社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－ＳＬＤＭ－128（16）
2007／1／18

SoC 埋め込み型プログラマブルロジック ePLX の設計アーキテクチャの検討と回路マッピングの評価

菱田 智雄† 石橋 宏太†† 木村 峻†† 奥野 直樹†† 松本 光崇†
中野 裕文‡ 岩男 剛宜‡ 奥野 義弘‡ 有本 和民‡
泉 知論†,†† 藤野 毅†,††

† 立命館大学大学院 理工学研究科 〒525-8577 滋賀県草津市野路東 1-1-1
†† 立命館大学 理工学部 〒525-8577 滋賀県草津市野路東 1-1-1
‡ 株式会社 ルネサス テクノロジ 〒664-0055 兵庫県伊丹市瑞原 4-1

E-mail: † ††{re009018, re000038, re004036, re002037, re010029, t-izumi, fujino}@se.ritsumei.ac.jp,
‡ {nakano.hirofumi, iwao.takenobu, okuno.yoshihiro, arimoto.kazutami}@renesas.com

あらまし 近年、少量生産の SoC（システムオンチップ）製造において、マスク費用やレイアウト・検証設計費用などの初期開発費用の増加が大きな問題になっている。FPGA を製品に使用することも行われているが、FPGA は SoC と比較して性能面で劣っており、１チップあたりの製造コストが高いという問題がある。本論文では、SoC 内の一部にプログラマブルロジック ePLX(embedded Programmable Logic matriX)を配置することを提案する。ePLX を用いることにより、SoC 内の回路ブロックで、アプリケーションや顧客に特有の機能は論理を変更することができる。ePLX アーキテクチャはローカルアーキテクチャ（マトリクス状に配置した２入力 LUT と、マトリクス端に配置した FF）とそれらを接続する階層的な配線リソースを有している点に特徴がある。加算器、乗算器、DES 暗号回路などのサンプル回路を ePLX 上でマッピングした結果から、LUT の使用率を議論する。また、HDL コードから ePLX のコンフィグレーションデータを生成する自動設計フローを紹介し、その一部として開発している自動ツールを用いたローカルアーキテクチャのマッピング結果を報告する。

キーワード プログラマブルデバイス，細粒度，LUT マトリクス

Analysis of design architecture of ePLX (embedded Programmable Logic matriX) and Evaluation of circuit mapping

Tomoo Hishida† Kouta Ishibashi†† Shun Kimura†† Naoki Okuno†† Mitsutaka Matsumoto†
Hirofumi Nakano‡ Takenobu Iwao‡ Yoshihiro Okuno‡ Kazutami Arimoto‡
Tomonori Izumi†,†† Takeshi Fujino†,††

† Graduate school of Science and Engineering, Ritsumeikan University
†† Faculty of Science and Engineering, Ritsumeikan University
1-1-1 Nojihigashi, Kusatsu-shi, Shiga, 525-8577 Japan
‡ Renesas Technology Corp. 4-1 Mizuhara, Itami-shi, Hyogo, 664-0055 Japan

E-mail: † ††{re009018, re000038, re004036, re002037, re010029, t-izumi, fujino}@se.ritsumei.ac.jp,
‡ {nakano.hirofumi, iwao.takenobu, okuno.yoshihiro, arimoto.kazutami}@renesas.com

Abstract Recently, non-recurring engineering costs (NREs), including cost of mask-sets, and engineering design efforts are critical problems in a small-volume SoC(System on a Chip) manufacturing. FPGAs are used for some electrical products, but FPGAs still have lower performance and higher chip-cost than SoC. In this paper, we propose ePLX(embedded Programmable Logic matriX) that is embedded in SoC. Application-specific or customers-specific logic function in SoC can be changed using ePLX. The ePLX architecture is based on the programmable local-clusters, which are composed of two input Look-Up-Table(LUT) matrix and the D-FlipFlops on the matrix side. The hierarchical wiring resources are located between the local-clusters. We demonstrate the ePLX mapping results for sample circuits such as an adder, a multiplier, and a DES encryption circuit, and discuss LUT utilization efficiency. Lastly, we introduce ePLX design flow from HDL code to ePLX configuration data, and experimental results using the mapping tool which is newly-developed for ePLX.

