

■6 群(コンピュータ 基礎理論とハードウェア) - 5 編(コンピュータアーキテクチャ(II) 先進的)**6 章 リコンフィギャラブルコンピュータ**

(執筆者：天野英晴) [2010年4月 受領]

■概要■

リコンフィギャラブルコンピュータは、動作時に、ハードウェア構成を変更することのできるコンピュータであり、リコンフィギャラブルデバイスを用いて構成される。リコンフィギャラブルデバイスで良く用いられているのは、細粒度のSRAM型FPGA(Field Programmable Gate Array)と、粗粒度構成のPE(Processing Element)アレイの二種類である。

前者は、汎用のデジタル素子として広く用いられており、リコンフィギャラブルコンピュータは、この素子を用いてシストリックアルゴリズムやデータ駆動処理など様々なハードウェアアルゴリズムを柔軟に実行する。FPGAの大規模化と高性能化によって最近様々な応用分野での利用が進んでいる。

一方で、粗粒度構成のPEアレイを用いたリコンフィギャラブルデバイスは、動的再構成が可能なものが多く、PEの命令やPE同士の接続を問題を解きながら高速に変更することで、少ない面積と電力で効率の良い処理を行う。SoC(System-on-a-Chip)上のアクセラレータとして用いられ、多くの日本半導体企業が開発を進めている。

【本章の構成】

まず6-1節で、SRAM型FPGAを中心としたリコンフィギャラブルデバイスの構成と最近の動向を紹介する。続いて6-2節で、リコンフィギャラブルデバイス上で実現するハードウェアアルゴリズムを紹介する。6-3節では、主にFPGAを用いて構成する大規模なリコンフィギャラブルシステムの開発例について紹介する。最後に、6-4節で、粗粒度のPEアレイを用いた動的リコンフィギャラブルシステムの動作と構成、開発例について紹介する。

■6群 - 5編 - 6章

6-1 書き換え可能デバイス

(執筆者：末吉敏則・飯田全広) [2009年4月 受領]

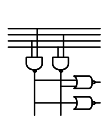
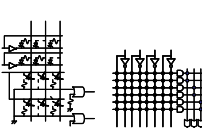

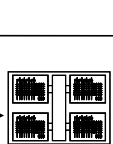
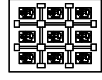
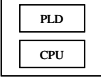
書き換え可能デバイス^{1),2)}は、論理仕様をプログラムすることで様々な論理回路を実現できる集積回路の総称である。ASICのように製造時に機能が確定しているデバイスとは異なり、このデバイスの特徴はユーザが手元で機能を変更することができる点にある。また、ASICが高額なマスク費などの高い開発費と長い開発期間を要するのに対し、書き換え可能デバイスは、デバイス単価は高いものの開発費は低く、開発時間は比較的短い。このような特徴から、このデバイスは多品種少量生産向きデバイスとして規格や仕様が頻繁に変更される用途で大きな市場を獲得している。

本節では書き換え可能デバイスの歴史からデバイスアーキテクチャ、そして部分的な再構成や動作中の再構成を実現できる機能を付加した発展形デバイスについて説明する。

6-1-1 歴史と概要

書き換え可能デバイスの原型は、AND-OR アレイ構造（プロダクトターム方式）をもつ比較的小規模のPLD（Programmable Logic Device）にはじまる。その後、1980年代の後半から1990年代に入ると集積回路技術の進歩に伴い大規模なPLDが作れるようになり、AND-ORアレイ構造を複数組み合わせたCPLD（Complex PLD）がAltera社から発表された。一方、AND-ORアレイ構造ではない新しい方式のPLDとして、SRAMベースのルックアップテーブル（Look Up Table:LUT）を基本論理ブロックとしたFPGA（Field Programmable Gate Array）がXilinx社から発表され、現在のFPGA/CPLD時代の幕開けを迎えた。1990年代以降、集積度の向上によるSRAM（Static Random Access Memory）ベースFPGAの大規模化、メモリや専用演算器、プロセッサの搭載などが進み、急速に性能が向上している。

