システムＬＳＩ設計技術　108－7
(2003. 1. 28)

論理関数の畳み込み機構を導入した省面積 FPGA の実現と評価

梶原　裕嗣†　　中西　正樹†　　堀山　貴史††　　木村　晋二†††　　渡邊　勝正†

† 奈良先端科学技術大学院大学 情報科学研究科　〒630–0192 奈良県生駒市高山町 8916–5
†† 京都大学 情報学研究科　〒606–8501 京都府京都市左京区吉田本町
††† 早稲田大学 情報生産システム研究科　〒808–0135 福岡県北九州市若松区ひびきの 2–2
E-mail: †{hirots-k,m-naka,watanabe}@is.aist-nara.ac.jp, ††horiyama@i.kyoto-u.ac.jp,
†††shinji_kimura@waseda.jp

あらまし　論理関数の畳み込み機構を導入した新しい省面積 FPGA の構造とその実現手法を提案し，LSI 実現での面積および遅延の評価を示す．配線構造としては，広く用いられている island スタイルに基づいている．複数のベンチマーク回路での評価により，通常の 4-1 LUT と比較して，最大で 32.4%，平均でも 12% の面積削減が可能であることがわかった．

キーワード　FPGA アーキテクチャ，論理関数の畳み込み，ルックアップテーブル

Area Efficient FPGA Architecture with Logic Function Folding

Hirotsugu KAJIHARA†, Masaki NAKANISHI†, Takashi HORIYAMA††, Shinji KIMURA†††, and Katsumasa WATANABE†

† Information Science, Nara Institute of Science and Technology　8916–5 Takayama, 630–0192 Japan
†† Informatics, Kyoto University　Yoshida-Honmachi, 606–8501 Japan
††† Information, Production and systems, Waseda University　2–2 Hibikino, 808–0135 Japan
E-mail: †{hirots-k,m-naka,watanabe}@is.aist-nara.ac.jp, ††horiyama@i.kyoto-u.ac.jp,
†††shinji_kimura@waseda.jp

Abstract　The paper describes an area efficient FPGA architecture based on LUTs with logic function folding. Each LUT is a 3-1 LUT but is enhanced to implement a full adder function with only one LUT. The area of our 3-1 LUT is about 56 % compared to that of a simple 4-1 LUT. In the paper, we measure not only the LUT area but also the area of routing resource. We adopt the well-known island style-architecture for the routing mechanism, and find that the total FPGA area can be saved up to 32.4 % and on average 12 % by the experiments on several benchmark circuits compared to FPGA architecture based on 4-1 LUTs.

Key words　FPGA Architecture, Logic Function Folding, Look Up Table

1. はじめに

回路を設計して，その場ですぐに実現できる FPGA(Field Programmable Gate Array) は，論理回路のプロトタイピングやデバッグなどに広く用いられている．現在，LUT(Look Up Table) を利用した LUT 型の FPGA が多く用いられている [1]～[3]．しかし，FPGA は AND や OR などの基本論理素子に比べると無駄が大きく，アーキテクチャの革新が不可欠である．近年，動的な書き換えを考慮した FPGA [4], [5] や，ALU アレイのように粒度を粗くした FPGA [6], [7] など，アーキテクチャの研究が盛んになりつつある．本稿では，LUT に論理関数の畳み込み機構 [8] を導入した FPGA のアーキテクチャについて述べる．

通常の FPGA の LUT はメモリを基本としており，例えば 4 入力の任意の関数が実現できる 4-1 LUT は，16 ビットのメモリ (レジスタ) と 16-1 の選択回路 (マルチプレクサ,MUX) から構成される．関数への 4 つの入力は選択回路の制御入力となっており，16 ビットのメモリに真理値表を格納することで，4 入力 1 出力の任意の関数を実現できる [9]．

過去，FPGA を構成する LUT に関する研究は数多く行われており [9], [10]，文献 [10] では 4-1 LUT が最も省面積な FPGA であると示されている．市販の FPGA でも 4-1 LUT が使用

されている (Altera 社 FLEX シリーズ [11] や APEX シリーズ [12] など). 我々は 3-1 LUT の構造に論理関数の畳み込み (Folding) の手法を導入した. 畳み込みとは, 実現したい真理値表の部分間の関係に着目し, 真理値表の一部のみを LUT に格納して, 付加的な論理ゲートを用いて全体の真理値を構成する手法である. 提案する LUT では, 全加算器の実現に必要なメモリを 16 ビットから 8 ビットに減らすことができるほか, "多ビットの AND/OR, 等価判定, 大小比較などの実現において Cascade Chain などの専用回路を必要としない", "4 変数以上の論理関数を配線リソースを消費しないで実現可能", といった利点がある. この LUT を FPGA アーキテクチャの基本構造とすることで, 従来の 4-1 LUT を基本構造とした FPGA よりも面積効率の良い FPGA を実現した.