Keyword programmable device, small grain, LUT matrix

1. はじめに

近年、集積回路の微細化・高集積化技術の進歩により、SoC 製造に必要なマスク費用や設計費用などの初期開発費用が増加している。そのため、生涯生産個数が 10 万個程度以下の SoC においては、1 チップあたりの製造コストが高くなりすぎ、ビジネス的に生産をすることが難しいという問題が発生している。

このため、生涯生産個数が少量から中量の SoC においては、FPGA（Field Programmable Gate Array）と呼ばれる、製造後に論理を変更可能なプログラマブルロジックデバイス（PLD）を製品に使用することも開始されている。しかしながら、FPGA は SoC と比較して設計自由度や速度、消費電力などの性能面で劣っているという問題点がある。

上記の問題点を解決するため、マイコンやメモリから構成された汎用的な SoC の一部に、FPGA と同様にプログラム可能な領域を配置する。この領域で SoC 開発時に変更を予定している回路や、品種展開させる回路、また製造後の修正による設計変更が発生する確率が高い回路を作成する。このようにすることで、SoC の低消費電力性や低コスト性、高速動作を維持しながら、製造後も論理の変更が可能となる。

本論文ではプログラム可能な領域に配置するプログラマブルデバイスとして、ePLX（embedded Programmable Logic matriX）を提案する。ePLX のアーキテクチャは、LUT(Look-Up Table)マトリクスと FF、及びそれらを接続する階層的な配線リソースから構成される。このアーキテクチャを評価するために、共通鍵暗号の 1 つである DES 暗号などのサンプル回路のマッピングを試行した。また、現在検討中である ePLX の自動マッピング手法についても述べる。

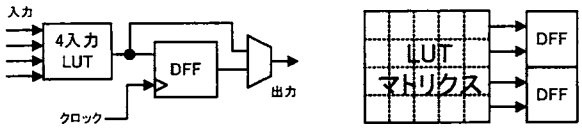
以下、第 2 章では提案する ePLX の設計アーキテクチャについて検討した結果を述べる。第 3 章で手動マッピング手法の提示とマッピング結果の評価を行い、第 4 章で自動マッピング手法について説明する。最後に、第 5 章でまとめと今後の課題を述べる。

2. ePLX アーキテクチャの検討

SoC 埋め込み型プログラマブルロジックである ePLX について、アーキテクチャを検討した。ここではその結果について述べる。

2.1. LUT の入力数について

一般的な LUT 型 FPGA のアーキテクチャは図 1(a)のような構造である。FPGA では入力数 4（あるいはそれ以上）の LUT と FF を LE（ロジックエレメント）として使用している。4 入力の LUT を用いることで、エリアを最小化できることは既に報告されているが[2]、



(a) 一般的な FPGA　　(b)ePLX

図 1. アーキテクチャ比較

5 入力以上または 3 入力以下の論理を構成しようとした場合には LE の使用効率が悪化する。このため、最近では入力・出力数を可変にできるさまざまな LUT の構成が考案されている[3]。

本研究では、図 1(b)のように 4 入力以下の LUT を複数並べた LUT マトリクスに対して複数の FF を配置し、等価的に、様々な入力数の LUT に対応できるようなプログラマブルロジックアーキテクチャの検討を行った。本 LUT の入力数を 2～4 まで変化させた場合の面積効率に関して以下のように評価を行った。

k 入力 LUT 1 つの面積 LA(k)、スイッチ 1 つの面積 SA 、1 つの LUT に必要なスイッチ数 SN(k)、回路を実現するのに必要な LUT 数 LN と定義すると、LUT マトリクスの面積 A は式(1)のようになる。