表6-1 書き換え可能デバイスの歴史（出展：特許庁HP「特許から見たPLDの展望」³⁾）

年代	1970年代前半	1970年代後半	1980年代	1990年代	2000年～	
チップ構造						
最大ゲート規模数		数十～100	数百	数千～数万	数十万～数百万	
代表的デバイス名	MOS Programmable Logic Array	Field Programmable Logic Array (FPLA) Programmable Array Logic (PAL)	Generic Array Logic (GAL)	Complex Programmable Logic Device (CPLD) Field Programmable Gate Array (FPGA)	System on PLD 時代の到来	
技術的特徴	PROMを用いたプログラム可能なランダムロジックデバイス	ユーザの手元でプログラム可能なデバイス ORアレイ固定型により高速化 ヒューズROMにより1回の書き込み可能	CMOSを採用し、低消費電力化 プログラム素子に電氣的読み書き可能なデバイスを採用	複数のAND-ORブロックを有し、高密度化、大容量化、高速化を実現	ロジックエレメント、インタコネク、I/Oセルをプログラマブルとしたロジックアレイの基本アーキテクチャ	エンベデッド・プロセッサの採用 半導体IPの利用 環境の整備
代表的企業	Texas Instrument	Signetics (現Philips) Monolithic Memories Inc (Vanti社を経て 現 Lattice Semiconductor社の傘下)	Lattice Semiconductor	Altera	Xilinx Altera QuickLogic Xilinx	

6-1-2 論理ブロックアーキテクチャ

書き換え可能デバイスは内部に多数のプログラマブルな論理ブロックを含んでおり、それらをプログラマブルな配線を介して接続することで大規模なデジタル回路を実現する。このため、論理ブロックのアーキテクチャは極めて重要である。一般に、 k 入力 LUT は k 本の信号入力をもつ任意の関数を実装できる。入力数の k の値が大きいと大きな論理回路を一つの LUT で実装できるため、所望のデジタル回路の論理段数は少なくなり、結果として速度的に有利になる。しかしながら、入力数 k 未満の回路部分も多く存在し、実装上の面積効率は悪い。一方、 k の値が小さいなら論理段数は多くなり、速度的には不利だが面積効率はよい。このように LUT の入力数にはトレードオフが存在する。

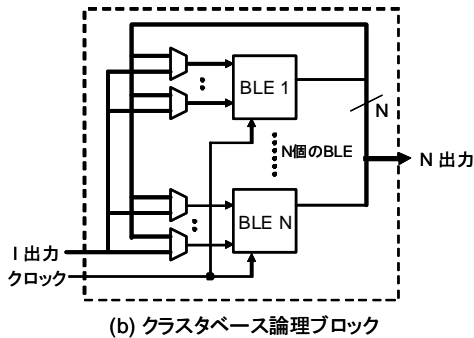
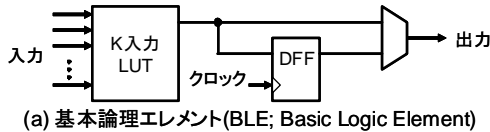


図 6・1 クラスタベース FPGA の論理ブロックの構成

最近の FPGA の論理ブロックは、単一の LUT ではなく複数の LUT とフリップフロップをまとめて大きな論理ブロックにグループ化している。これを論理ブロッククラスタという。

図 6・1 はクラスタベース FPGA の論理ブロックの構成を示している。クラスタベースの論理ブロックでは、クラスタ内で局所的な接続で論理ブロックが接続されている。この局所的な接続は論理ブロック外部の配線構造での接続より高速であるため、回路全体の遅延改善に効果的である。また、クラスタ内に配線を多く取り込むことで外部の配線数を少なくでき、配線リソースの削減と配置配線時間の短縮につながる。

6-1-3 配線アーキテクチャ

書き換え可能デバイスの代表的な配線構造は、1 次元配線方式、2 次元配線方式、階層方式がある。配線構造は、デバイス内の論理ブロックを効率よく接続するための構造である。図 6・2 は 2 次元配線方式の代表的な例として Xilinx 社⁴⁾の XC 4000 シリーズで採用されているセグメント化配線構造を示している。また、図 6・3 では階層方式の例として Altera 社⁵⁾の APEX 20K シリーズで採用されている階層的配線構造を示す。

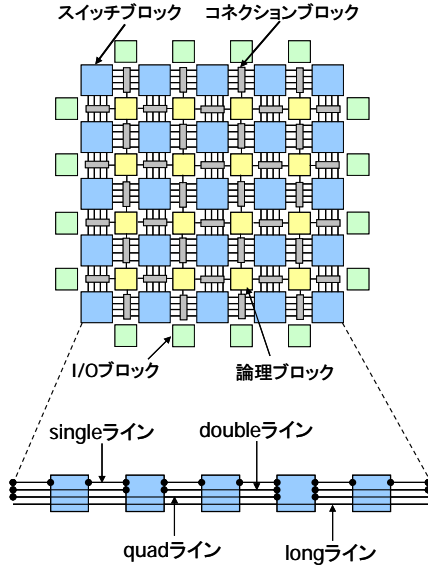


図 6・2 セグメント化配線構造の例 (Xilinx 社 XC 4000 シリーズ)