提案する FPGA と 4-1 LUT を基本構造とした FPGA をそれぞれ VDEC ROHM 0.35μm 上で LSI として実装し比較評価を行った. 様々な回路をマッピングした結果, 従来の 4-1LUT の FPGA と比べて最高で 33%, 平均で 12%のマッピング面積の削減が得られた. 8 ビットの加算器を構成する場合の最大遅延パスは 4-1 LUT より 9%短い 2.85ns であった.

2. 一般的な FPGA の構造

本論文では, 論理関数の畳み込み手法を導入した FPGA を公平に評価するために, LUT 以外の構成は一般的なものを採用することにした. この章では一般的な FPGA の構成について述べる.

2.1 LUT の構成

一般に使用されている FPGA では, 4-1 LUT に様々な機能を追加したものが使われている [11], [12]. 図 1 の破線内部に一般的な 4-1 LUT を示す. この 4-1 LUT には二つのモードが用意されている. 一つは通常の 4-1 LUT として任意の 4 変数論理関数を実現できる通常モード. もう一つは算術演算モード (Arithmetic mode) で, Sum と Carry の二つの 3 変数論理関数をそれぞれ 3-1 LUT に格納することで全加算器を実現できる. 二つのモードはモードビットにより制御される. このような加算器のための付加回路は, LUT に基づく論理エレメントの実現において一般的になりつつある [13]. この他にも Cascade Chain と呼ばれる回路が用意されており, 多ビットの AND/OR, 等価判定, 大小比較などの機能を実現する際, LUT 外部の配線リソースを消費することなく実現できる. 本稿ではこのような機能をもった LUT を Altera 型 4-1 LUT と呼ぶ. LUT の出力にはレジスタと選択回路が接続されており, LUT 部分と併せて基本論理エレメント (Basic Logic Element, BLE) と呼ぶ (図 1). 組合せ回路を実現する場合は LUT 出力が, 順序回路を実現する場合はレジスタ出力が BLE の出力となる.

近年の FPGA では, BLE を N 個単位でまとめた構成が一般的である [11], [12], [14]. これを論理クラスタ (Logic Cluster) と呼ぶ. 文献 [14] で紹介されている論理クラスタの構成を図 2 に示す. BLE の各入力は論理クラスタ外部からの I ビットの入力と論理クラスタ内部の N ビットの BLE 出力から選択することができる. 論理クラスタの内部で BLE 出力と BLE 入力のループ経路を持つことで, 論理クラスタの外部入力 I は BLE の総入力数の約 6 割で十分であることが文献 [14] で示されている. これにより論理クラスタ外部の配線リソースを削減することが可能となる.

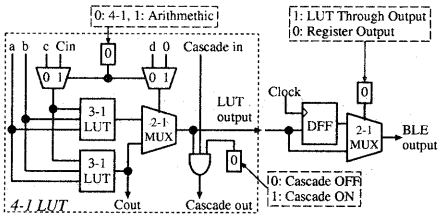

図 1 一般的な 4-1 LUT を含む基本論理エレメント (BLE)
Fig. 1 Plain 4-1 LUT within Basic Logic Element(BLE)

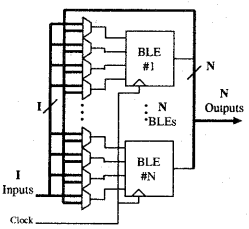

図 2 論理クラスタ
Fig. 2 Logic Cluster

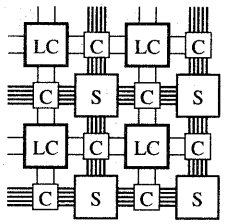

図 3 island スタイル
Fig. 3 Island-style

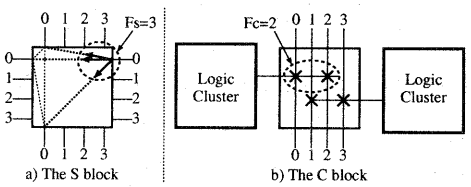

図 4 S ブロックと C ブロックの自由度
Fig. 4 S and C Block Flexiblity

2.2 FPGA 全体の構成

FPGA アーキテクチャとして一般的に知られているものとして, island スタイルが挙げられる (図 3). island スタイルでは, 論理クラスタ (LC) が 2 次元配列状に並び, その間をバスが縦横に配置される. バスの交差点を制御する S ブロック (Swtich blocks) と, 論理クラスタの入出力ポートとバスとの接続を制御する C ブロック (Connection blocks) の二つのブロックにより, 論理クラスタ間の配線が制御される.