$$A = \{LA(k) + SA \times SN(k) \times \frac{k}{2}\} \times LN \qquad (1)$$

レイアウトの評価から LA(k)、SA、SN(k)を表 1 の値とした。

表 1. LUT の入力数によるエリアの比

k(入力数)	LA(LUTエリア)	SA(スイッチエリア)	SN(スイッチ数)
2	1	0.1	32
3	2	0.1	48
4	4	0.1	64

いくつかのサンプルにおいて、以下の手順を用いて LUT マトリクスの面積 A を算出した。

(1)2 入力の全論理ゲート（9 種類）と FF から構成されるライブラリで、対象回路を論理合成する。

(2)出力されたネットリストに対して、RASP[4]の DAO マップ[5]を用いて 2、3、4 入力 LUT への論理最適化を行う。

(3)テクノロジーマッピング後に必要な LUT 数 LN を調べ、(1)式を用いて面積 A を算出する。

結果のグラフを図 2 に示す。サンプル回路のソースコードは、OPENCORES[6]のコードから共通鍵暗号回路である AES と DES、誤り訂正符号であるリードソロモン符号のエンコーダー回路を用いた。これらの結果より、LUT の入力数が 2～3 のときに最もエリアが小

さくなることを確認できた。設計フローの簡易化の理由から、本論文ではePLXに2入力LUTを採用する。

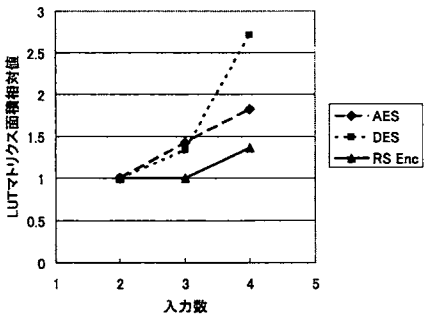


図2. LUTの入力数と面積の相対比較

2.2. ローカルアーキテクチャ

図3にローカルアーキテクチャを示す。2入力LUTをマトリクス状に配置し、その右端にFFを配置した構造になっている。図3に示すように、1つのLUTからはスイッチを経由して13ヶ所のLUTに接続でき、左から右への片流れの構造をとっている。スイッチはON/OFF状態を決定する1bitのSRAMを保持しており、このデータを書き換えることで、配線の接続の切り替えを行う。LUTは、バッファ(BUF)として用いることで配線チャネルとしても使用できる。複数行または複数列先のLUTへの接続を可能にすることで、図4のように1行（または1列）で複数ビットを伝播させることができエリアを効率よく利用できる。さらには、ゲートが密集し配線が困難になることを避けるため、図3のように4行または4列先のLUTにも接続できるようにした。

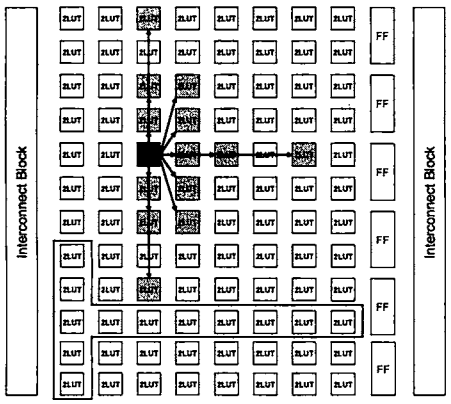


図3. ePLXローカルアーキテクチャ

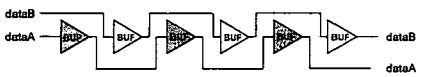
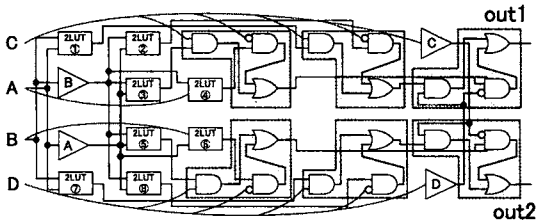
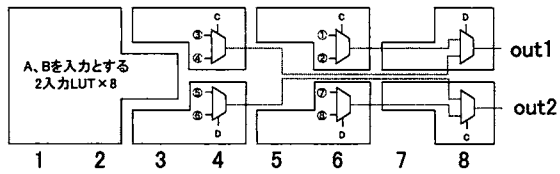