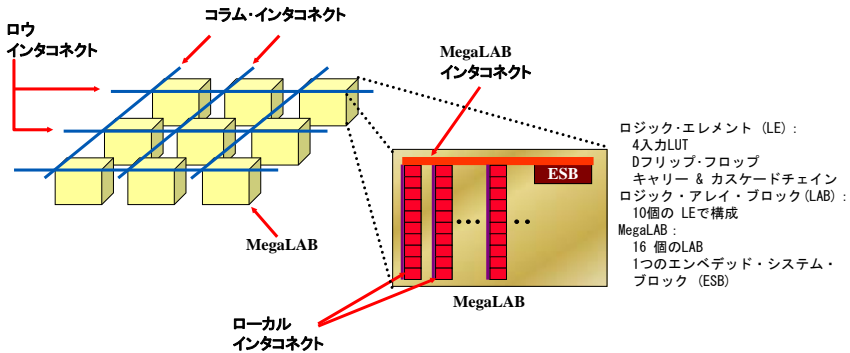


図 6・3 階層的配線構造の例 (Altera 社 APEX20K シリーズ)

2次元配線方式であるセグメント化配線構造は、長さの異なる複数の配線が縦横方向に用意されており、これらを用いて任意の長さの接続を最小スイッチ数で接続できるようにしている。一方、階層化配線構造は、クラスタ内の高速な配線の利点を配線構造に拡大した方式である。複数の論理ブロックをグループ化し、それらを接続する最下層の配線とそのグループ間を接続する中間層、そして最上位層ではグループをグループ化したメガブロック間の接続を提供する。階層化配線構造はこのような階層化した配線資源をもち、ツリー上に接続することで、任意の論理ブロック間の接続スイッチ数を削減している。

6-1-4 部分再構成可能 FPGA とマルチコンテキスト FPGA

従来の SRAM ベース FPGA は、回路の書き換え回数が事実上無限であり、プログラム柔軟性に優れている。しかし、再構成時にはたとえ部分的な変更であってもデバイス全体を書き換える必要があり、FPGA 内のレジスタの内容も失われてしまうといった大きな制約がある。これらの問題を解決するため、部分的な再構成や動作中の再構成を実現できる機能を付加したものに部分再構成可能 FPGA がある。部分再構成可能 FPGA は、バーチャルロジック (Virtual Logic) またはキャッシュロジック (Cache Logic) とも呼ばれる。バーチャルロジックという名称は計算機用語の仮想記憶 (Virtual Memory) に由来しており、両者とも実在する以上のものを提供するところに類似性がある。バーチャルロジックの概念を図 6・4 に示す。

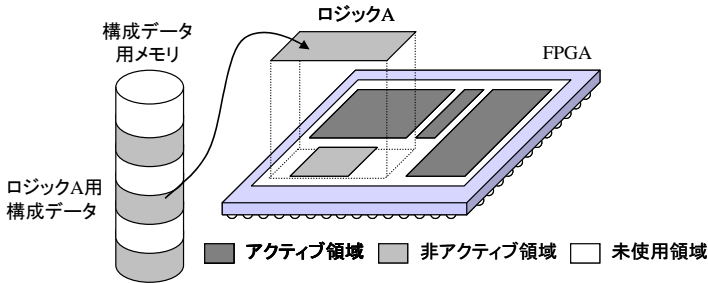


図 6・4 バーチャルロジックの概念

通常、実装される回路すべてが同時に動くわけではない。そこで、FPGA 上の非アクティブな領域に対し、構成データを随時書き換えることで比較的少量のロジックで大規模な回路を実現する。すなわち、ロジックの仮想化である。

部分再構成可能 FPGA のハードウェア構成を変更するためには、外部から新たに構成データをダウンロードする必要があり、部分再構成をスケジューリングすることによって多少隠蔽できたとしても、なお相当な再構成オーバーヘッドが発生してしまう恐れがある。つまり、この再構成オーバーヘッドの短縮が切に望まれていた。

そこで、構成データを複数セット内部に保持し、瞬時に構成データを切り替えることによって外部からの構成データのロード時間を軽減する方式が考案された。一般に、この方式を Xilinx 社の用語に従ってマルチコンテキスト方式と呼び、それぞれのメモリに対応する構成データをコンテキスト、構成データを入れ替えることをコンテキストスイッチと呼ぶ。

マルチコンテキスト FPGA は内部にもつ複数の回路情報を実行時に切り替えることによって、再構成時間の大幅な短縮や回路の仮想化を実現する。具体的には図 6・5 に示すように、チップ内に保持したコンテキストをマルチプレクサで切り替えることで、FPGA 上の回路を高速に入れ替えることが可能となる。例えば、1 サイクルで動的に再構成可能であれば、複数のアプリケーションを必要に応じて呼び出して実行するマルチ機能のみならず、アプリケーションを複数に分割して実行する時分割処理も可能となる。また、回路情報をメモリに蓄えるため、占有面積をロジックに比較して格段に少なくでき、実装密度の向上も期待できる。