island スタイルのマッピング自由度はいくつかのパラメータによって決めることができる. 最も重要なパラメータは外部バスの本数 W である. また論理クラスタを構成する BLE の数は N である. S ブロックの自由度はパラメータ Fs によって与えられる. Fs は S ブロックに入ってきた各トラックがいくつのトラックと接続可能かを決める. 市販されている FPGA では, Fs = 3 であることが多い. W=4,Fs=3 の時のトラック番号 0 に関する配線例を図 4a) に示す. 図からわかるように, 各線は同じ番号のトラックとだけ接続できる. C ブロックの自由

表 1　全加算器の真理値表
Table 1　Truth Table of A Full Adder

C_{in}	a	b	s	C_{out}	C_{in}	a	b	s	C_{out}
0	0	0	0	0	1	0	0	1	0
0	0	1	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	0	1
0	1	1	0	1	1	1	1	1	1

度はパラメータ Fc によって与えられる．Fc は論理クラスタの各入出力ポートが接続可能なトラック数である．図 4b) では W=4,Fc=2 の例を示している．Fc=W のとき各ポートは全てのトラックと接続可能となり自由度は最高になるが，文献 [15] では，Fc=W/N で回路のマッピングに問題がないことが示されている．

3. 畳み込みを導入した LUT の構造

ここでは，加算を実現するための LUT の構造に関し，加算の論理関数の性質から，効率良くより少ないハードウェア (メモリ) で実現する機構を提案する．

3.1 論理関数の畳み込み

3-1 LUT は，真理値表を記憶する 8 ビットのメモリとその選択回路からなる．1 ビットのメモリは十数個のトランジスタから構成されるので，メモリの削減により，LUT の回路全体のハードウェア量を削減できる．そこで，真理値表の部分関数間の関係に着目し，記憶すべき真理値表のサイズを削減する畳み込み (folding) 手法を導入する．

まず，全加算器の真理値表を表 1 に示す．C_{in} が下位からの桁上げ入力で，a, b はその桁の入力を表す．s は和出力で，C_{out} は桁上げ出力を表す．s, C_{out} はともに 3 入力の関数であるので，1 ビットの全加算器は，通常は 3-1 LUT を 2 個用いて実現される．ここで，論理関数値を並べた $f(0,0,0)$, $f(0,0,1)$, ..., $f(1,1,1)$ を関数のベクトル表現と呼び，論理関数と同一視する．s のベクトル表現は 01101001 であり，C_{out} のベクトル表現は 00010111 で表せる．LUT のメモリにはこれらの値が順番に入れられる．

ここで，この真理値表における部分データ間の相関に着目する．まず s の真理値表 01101001 に対し，C_{in} が 0 の場合と 1 の場合で分けて考える．C_{in} が 0 の場合，s は 0110 であり，1 の場合は 1001 である．この分割された真理値表を見比べると，ビット毎の否定の関係が成立していることがわかる．よって，NOT ゲートを導入し，そのまま出力するか，否定の結果を出力するかを C_{in} で選択することで，0110 の部分のみを LUT に記憶するだけで s を実現できる．記憶すべきビット数は半分に削減される．NOT を用いて圧縮しているので，NOT による畳み込み (folding) と呼ぶ．

一方，C_{out} は 0001 0111 である．この関数のみでは単純な論理関数に基づく畳み込みができないので，s のベクトル表現 0110 1001 との相関を考える．今，NOT による畳み込みの結果，s を実現する LUT には 0110 が記憶されている．C_{out} の LUT として前半部 0001 のみを記憶し，後半部の 0111 を s の 0110 と 0001 の論理和で構成することとする．すなわち，OR ゲートを導入し，0001 を出力するか，0110 との論理和の結果の 0111 を出力するかを C_{in} で選択することで，C_{out} を実現する．複数の部分関数から論理和を用いて関数を構成しているので，OR に基づく畳み込み (folding) と呼ぶ．