図4. LUTによる配線チャネルの実現法

インターコネクトブロックからは、LUTアレイ1行につき1bitを同一行の8LUTと1列目の上下4LUT(図3中で囲まれている範囲）に接続できる。この構造とローカルアーキテクチャを組み合わせて使うことで、複雑な回路も配置・配線可能になる。例として、図5に4入力2出力のLUTを示す。灰色のエリアは、2入力LUTを用いて構成した2to1MUXを表しており、4行8列のマトリクスで構成可能であった。従来の典型的なFPGAでは、4入力1出力LUTにつき1つFFを配置していた。これと同等になるように考慮して、2行8列のLUTマトリクスにつきFF1つを割り当てるアーキテクチャにした。このアーキテクチャであれば、2.1節で議論したように様々な入力数のLUTに対応することができる。LUTマトリクス8列とFFで構成されたローカルアーキテクチャ領域を、以降はローカルクラスタと呼ぶ。



(a)論理ゲート配置・配線図



(b)簡略イメージ図

図5. 2入力LUTを用いた4入力LUTの構成

2.3. グローバルアーキテクチャ

図6はグローバルアーキテクチャを表している。図3のローカルクラスタとインターコネクトブロックを交互に並べた構造をとっている。このような構造を採用することで、効率よく左右のローカルクラスタ間のデータを伝播することができる。

インターコネクトブロックは、データの中継を行うブロックであり、一般的なFPGAの配線網に相当する。セグメントチャネルというインターコネクトブロック内の上下方向への自由配線（図6①）、及び異なるインターコネクトブロック間の配線（図6②）を行う。セグメントチャネルを用いることでローカルクラスタ内における縦方向の配線性の低さを補うことができる。さらに、ローカルクラスタ間のデータのやりとりだけではなく、ePLXと外部の間のデータの入出力もこの

ブロックで行う。

このブロックを用いることで、ローカルクラスタ内では不可能であった右から左へのフィードバックも可能になる。

これらの配線リソースを効率よく用いローカルクラスタ内の BUF を用いた配線チャネルの使用率を低下させることで、ローカルクラスタ内の論理ゲートの充填率を上昇させ、ePLX のエリアを効率よく使用することができる。

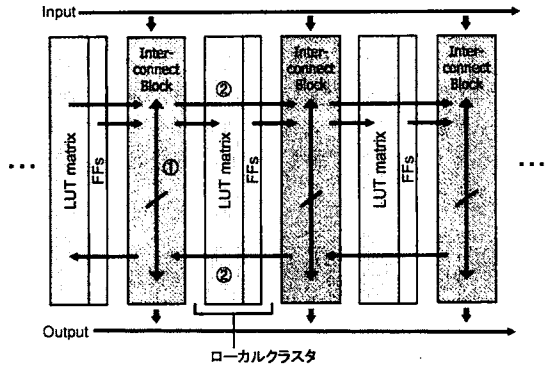


図 6. ePLX グローバルアーキテクチャ

3. ePLX の手動マッピング手法と評価

ePLX のアーキテクチャを評価するために、例として加算器や乗算器、DES 回路のマッピング・評価を行った。

3.1. 加算器、乗算器

小規模な回路の例として、図 7 に 4bit 加算器のマッピング結果を示す。空白 LUT を含む、点線で囲まれた 6LUT のエリアで全加算器を構成している。また、4bit 乗算器も図 7 中の全加算器を規則的に配置することでマッピングすることができ、14 行 8 列の LUT マトリクスに収まり、1 ローカルクラスタ内で実現できる。乗算器の論理ゲート使用率は 70%程度、BUF 使用率 20%程度であり、全体としては 90%程度という高い使用率でマッピング可能であることを確認した。