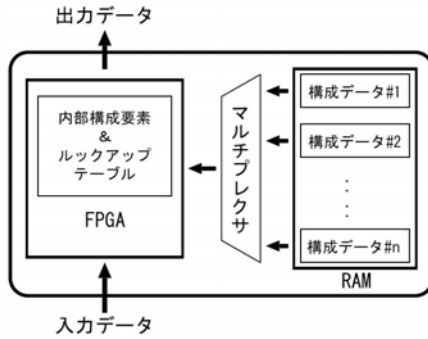


図 6・5 マルチコンテキスト FPGA

■参考文献

- 1) 末吉敏則, 天野英晴 (編著), “リコンフィギャラブルシステム,” pp.1-36, オーム社, 2005.
- 2) Ian Kuon, Russell Tessier, and Jonathan Rose, “FPGA Architecture: Survey and Challenges,” now Publishers Inc., 2008.
- 3) 特許庁, “平成 13 年度特許出願技術動向調査報告書・プログラマブルロジック技術,” 2002.
- 4) <http://www.xilinx.com/>
- 5) <http://www.altera.com/>

■6群 - 5編 - 6章

6-2 リコンフィギャラブルシステム向きアルゴリズム

(執筆：柴田裕一郎) [2009年11月 受領]

リコンフィギャラブルシステムで高い実行性能や処理効率を実現するには、いかに簡単な制御機構でアプリケーションのもつ並列性を引き出すかが鍵となる。このため、パイプライン処理、SIMD 処理、データフローアルゴリズム、シストリックアルゴリズムなどの並列処理技法を効果的に用いる必要がある。本節ではデータフローアルゴリズムとシストリックアルゴリズムについて述べる。

6-2-1 データフローアルゴリズム

データフローアルゴリズムは、Dennis らによって提案されたデータフローモデル¹⁾に基づくアルゴリズムである。このモデルではまず計算をデータフローグラフで表現する。例えば、計算式 $(a + b) - (c + d) \cdot e$ は図 6・6(a) に示すようなデータフローグラフで表現される。データフローグラフの各ノードは演算を表し、ノード間を接続するエッジはデータの依存関係を示している。計算に必要なデータの値はグラフのエッジ上をトークンによって運ばれる。

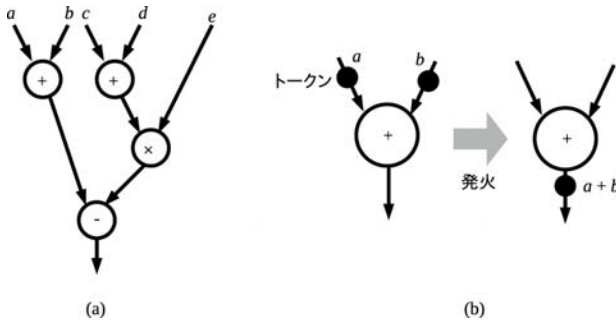


図 6・6 データフローグラフの例と発火のメカニズム

データフローモデルには「制御の流れ」の概念はなく、それぞれの演算の実行はデータ（トークン）の到着によって駆動される。すなわち、各ノードはすべての入力エッジにトークンが到着すると演算可能となり「発火」する。発火したノードは、入力トークンによって運ばれた値をもとに計算を行い、その結果を出力エッジ上に出力トークンとして出力する。この様子を図 6・6(b) に示す。このように、データフローモデルでは自然に処理の並列性を抽出することができる。

リコンフィギャラブルシステムでは、データフローグラフで表現されたアルゴリズムを直接ハードウェア化することで高い並列処理効果を期待できる。しかし、すべての演算ノードにおいて入力トークンの待ち合わせを行うと、制御機構のオーバーヘッドが演算に対して相対的に大きくなる²⁾。そこで、システムの動作時の振る舞いが静的に予見できる場合には、設計時にデータフローの概念を用いて各演算ノードの実行時間を適切に定め、並列性の高いパイプラインを小規模な制御機構で構成することが多い^{3), 4)}。Dennis らのモデルが一つのエッ

上に一つのトークンの存在しか許さないのに対し、一つのエッジ上に複数のトークンの存在を許しトークンの「色」ごとに待ち合せを行う動的データフローモデルも提案されている⁵⁾。しかし、制御機構が複雑になることから、リコンフィギャラブルシステムではあまり用いられない。