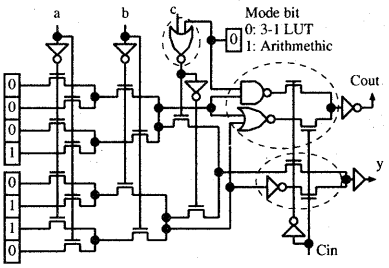

図 5　畳み込みを導入した LUT の構造
Fig. 5　LUT Architecture with Logic Function Folding

表 2　通常モードと算術モードの動作
Table 2　Normal Mode and Arithmetic Mode

Mode	通常 (Mode bit=0)	算術 (Mode bit=1)
C_{in}	0	下位からの桁上げ
C_{out}	0	$\overline{C_{in}}F_U + C_{in}(F_U + F_L)$
y	$f(a,b,c)$	$\overline{C_{in}}F_L + C_{in}\overline{F_L}$

F_U: Upper 4-1 LUT の出力, F_L: Lower 4-1 LUT の出力

3.2 畳み込み機構を導入した LUT

ここでは，前節で説明した畳み込みを導入した LUT のアーキテクチャについて説明する．これは 8 ビットのメモリとその選択回路からなる 3-1 LUT に，NOT に基づく畳み込み回路と OR に基づく畳み込み回路を付加した構成である．また，桁上げ信号を可能な限り高速に伝播させることも考慮している．

畳み込み機構を導入した LUT の構造を図 5 に示す．図からわかるように，a, b, c の通常の入力と，y という通常出力の他に，桁上げ伝播用の C_{in} 入力と C_{out} 出力を持つ．入力 a, b, c と 出力 y に関して通常の 3-1 LUT として使える通常モードの他に，入力 a, b, C_{in} と 出力 y, C_{out} に関して，二つの 2-1 LUT と付加回路で，全加算器などの二つの 3 入力関数を実現する算術演算モードとしても使える．二つのモードは Mode bit によって制御される．通常モード時と算術演算モードの各入出力を表 2 にまとめる．

通常モード (Mode bit=0) で使う場合は，下位からの桁上げ信号 C_{in} は 0 である必要がある．C_{in} が 0 であれば，NAND 回路と出力の NOT ゲートによって C_{out} も常に 0 となる．算術演算モード (Mode bit=1) で使う場合は，C_{in} が 0 の場合は，メモリに記憶されている値が y と C_{out} に出力される．C_{in} が 1 の場合は，y には NOT による畳み込みの結果が，C_{out} には OR による畳み込みの結果が出力される．例えば全加算器として使う場合，C_{in} が 0 の場合は上位の 0001 がそのまま C_{out} に出力され，C_{in} が 1 の場合には 0001 と 0110 の 論理和が出力される．y に関しても C_{in} が 0 の場合はメモリの内容である 0110 が出力され，C_{in} が 1 の場合は 0110 の否定である

1001 が出力される．

実際の FPGA で加算器を実現する場合は，桁上げ伝播がクリティカルパスとなりうるが，C_{in} から C_{out} までは 1 段の選択回路および NOT ゲートしかない．また，C_{in} から y についても同様の構造となっている．桁上げ伝播の遅延については，実際のレイアウトにおいて，ゲートや配線のサイズを含めて最適化を行う必要があるが，回路図上は最適な構造であるといえる．

3.3 回路のマッピングについて

この節では畳み込み機構を導入した LUT の機能が Altera 型 4-1 LUT とほぼ同等であることを示す．Altera 型 4-1 LUT では，4 変数論理関数および 1 ビットの全加算器の二つの機能が実現できる．さらに Cascade Chain と呼ばれる専用回路を用いることで，多ビットの AND/OR や大小比較，等価判定などの演算を，LUT 外部の配線リソースを消費せずに実現を可能としている．

我々の LUT では Cascade Chain などの専用回路なしで多ビットの AND/OR や大小比較，等価判定などの演算を実現可能であることがわかっている [8]．実現に当たっては桁上げ信号と論理和による畳み込みを利用する．例えば等価判定を考える．1 桁分の真理値表を表 3 に示す．1 つの 3-1 LUT で，1 桁分の等価性判定を行うこととし算術演算モードで動作させる．このとき C_{in} を下位の桁からの等価信号とし，C_{out} には現在の桁までの等価判定の結果を出力する．つまり真理値表は，C_{in} が 0 の時には 0000，1 の時には 1001 となる．0000 と 1001 の論理和を取っても 1001 なので，論理和による畳み込みが使え，結果を桁上げ信号として出力できる．大小比較や多入力の AND 演算と OR 演算についても，等価性判定と同様に論理和による畳み込みと桁上げ信号を使うことで実現できる．