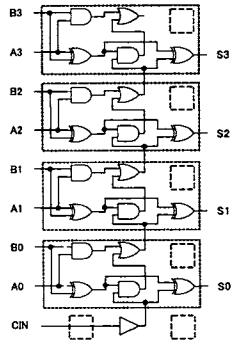


図 7 4bit 加算器のマッピング例

3.2. DES 暗号回路

中規模な回路の例として、共通鍵暗号回路の 1 つである DES 暗号回路のマッピングを試行した。

マッピング回路のサンプルとして、2 章と同様に OPENCORES の DES 暗号回路のソースコードを用いた。ただし、一部の記述に関しては ePLX のアーキテクチャに適するように修正を加えている。図 8 に DES 暗号の主な処理を表したブロック図を示す。

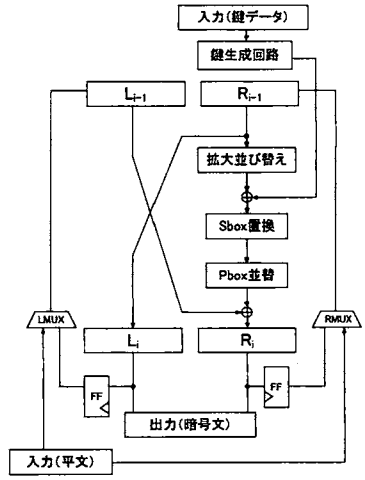


図 8. DES 暗号のブロック図

まず、回路全体を論理合成し、大まかな回路規模を見積もった。DES 回路は図のように処理がブロック毎にわかりやすく分割されている（具体的な処理の内容はここでは省略する)。そこで、先に各ブロックの詳細な配置・配線を行い、最後にインターコネクトブロックを使いブロック間の受け渡しを行う。

マッピングの試行に用いるローカルクラスタサイズは、LUT : 256 行 8 列、FF : 128 行 1 列であり、256 行の LUT マトリクスから構成されるローカルクラスタを 1 アレイとし、このアレイを 6 個使用して DES 暗号のマッピングを行った。DES 暗号のアレイへの割り当て例を図 9 に示す。実際には、アレイ間にインターコネクトブロックが存在し、そのブロックから入出力やフィードバック等を行う。

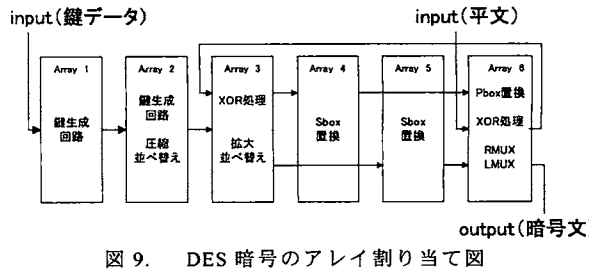


図 9. DES 暗号のアレイ割り当て図

図9のようにマッピングを行ったときのLUT使用状況を表2に示す。アレイは図9中のアレイ番号と対応しており、比較のためDesign Compilerを用いて同じ回路を2入力ライブラリを対象に論理合成した際のゲート数を②に示す。③はePLXに最適化した回路で使用する論理ゲート数、④はLUTをBUFとして使用した数を表しており、最下段にはこれらの合計数を示す。

表2. DES暗号回路のマッピング結果

	鍵生成 Array 1、2	XOR処理 Array 3	Sbox Array 4、5	Pbox他 Array 6	トータル
① 全LUT数	4096	2048	4096	2048	12288
② 論理合成によるゲート数	710	1705		232	2647
③ 論理ゲート数（使用率）	820 (20.0%)	48 (2.3%)	1952 (47.7%)	224 (10.9%)	3044 (24.8%)
④ BUF数（使用率）	812 (19.8%)	508 (24.8%)	718 (17.5%)	216 (10.5%)	2252 (18.3%)
⑤ 使用数（使用率）	1632 (39.8%)	556 (27.1%)	2668 (65.1%)	440 (21.5%)	5296 (43.1%)