6-2-2 シストリックアルゴリズム

シストリックアルゴリズムは、Kung らによって提案されたシストリックアレイ (Systolic Array)⁶⁾での処理を前提としたアルゴリズムである。シストリックアレイは元々専用 VLSI における並列処理用アーキテクチャとして提案され、配線構造や制御構造をできるだけ単純かつ規則的にすることで、演算器の集積度を向上と高い並列処理効果を目指している。シストリックアレイは、単純な演算機構と少量の記憶をもつ処理エレメント (Processing Element : PE) を多数規則的に配置し、隣接 PE 間だけに配線をもたせた構造をもつ。各 PE は規則的にデータを処理することで高い並列処理を実現する。データの入出力もアレイの外周に配置された PE のみが逐次的に行うことで、外部との入出力バンド幅の制約を緩和するとともに長距離配線を排除している。すべての PE は同期して動作し、並列演算効果とパイプライン効果を単純な制御によって抽出する。

PE の配置と接続形態は実行対象のアルゴリズムによって、直線状、2次元アレイ、2次元六角形アレイなど様々に変更する必要がある。また、PE で行う処理の内容も当然ながらアルゴリズムによって変化する。この点はリコンフィギャラブルシステムで実現することにより柔軟に対応できる。

図 6・7 に帯行列 $A = (a_{ij})$ とベクトル $x = (x_1, \dots, x_n)^T$ の積 $y = (y_1, \dots, y_n)^T$ 、すなわち、

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & & & & \\ a_{21} & a_{22} & a_{23} & & & \\ a_{31} & a_{32} & a_{33} & a_{34} & & \\ & a_{42} & a_{43} & a_{44} & a_{45} & \\ & & & & a_{53} & \\ & & & & & & \mathbf{0} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \end{pmatrix}$$

を求めるシストリックアレイの構成例を示す⁷⁾。この例では PE は直線状に配置され、各 PE は右方向から入力されるベクトル y の要素に、左方向から入力されるベクトル x の要素と上方向から入力される行列 A の要素の積を加算し左方向に出力する。また、右方向には左方向から入力されたベクトル x の要素をそのまま出力する。初めに y の要素をすべて 0 に初期化して右端の PE から 2 クロックサイクルに一度ずつ入力すると、左方向に進につれ部分積が加算され、左端の PE からは計算結果 y の要素が 2 クロックサイクルに一度出力される。このほかにも、構成を様々に工夫することにより、行列計算、信号処理など幅広い分野のアルゴリズムをシストリックアレイで実現できることがわかっている。

FPGA を用いたシストリックアルゴリズムの実装例には、記号列検索や画像処理⁸⁾、流体計算⁹⁾など多数あり、多くのアルゴリズムで成果が得られている。また、様々なシストリックアルゴリズムを効率よく実現するためのリコンフィギャラブルアーキテクチャも提案されている¹⁰⁾。ただし、いかなる処理にもシストリックアルゴリズムを適用できるわけではない