Altera 型の 4-1 LUT では 4 変数論理関数を実現できる．一方，畳み込み機構を導入した 3-1 LUT は 3 変数論理関数までしか実現できない．しかし，論理和による畳み込みと桁上げ信号を使うことで，3-1 LUT を 2 個もしくは 3 個使って 4 変数関数が実現できる [8]．その方法は以下のとおりである．3-1 LUT 2 個を図 6 のように使用し，4 変数関数のマッピングを行う．下位の 3-1 LUT は算術演算モードで使用し，上位の 3-1 LUT は通常モードで使用する．下位の LUT は桁上げ伝播用の C_{out} のみを使用し，OR による畳み込みの影響を C_{out} に与えないようにメモリ後半部には 0000 を記憶する．上位の LUT は通常モードで動作させるが，C_{in} の値は 0 ではなく，下位からの C_{out} 出力が入力される．図 6 の回路には入力線が x, y, s, t, u の 5 本あり，出力 y が論理関数 $g(x,y)$, $h(s,t), k(s,t)$ によって表現できた場合，その関数は図 6 の回路にマッピングできることになる．

以上の条件の下，4 変数関数の全探索を行なった結果，65,536 個中 31,848 個 (NPN 同値類 222 個中 109 個) の関数がマッピングできることがわかっている．ほぼ半数の関数が 3-1 LUT 2 個でマッピングできる．桁上げ伝播させる LUT(算術演算モードで使用) を 1 つ増やし，3 個の LUT に対してマッピングを行ったところ，4 変数関数すべてがマッピング可能であった．同様に LUT を増やすことで 5 変数関数以上の論理関数もマッピング可能である．この実現方法は桁上げ伝播用の Carry 線を使うだけで済むので，LUT 外部の配線リソースを消費する必要がなく，効率の良い実現といえる．

表 3 等価判定の真理値表

Table 3 Truth Table of Equality

C_{in}	a	b	=	C_{in}	a	b	=
0	0	0	0	1	0	0	1
0	0	1	0	1	0	1	0
0	1	0	0	1	1	0	0
0	1	1	0	1	1	1	1

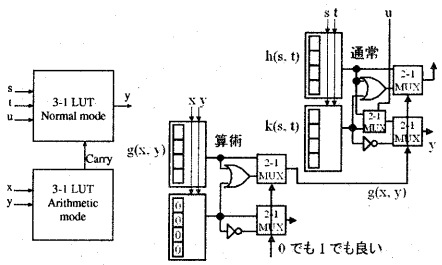

図 6 4 変数関数のマッピング

Fig. 6 Implement 4 Input Functions

表 4 設計に使用したパラメータ

Table 4 Parameters used for a FPGA design

	3-1 LUT	4-1 LUT
N	8	8
I	15	21
W	20	28
Fs	3	3
Fc(Inputs)	4	4
Fc(Outputs)	6	7

4. 実装と評価

畳み込み機構を導入した 3-1 LUT を基本構造とした FPGA と Altera 型 4-1 LUT を基本構造とした FPGA をそれぞれ設計し評価を行った．

4.1 設計詳細

FPGA を構成するアーキテクチャには，2 章で述べた island スタイルを採用した．このときのパラメータは表 4 に示す通りである．論理クラスタを構成する BLE の数 N は 8 とし，論理クラスタの入力ポート数 I は“BLE の入力総数+出力回路の制御線”の 6 割とした．論理クラスタの入出力ポート数の違いに伴いバス幅 W を設定した．S ブロックの自由度を決める Fs は市販の FPGA と同様の 3 とし，C ブロックの自由度を決める Fc については，論理クラスタの入力ポートについては 4，出力ポートについては 3-1 LUT は 6，4-1 LUT は 7 とした．出力側を多くすることで配線制御の自由度を上げている．

論理クラスタ，S ブロック，C ブロックの繰り返しパターンを 2 次元アレイ状に配置することで FPGA 全体が構成される．構成の詳細を図 7 に示す．論理クラスタの出力ポートを全て右辺

