 Lut/結線	BLM Lut/結線/CPU	IC Lut/結線/CPU	SC Lut/結線/CPU
alu2	137/517	135/499/4.9	130/491/162.8	**125/467**/6392.5
alu4	250/1009	248/968/23.1	248/963/423.8	**234/903**/829.8
apex6	376/1005	376/986/20.1	376/986/249.8	**350/883**/1600.3
apex7	124/302	120/265/3.1	113/251/45.3	**112/240**/59.6
dalu	489/1772	462/1588/119.2	464/1590/1621.1	**424/1457**/3610.2
example2	194/413	193/400/3.7	193/400/48.4	**187/378**/69.0
i5	245/459	245/459/3.1	245/459/26.3	245/**447**/677.2
i9	351/1121	351/1002/84.5	350/999/583.2	**346/977**/253.6
sct	61/164	61/164/1.4	59/158/18.5	**54/139**/6.3
term1	171/549	153/470/3.6	147/444/114.3	**98/252**/66.3
ttt2	100/300	94/271/2.2	90/270/64.1	**81/219**/16.8
vda	583/1866	583/1866/9.3	571/1850/115.0	**377/1367**/711.7
x3	379/1073	374/1044/15.3	372/1031/153.6	**350/948**/1385.2
x4	291/721	290/692/10.0	290/688/153.6	**250/564**/128.4

表 3: 各手法ごとのLUT数・結線数平均減少率

手法	LUT	結線	結果のよい回路
BLM	2.0%	5.5%	term1
IC	3.7%	6.7%	term1, apex7
SC	13.0%	16.9%	term1, vda

示した原因の一つに、その内部処理にIC手続きを利用していることで切断された入力結線の代りとなる他のLUT出力を発見できる可能性が大幅に増えていることがあげられる。

次に、本手法と同様にトランスダクション法のCSPFの概念を応用した他のFPGA回路最適化手法[9]との最適化結果の比較を行う。比較対象手法(以下、手法Aと表現)はCSPFを元にLUTのマージなどを中心に回路の最小化を行う手法である。手法Aは内部論理を変更することで最適化能力を向上させた手法であるが、エラー補償の概念を用いていない点で本稿で提案されている最適化手法とは異なる。

表4にSC手続きを利用した最適化手法と手法Aとの比較結果を示す。尚、ここでLUT数・結線数については数そのものの比較でなく、それぞれの平均減少率に着目して比較を行っている。CPU時間については、提案手法がSUN Ultra1で計測したもの、手法AについてはSUN SS10で計測したものである。太字の部分はLUT数・結線数について減少率が極めて結果のよかった個所である。

表 4: 他手法との比較

回路名	提案手法 LB/結線/CPU	手法A LB/結線/CPU
C432	7.8/8.9/637	**10.7/12.6**/68
alu2	8.8/**37.0**/6393	2.8/2.6/4
alu4	6.4/**10.8**/830	2.3/2.4/19
apex7	9.7/**20.5**/60	0.8/1.6/4
cordic	**15.9/32.9**/6	7.3/**10.3**/2
i9	1.4/**12.8**/254	**12.1/15.3**/253
sct	**14.8/20.1**/6	4.6/4.1/2
term1	**43.3/54.5**/66	**13.9/23.3**/21
vda	**35.3/26.7**/712	8.3/9.8/12
	(%/%/sec.)	(%/%/sec.)

表4の結果から、実験を行った回路の大部分についてエラー補償手続きを利用した最適化手法の方がよい結果を示した。しかし、計算時間については手法Aの方が大幅に小さいものとなっており、提案手法の高速化は大きな課題である。

4.3 FPGA 回路最適化手法についての考察

本稿で提案した FPGA 回路最適化手法は、エラー補償手続きを応用した手法の一例である。この手法を実装後実験を行ったところ、強力な最適化能力を持つことが示された。本手法は、エラー補償手続きを利用することによって、単なる回路設計ではなく回路の再利用に適した手法である。そのため、FPGA 回路の新規設計時のみならず仕様変更の際にも力を発揮し、従来までの手法に比べてより低コスト化された回路を高速に生成することが可能であると考えられる。

しかし、計算速度面でのアルゴリズム改良の余地があることと、FPGA のもつ動的な回路生成能力を十分に生かしきれていないことなどが本最適化手法における残された問題である。

5 結論

今回は FPGA という新しいタイプの素子に注目し、今後予想される動的に変化する回路という新しい論理回路の形を考察の上で、そのために必要な設計手法を考案した。

本稿では、回路内の仕様の変更をエラーという形で表現し、そのエラーを修正する手続きである 3 つのエラー補償手続き (内部論理補償手続き、入力補償手続き、接続変換手続き) を提案した。これらの手続きは FPGA の持つ特徴を生かしたもので、可能な限り物理的な配線の変更をせずに内部論理をプログラムにより変更するだけでエラーを修正するという特徴を持っていた。FPGA の可変性を生かすこれらの手続きは、FPGA をもちいた回路設計の可能性を大きく広げるものと考えている。

また本稿においては、エラー補償手続きの利用例として FPGA 回路の最適化手法についても提案を行った。さらに実験により、提案した最適化手法の有用性を検証し、本手法が回路最小化に有用であることも示した。

今後の課題としては、新なエラー補償手続きの考案とそれらを用いた FPGA 回路設計手法についての研究があげられる。

謝辞

SBDD パッケージを使用させていただいた矢島研究室の皆様に深謝します。また、有益な助言をしていただいた上林研究室の皆様に感謝します。尚、本研究は新エネルギー・産業技術総合開発機構 (NEDO) の提案公募型・最先端分野研究開発のプロジェクトとして支援をうけている。

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] 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.

[9] 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

[10] 幸田武範, 上林彌彦," 内部論理補償と入力補償による FPGA 回路設計", 情報処理学会 第 52 回全国大会,Vol 6,1K-10,pp19,Nov.1996