点には注意が必要である。

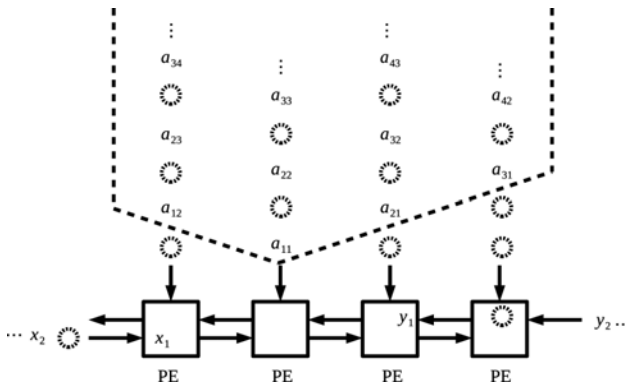


図 6・7 シストリックアレイの構成例⁷⁾

■参考文献

- 1) J. Dennis, "Dataflow Supercomputer," IEEE Computer, vol.13, no.11, pp.48-56, 1980.
- 2) M. Uno, Y. Shibata, and H. Amano, "Implementation of Data Driven Applications on a Multi-context Reconfigurable Device," IEICE Transactions on Information and Systems, vol.E86-D, no.5, pp.841-849, 2003.
- 3) T. Hamada, T. Fukushige, A. Kawai, and J. Makino, "PROGRAPE-1: A Programmable, Multi-Purpose Computer for Many-Body Simulations," Publications of Astronomical Society of Japan, vol.52, pp.943-954, 2000.
- 4) 長名保範, 吉見真聡, 岩岡 洋, 小嶋利紀, 西川由理, 舟橋 啓, 広井賀子, 柴田裕一郎, 岩永直樹, 北野宏明, 天野英晴, "FPGA を用いた汎用生化学シミュレータ ReCSiP," 電子情報通信学会論文誌, vol.J89-D, no.6, pp.1163-1172, 2006.
- 5) Arvind and D. Culler, "Dataflow Architectures," Annual Review in Computer Science, vol.1, pp.225-253, 1986.
- 6) H. Kung and C. Leiserson, "Systolic arrays (for VLSI)," Sparse Matrix Proceedings, pp.256-282, 1978.
- 7) C. Mead and L. Conway, "Introduction to VLSI Systems," Addison-Wesley, 1980.
- 8) J. Arnold, D. Buell, E. Davis, "SPLASH2," Proc. ACM Symposium on Parallel Algorithms and Architectures, pp.316-322, 1992.
- 9) K. Sano, T. Iizuka, and S. Yamamoto, "Systolic Architecture for Computational Fluid Dynamics on FPGAs," Proc. IEEE Symposium on Field-Programmable Custom Computing Machines, pp.107-116, 2007.
- 10) R. Hartenstein, M. Herz, T. Hoffmann, and U. Nageldinger, "On Reconfigurable Co-processing Units," Lecture Notes in Computer Science, vol.1388, (RAW'98), pp.67-72, 1998.

■6 群 - 5 編 - 6 章

6-3 リコンフィギャラブルシステム

(執筆者：佐野健太郎) [2009年11月受領]

リコンフィギャラブルシステムとは、リコンフィギャラブルデバイスを取り入れ、その可変構造によりアプリケーションへの適応性をもつハードウェアを実現するシステムのことであり、これまで、研究開発のためのテストシステムから商用システムまで数多くが開発されている。これらのシステムは、FPGA (Field-Programmable Gate Array) や CPLD (Complex Programmable Logic Device) などのリコンフィギャラブルデバイスを既存のマイクロプロセッサと組み合わせたものや、新たに回路再構成機能をもつ LSI を開発したもので様々である。特に近年では、演算器や内部メモリ、高速 I/O ユニットが多数内蔵された大規模高性能 FPGA を従来の並列計算機に組み込み、高性能計算分野への応用を狙った製品も複数登場している。以下、リコンフィギャラブルシステムの用途、及びシステムの構成方法について分類を行いながら、リコンフィギャラブルシステムの概要を述べる。

これまで、参考文献 1)~6) に見られるように、リコンフィギャラブルシステムの異なる分類方法が提案されている。例えば、参考文献 1), 4), 5) では、リコンフィギャラブルシステムをリコンフィギャラブルユニット (RU) と汎用のマイクロプロセッサ (μ P) との関係により、ハードウェアエンジン型、コプロセッサ型、プロセッサ型の三つに分類している。また、参考文献 2), 3) では、システムを大規模化する際の基本となる構成要素 (ノード, node) の均一性に着目し、均一システムと不均一システムの二つに分類している。ここでは、提案されているこれらの分類方法をまとめて、リコンフィギャラブルシステムを図 6・8 のように分類する。

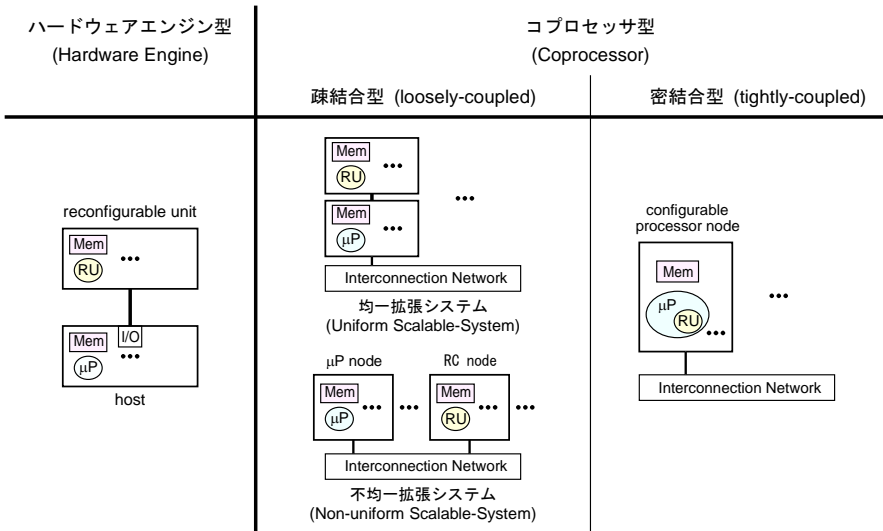


図 6・8 リコンフィギャラブルシステムの分類

図 6・8 の分類では、リコンフィギャラブル部により行われる処理の粒度と独立性により大きくハードウェアエンジン型とコプロセッサ型の二つに分類するほか、コプロセッサ型については、狭義の分類として、リコンフィギャラブル部により行われる処理のマイクロプロセッサへの依存の度合いにより、更に疎結合型と密結合型の二つに分ける。加えて、拡張可能なシステムを構成する場合のノードの同種・異種に着目し、疎結合コプロセッサ型システムを均一拡張システムと不均一拡張システムの二つに分類する。