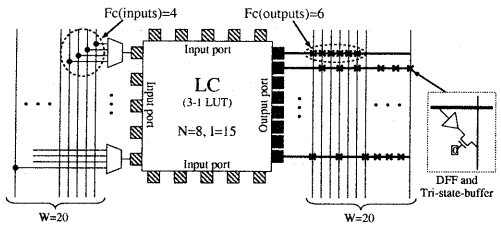

図 7 FPGA の構成図
Fig. 7 FPGA Design

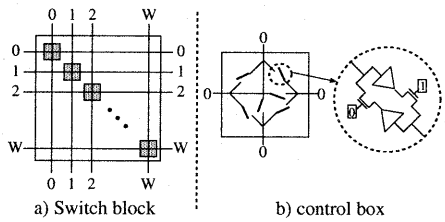

図 8 S ブロックと制御ボックス
Fig. 8 a Switch block and a Control Box

にまとめ，入力ポートを他の 3 辺に均等に配置してある．入力ポートとバスとの接続制御にはマルチプレクサを使用し，4 本のトラックから 1 本を任意に選択できる．出力ポートとトラックとの接続制御には，1 ビットのメモリと 1 個の Tri-state-buffer を用いたクロスバースイッチを使用する．各出力ポートごとに Fc の値だけクロスバースイッチが配置されており，各スイッチは独立して制御できる．論理クラスタと C ブロックの位置関係から，論理クラスタの左右に位置する C ブロック，すなわち出力ポートと入力ポートとで挟まれているものを C ブロック (A) と呼び，論理クラスタの上下に位置する C ブロック，すなわち入力ポート同士で挟まれているものを C ブロック (B) と呼ぶ．

S ブロックの全体の構成を図 8a に示す．Fc=3 という条件から同じトラック番号間との接続のみが許される．各トラックの交差点は制御ボックス (Control Box) と呼ばれる回路で制御される．制御ボックスはメモリと Tri-state-buffer で図 8b のように縦横斜めの計 6 箇所を双方向に制御できる．

以上の条件のもと，畳み込み機構を導入した 3-1 LUT および Altera 型 4-1LUT を基本構造とする FPGA をそれぞれ VDEC の提供する ROHM 0.35μm EXD ライブラリを用いて論理合成およびレイアウトを行った．設計言語には VHDL を使用し，実現される回路を意識して RTL 設計を行った．論理合成には synopsys 社の Design Analyzer を，レイアウトには Avant! Apollo を使用した．論理合成時に全加算器のクリティカルパスになりうる桁上げ伝播を高速にするように遅延制約をかけた．また，なるべく回路面積が小さくなるように面積制約もかけた．

4.2 面積評価

各ブロックの面積を表 5 に示す．各列は左から 3-1LUT の場合の面積，4-1LUT の場合の面積，4-1LUT に対する 3-1LUT の面積比を表している．Total は，FPGA を構成する際の繰り返しパターンである論理クラスタ，S ブロック，C ブロック (A,B) の合計値である．LUT1 個あたりに必要となる面積はそれぞれ 3-1 LUT は 40,977μm^2，4-1LUT は 61,708μm^2 である．3-1LUT と 4-1LUT の面積比は 66.4%である．以下この値を用いて様々な回路をマッピングした時の LUT の使用数と面積について評価を行う．評価の対象は“Altera 型 4-1 LUT の FPGA”，“畳み込み機構を導入した 3-1 LUT の FPGA”，“通常の 3-1 LUT の FPGA”の 3 種である．“通常の 3-1 LUT の FPGA”とは，畳み込み機構を導入した FPGA の LUT 部のみを一般的な 3-1 LUT に入れ替えたものであり，畳み込み機構の有無による回路面積の違いはほぼ無視できるものとして畳み込み機構を導入した FPGA の面積と同等とした．評価の方法としては，Altera 社の 論理合成ツールである QuartusII ver. 1.1 を用いて，同社の FPGA である APEX 20 K 向け (4-1 LUT 向け, [12]) にマッピングを行う．4-1 LUT 向けに論理合成された結果から，提案する 3-1 LUT 向けにマッピングをし直す．このとき，全加算器や 3 変数関数ついては 3-1 LUT 1 個で実現，4 変数関数については 3.3 節で述べたとおり 3-1 LUT 2 個で実現可能なものと 3 個で実現可能なものに分類しマッピングを行った．なお通常の 3-1 LUT の場合は全加算器は LUT 2 個で実現，4 変数関数を LUT 3 つで実現する．