Design Compilerの論理合成結果を使用するのではなく、ePLXの配置・配線性を考慮した回路にしたため、ゲート数が若干増加した。本回路ではインターコネクトブロック内の配線リソースが不足したため、図4に示すようにLUTをBUFとして使用し配線チャネルとした。その結果、論理ゲート数に対して約70%程度BUFを使用した。最終的なLUT使用率は50%程度となった。

4. ePLXの自動マッピング手法

3章の結果は、手動によるマッピング結果を示したものであり、自動でのマッピングに関しては現在開発中である。ここでは、ePLXの設計フロー、及びマッピングの自動化フローとツールの現状ついて述べる。

4.1. ePLX全体設計フロー

ePLXの設計フローを図10に示す。まず、ハードウェア記述言語（HDL）で実現したい機能を記述する。次に、HDLコードを入力として、2入力の論理ゲートとFFをライブラリとして論理合成を行う。論理合成結果のネットリストを元に、ePLXのアーキテクチャに従ってマッピングを行い、最後にコンフィグレーションデータを作成する。論理合成にはSynopsys社のDesign Compilerを用い、その他のフローでは新たにツールを作成することで対応する。

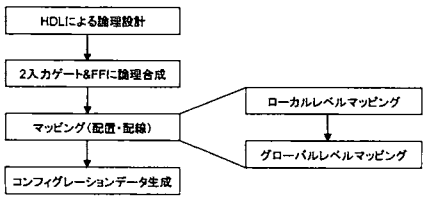


図10. ePLX設計フロー

4.2. ローカルレベルマッピング

図10の中で、マッピングが最も重要なフローである。小規模なモジュールであれば1ローカルクラスタ内で実現することができ、以下の手法でマッピング可能である。

(1)Design Compilerによる論理合成後のネットリストを読み込む。

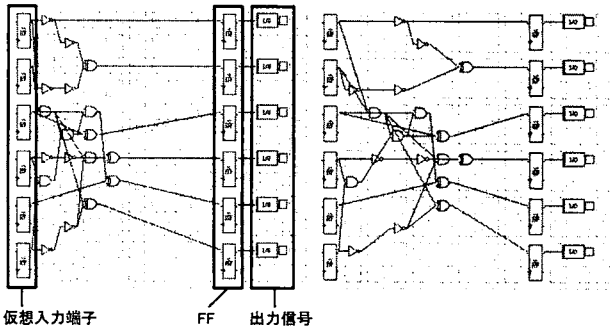
(2)入力信号を左辺に、FF及び出力信号を右辺に配置する。FFの出力信号がフィードバックされる場合には仮想入力端子として左辺にも配置する（後処理として、インターコネクトブロックで接続）。

(3)入力信号を「0」として、入力信号に接続されるゲートを「1」、そのゲートに接続されるゲートを「2」という要領でナンバー付けし、X座標とする。

(4)mincutアルゴリズムを用いて、配線の交差が少なくなるようにY座標を定める。

(5)ゲートが局所的に集中配置され、配線の混雑度が高くなり過ぎるのを防ぐため、ゴムモデルを用いてゲート配置位置を分散させる（図11に、6bitのカウンタ回路に対して本処理を行った例を示す）。

(6)必要に応じて、BUFの挿入やコンパクションを行い、詳細配線を完成させる。



(a)ゴムモデル適用前　　(b)ゴムモデル適用後

図11. 6bitカウンタ回路の自動マッピング例

4.3. グローバルレベルマッピング

多数のモジュールからなる大規模な回路であれば、複数のブロックに分割し、それぞれをローカルレベルでマッピングする必要がある。これらのマッピング結果を結合するグローバルマッピングフローを以下に説明する。