6-3-1 ハードウェアエンジン型リコンフィギャラブルシステム

ハードウェアエンジン型のシステムでは、比較的大きな粒度の独立した処理がリコンフィギャラブル部により行われるのが特徴である。システムの起動や制御、そして入出力のために汎用のプロセッサがホストとして使用されることもあるが、あくまでリコンフィギャラブル部が処理の大半を担当する。ハードウェアエンジン型システムは、FPGA が搭載されたボードなどやホストプロセッサを、CPU バスや、PCI バス、PCI-Express I/F、USB などのシステム I/O により接続して構築される。ハードウェアエンジン型システムの例としては、米国計算機科学センターの Splash/Splash 2、神戸大学の RM (Reconfigurable Machine)-I/II/III/IV、ヒューレット・パッカード社の Teramac、三菱電機の RASH (Reconfigurable Architecture based on Scalable Hardware) がある。

6-3-2 疎結合コプロセッサ型リコンフィギャラブルシステム

一方、コプロセッサ型のシステムでは、主に目的とする処理の一部分を高速化するためにリコンフィギャラブル部が用いられる。このため、コプロセッサ型システムにおけるリコンフィギャラブル部はハードウェアエンジン型ほど独立しておらず、残りの処理を担当するマイクロプロセッサに依存するシステムとなる。また、リコンフィギャラブル部が行う処理の粒度は、ハードウェアエンジン型よりも小さいことが多い。

ここでは、コプロセッサ型のシステムを更に、疎結合型と密結合型に分類する。疎結合コプロセッサ型システムは、それ単体で開発された既存のマイクロプロセッサに FPGA などのリコンフィギャラブルデバイスを組み合わせたもので、マイクロプロセッサが苦手とする処理を比較的大きな粒度でリコンフィギャラブル部により高速化するアプローチである。マイクロプロセッサとリコンフィギャラブル部はそれぞれ単独でも存在し得るものであることから、密結合型と比べてマイクロプロセッサへの依存度は弱い。

拡張性が限定的な初期の疎結合コプロセッサ型システムは、ハードウェアエンジン型と同様にマイクロプロセッサを有するホストに主に I/O やシステムバスによりリコンフィギャラブル部を接続して構成される。このようなシステムの例としては、米国 Brown 大学の PRISM/PRISM-II、フランス Bretagne Occidentale 大学の ArMen、California 大学 Berkley 校の Garp、メルボルン大学の RECON、オクスフォード大学の HARP などがある。

一方、高性能計算向けのリコンフィギャラブルシステムとして、マイクロプロセッサやリコンフィギャラブル部を自由に拡張可能な疎結合コプロセッサ型システムが開発されている。このような拡張可能なシステムは、FPGA からなるリコンフィギャラブル部を PCI バスや PCI-Express I/F によりローカルに接続したマイクロプロセッサノードをネットワークにより接続する構成の均一拡張システムと、マイクロプロセッサノードと独立したリコンフィギャ

ラブルノードをネットワークにより接続する構成の不均一拡張システムに分けられる^{*1}。均一拡張システムは、ノード内で接続されるマイクロプロセッサとリコンフィギャラブル部の間では比較的広帯域幅で低遅延のデータ転送を実現しやすい一方で、一般に、異なるノード間ではリコンフィギャラブル部を連携させて動作させるのが難しい。FPGAボードを搭載したPCを複数台並べたBeowulfクラスタや、米国SRCコンピュータ社のSRC 6、SRC 7が均一拡張システムの例である。これに対し、不均一拡張システムは、マイクロプロセッサとリコンフィギャラブル部の構成比を自由に変えた柔軟な構成が可能であり、また、複数のリコンフィギャラブル部をまとめて使用することを想定して設計されていることが多い。米国Cray社のXD1やXT5、米国SGI社のAltix RASC (Reconfigurable Application Specific Computing) は不均一拡張システムに分類される。

6-3-3 密結合コプロセッサ型リコンフィギャラブルシステム

密結合コプロセッサ型システムは、リコンフィギャラブル部を取り入れて新たに開発されたリコンフィギャラブルマイクロプロセッサにより構成される。一般に、このようなプロセッサは既存のマイクロプロセッサと同様なデータパスとリコンフィギャラブル部が密接に融合した構造をもち、リコンフィギャラブル機能により命令セットを可変または拡張可能としている。頻出する一群の演算を一度に実行する命令や、一般的なマイクロプロセッサが効率よく扱えないようなビット幅の演算を行う命令などを必要に応じて追加することにより、プログラムの実行を高速化する。このように、密結合コプロセッサ型システムはリコンフィギャラブル部がマイクロプロセッサ部に強く依存する構造をもち、また、リコンフィギャラブル部において行われる処理の粒度は比較的小さい。アイピーフレックス社の DAPDNA-2 は密結合コプロセッサ型システムの例である。