表 5 レイアウト面積
Table 5 Layout area

	our 3-1 LUT	4-1 LUT	面積比 (%)
BLE	6,039	11,088	54.5
Logic cluster	184,041	293,040	62.8
S block	107,291	150,872	71.1
C block(A)	26,384	35,640	74.0
C block(B)	10,098	14,108	71.6
Total	327,814	493,659	66.4
Area required per BLE	40,977	61,708	66.4

(単位：μm^2)

マッピング結果を表 6 に示す．各列は左から“回路名”，“A：4-1 LUT にマッピングした場合の LUT の個数”，“B：畳み込み機構を導入した 3-1 LUT にマッピングした場合の LUT の個数”，“C：4-1 LUT にマッピングした場合に必要となる面積に対する畳み込み機構を導入した 3-1 LUT にマッピングした場合に必要となる面積の比 (表下部の式参照)”，“D：通常の 3-1 LUT にマッピングした場合の LUT の個数”，“E：C と同様に計算された面積比”である．最終行には面積比の平均が示してある．ベンチマークに用いた回路は 16 ビット配列型乗算器，iir フィルタといった回路や，本研究室で設計を行った音声認識回路，視線推定回路，16 ビットパイプライン CPU などである．それぞれ論理合成時に面積制約 (a)，タイミング制約 (s) をかけて合成したものについて評価を行っている．

IIR フィルタや cordic などの算術演算回路については，全加算器の畳み込み効果により面積が大きく削減している．最良で 33%削減である．また平均しても約 12%の削減が期待できる．通常の 3-1 LUT の結果を見てわかるように，3-1 LUT に有利な配線パラメータの設定や設計を行ったわけではないといえる．

表 6 マッピング
Table 6 Mapping

回路名	4-1 A	our 3-1 B	面積比 C*1	plain 3-1 D	面積比 E*2
Multi16x16(a)	381	508	88.7	635	110.8
Multi16x16(s)	381	508	88.7	635	110.8
cordic(a)	751	851	75.4	951	84.2
cordic(s)	893	1,075	80.0	1,257	93.5
iir filter(a)	148	149	66.9	150	67.6
iir filter(s)	148	149	66.9	150	67.6
SpeachRecog(a)	13,527	21,595	106.0	27,829	136.6
SpeachRecog(s)	17,151	25,238	97.7	32,933	127.5
EyeTracking(a)	11,184	14,644	87.0	17,192	102.1
EyeTracking(s)	12,276	16,352	88.5	19,772	107.0
16bitPCPU(a)	1,213	2,096	114.8	2,759	151.0
16bitPCPU(s)	1,593	2,552	106.4	3,505	146.1
rader7(a)	312	354	75.6	392	84.7
rader7(s)	299	340	75.6	381	84.6
Average			88.2		107.6

$*1 : C = \frac{B \times 40977 \times 100}{A \times 61708}$ $*2 : E = \frac{D \times 40977 \times 100}{A \times 61708}$

したがって面積削減効果は畳み込み機構によるものである．

パイプライン CPU などの複雑な制御関数を多く含むものは逆に面積が増えてしまっている．だが，4-1 LUT 向けに論理合成したものを再マッピングしていることを考慮すると，畳み込み機構を導入した 3-1 LUT 専用のテクノロジーマッパーを開発することで，改善が期待できる．

4.3 遅延評価

FPGA の基本構造である BLE の遅延データを表 7 に示す．この表は畳み込み機構を導入した 3-1 LUT と Altera 型 4-1 LUT を使用した各 BLE の入力ポートから出力ポートまでのクリティカルパスの遅延をそれぞれ計測したものである．全加算器を実現する際の最大遅延パスに影響する C_{in} から C_{out} への遅延は $0.22ns$ で，4-1 LUT と比較してもほぼ変わらない値 (差は $0.02ns$) となっている．一方，外部からの入力 (A,B,C) については，4-1 LUT よりも遅延が少ないと言える．これは回路規模が小さく，通過ゲート数が少ないためである．8 ビット加算器を実現した場合のクリティカルパスは，我々の 3-1 LUT は $2.85ns$ で，4-1 LUT は $3.13ns$ で 3-1 LUT のほうが高速である．しかし，ビット数を増やすごとに C_{in}–C_{out} 間の遅延の差 ($0.02ns$) が影響し差は縮まる．20 ビット付近で 3-1 LUT と 4-1 LUT の遅延の優劣は逆転する．