図12にグローバルマッピングフローを示す。まず、結合度の強いゲートを固めて配置できるように、階層を破壊せずに残したまま合成する（図12(1)）。次に、

論理合成を行ったブロック毎にローカルレベルでのマッピングを行う（図 12(2)）。最後に、それらを組み合わせて回路全体の配置・配線を行う（図 12(3)）。

マッピングの結果に問題があれば、配置・配線の再検討を行う。ePLX のエリアを有効に利用するためには、回路全体の再配置の検討は必要になる（図 12(4)）。

回路全体での再配置で問題が解決しなければ、他のブロックとの入出力関係を考慮に入れたローカルレベルでのマッピングのやり直しが必要になる。（図 12(5)）。

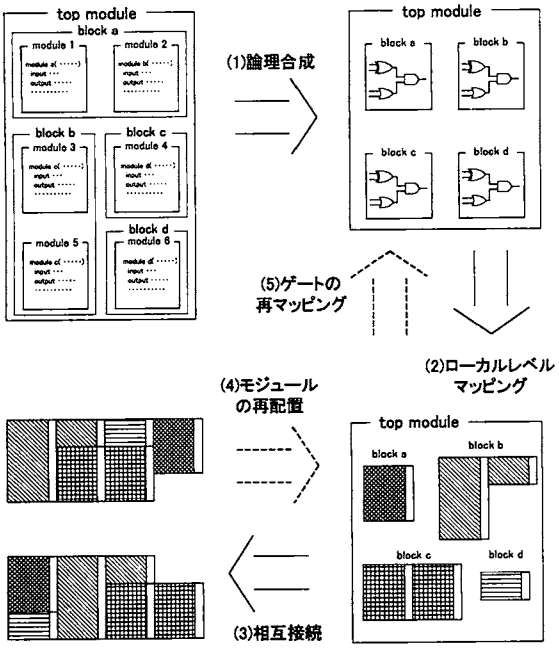


図 12.グローバルレベルの自動マッピングフロー

5. まとめと今後の課題

本論文では、提案している ePLX のアーキテクチャについて述べ、回路マッピングの評価を行った。

ePLX とは、マトリクス状に配置した 2 入力 LUT とマトリクス端に配置した FF を用いたプログラマブルデバイスである。細粒度のマトリクスアレイを用いているため、柔軟度の高いプログラマブルロジックを実現することができる。従来の 4 入力 LUT を考慮して、2 行 8 列の 2 入力 LUT につき FF を 1 つ割り当て、ローカルクラスタとしている。配線に関しては、ローカルクラスタと配線網の役割を担うインターコネクトブロックを交互に並べる構造をとっている。

また、手動によるマッピングの結果、LUT の使用率は乗算器で約 90%、DES 暗号回路で約 50%となった。

マッピングの自動化方法として、ローカル・グローバルそれぞれの設計方針を示した。ローカルレベルの自動マッピングツールを用いて、小規模な回路はマッピングできている。今後は現状のツールの最適化を続けるとともに、残りの自動化フローのツールも順次進める。

文　献

[1] H.Nakano, T.Iwao et al. An Embedded Programmable Logic Matrix(ePLX) for flexible functions on SoC, IEEE ASSCC, pp.219-222, Nov 2006

[2] J. S. Rose, et. al, Architecture of FieldProgrammable Gate Arrays : The effect of Logic Block Functionality on Area Efficiency, IEEE JSCC, pp.1217-1225, October 1990.

[3] 飯田全広、末吉敏則、"リコンフィギャラブル・ロジック向き論理ブロックの提案と評価"、情報処理学会論文誌、Vol.43、No.5、pp1181-1190、2002

[4] J. Cong, J. Peck, Y. Ding, "RASP: A General Logic Synthesis System for SRAM-based FPGAs, ACM Symp. FPGAs, pp137-143, 1996.

[5] D. Chen, J.Cong, DAOmap: A Depth-optimal Area Optimization Mapping Algorithm for FPGA Designs, ICCAD, pp752-759

[6] http://www.opencores.org/

[7] 末吉敏則、天野英晴、リコンフィギャラブルシステム、pp1-63、オーム社、2005

[8] 石畑清、アルゴリズムとデータ構造、pp229-276、岩波書店、第 1 刷　1989.