3.3 節で述べた 3-1 LUT 2 個で 4 変数論理関数を実現する場合のクリティカルパスは "$A \rightarrow Cout \rightarrow Cin \rightarrow BLE\ Outputs$" の経路で $1.53ns$ である．3-1 LUT 3 個で実現する場合のクリティカルパスは "$A \rightarrow C_{out} \rightarrow C_{in} \rightarrow C_{out} \rightarrow C_{in} \rightarrow BLE\ Outputs$" で $1.75ns$ である．2 個の場合も 3 個の場合もともに Altera 型 4-1 LUT の $1.94ns$ よりも遅延が少ないので高速に動作する．

5. おわりに

論理関数の畳み込み手法を 3-1 LUT に導入した FPGA を設計した．畳み込み機構の導入により，全加算器を 3-1 LUT 一つで実現できる．様々な回路をマッピングしても従来最も省面積

表 7 BLE の遅延評価

our 3-1 LUT			Altera-type 4-1 LUT		
Input	Output	Delay	Input	Output	Delay
C_{in}	C_{out}	0.22	C_{in}	C_{out}	0.20
A		0.80	A		0.79
B		0.57	B		0.91
C_{in}	BLE $Output$	0.73	C_{in}	BLE $Output$	1.02
A		1.53	A		1.94
B		1.30	B		1.90
C		1.12	C		1.26
(単位：ns)			D		1.06

とされていた 4-1 LUT よりも，マッピング面積が最良で 33%，平均でも 12%削減できる．今後は提案する FPGA 向けのテクノロジーマッパーの開発を行い，さらなる配線パラメータの調整および評価を行う必要がある．

謝辞

日頃から御討論いただく，奈良先端科学技術大学院大学 渡邊研究室の皆様に感謝します．また，マッピングに関して御討論いただく早稲田大学理工学部 柳澤研究室の清水友樹様に感謝します．なお，本研究は一部，文部科学省科学研究費補助金，日立中央研究所受託研究費，NEC 受託研究費による．

文献

[1] *Altera, http://www.altera.co.jp.*

[2] *Xilinx, http://www.xilinx.co.jp.*

[3] D. Buell et.al. *Splash2: FPGAs in a Custom Computing Machine.* IEEE Coputer Science Press, 1996.

[4] 飯田全広, 末吉敏則. リコンフィギュラブル・ロジック向き論理ブロックの提案. 情報処理学会 研究会報告, システム LSI 設計技術 103-23, pp. 141–146, Nov. 2001.

[5] T. Fujii, K. Furuta, M. Motomura, and et. al. A Dynamically Reconfigurable Logic Engine with a Multi-Context/Multi-Mode Unified-Cell Architecture. In *Proc. of International Solid State Circuits Conf.*, p. WA21.3, Feb. 1999.

[6] 越智裕之. FPAA: フィールドプログラムアキュムレータアレイ. 計算機アーキテクチャ, 情処研報, vol.97, no.102, pp.97–102, (Oct 1997)

[7] 山内宗，中谷正吾，犬尾武，梶原信樹. 再構成可能デバイス RHW 向け高速演算器の実装と評価. *THe 14th Workshop on Circuit and Systems in Karuizawa* April 23-24, 2001.

[8] S. Kimura, T. Horiyama, M. Nakanishi, H. Kajihara Folding of Logic Functions ans Its Application to Look Up Table Compaction. *Proc. of ICCAD 2002*, pp694-698, Nov. 2002.

[9] Varghese George and Jan M. Rabaey. *Low Energy FPGA.* Kluer Academic Publishers, 2001.

[10] J. Rose, R. J. Francis, D. Lewis and P. Chow. Architecture of Programmable Gate Arrays: The Effect of Logic Block Functionality of Area Efficiency. *IEEE Journal of Solid state Circuits*, pp1317-1225, oct. 1990.

[11] *1996 Data Book.* Altera Corporation, 1996.

[12] *APEXII Programmable Logic Device Family Data Sheeet.* Altera Corporation, 2001.

[13] Scott Hauck, Matthew M. Hosler, and Thomas W. Fry. High-Performance Carry Chains for FPGA's. *IEEE Trans. on Very Large Scale Integration Systems*, Vol. 8, No. 2, pp. 138–147, April 2000.

[14] V. Betz and J. Rose. Cluster-Based Logic Blocks for FPGAs: Area-Efficiency vs. Input Sharing and Size. *Proc. IEEE Custom Integrated Circuits Conf.*, IEEE Computer Society Press, Los Alamitos, Calif., 1997, pp551-554.

[15] V. Betz and J. Rose. How much Logic Should Go in an FPGA Logic Block?. IEEE Design and Test, 1998.