■参考文献

- 1) 末吉敏則, 天野英晴(編著), “リコンフィギャラブルシステム,” オーム社, 2005.
- 2) Tarek El-Ghazawi, Duncan Buell, Kris Gaj, and Volodymyr Kindratenko, “Reconfigurable Supercomputing,” Tutorial of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC07), 2007.
- 3) Tarek El-Ghazawi, “High-Performance Reconfigurable Computing,” Tutorial of the International Conference on Field-Programmable Technology (ICFPT’07), 2007.
- 4) 末吉敏則, “Reconfigurable Computing System の現状と課題—Computer Evolution へ向けて—,” 電子情報通信学会 信学技法, vol.96, no.426, pp.111-118, 1996.
- 5) Toshiaki Miyazaki, “Reconfigurable Systems: A Survey,” Proceedings of the Asia South Pacific Design Automation Conference (ASP-DAC), pp.447-452, 1998.
- 6) Steven A. Guccione and Mario J. Gonzalez, “Classification and Performance of Reconfigurable Architectures,” Proceedings of the International Conference on Field Programmable Logic and Applications (FPL), pp.439-448, 1995.

^{*1} 拡張性の限定的な疎結合コプロセッサ型システムは、均一拡張システムに分類される。

■6群 - 5編 - 6章

6-4 動的リコンフィギャラブルシステム

(執筆著：天野英晴) [2009年11月 受領]

6-4-1 動的リコンフィギャラブルシステムとは

動的リコンフィギャラブル^{1),2)} (動的リコンフィギュラブル, 動的リコンフィギャブル, 動的再構成可能) システム (プロセッサ, アレイ) は, 動作中に構造を変更することによって, 高い面積利用率を達成することのできる再構成可能 (リコンフィギャラブル) な演算ユニットである。

SoC (System on a Chip) 内のオフロードエンジンとして, 柔軟性が高く, コスト, 消費電力の面で有利な特性が注目され, IPFlex 社の DAPDNA 2, NEC electronics 社の DRP-1, 日立の FE-GA, 東芝の SAKE, サンヨーのカーチューナ用アーキテクチャ, 松下 (Elixent) の DFabrix など多くの半導体企業がこの方式を試みており, SONY の VME³⁾ などは, 既に製品に組み込まれて用いられている。国外でも PACT Xpp, IMEC ADRES, Rapport Kilocore など様々な開発例がある (個別のサーベイは文献 2) などを参照)。

近年, 半導体プロセスが進むにつれ, マスク代と半導体の製造費が高価になり, 単一目的の SoC の開発は, よほどの数が見込まれる場合を除いて困難になってきた。一つのチップを様々な目的に利用するためには, 特定用途別に高い性能を出す処理装置 (オフロードエンジン) に固定のハードウェアではなく, より柔軟な再構成可能 (リコンフィギャラブル) デバイスを用いることで可能となる。しかし, 現在, もっとも普及しているリコンフィギャラブルデバイスである FPGA (Field Programmable Gate Array) は, 構成要素が LUT (Look Up Table) などの小さなハードウェアブロック (細粒度) であるため, 柔軟性が高くても専用ハードウェアに比べて全体の面積が大きくなり, 性能, 消費電力の点でも不利である。そこで, 演算器とレジスタなどからなる PE (Processing Element) を構成要素とした粗粒度 PE アレイに, 動作中に構造を高速に変更する機能, すなわち動的再構成機構を付け加えたものが動的リコンフィギャラブルシステムである。動的再構成により, 面積効率を改善するとともに, PE アレイのサイズを越える問題でも簡単に実装することができる。

6-4-2 PE と PE アレイの構成

図 6・9 に典型的な PE の構成を示す⁴⁾。この PE は, 算術, 論理演算を行うユニット (ALU), シフト, マスク, コンスタントデータの出力などを行うユニット (SMU) 及びレジスタファイルをもつ。それぞれの機能と接続は構成情報 (Conguration Data) あるいは命令 (Instruction) により定義され, これを変更することにより動的再構成を行う。

図中では, データ幅は 32 bit であるが, 扱うビット数は 4 bit, 8 bit, 12 bit, 16 bit など様々であり, PE の構成も多様である。システムによっては, 乗算器など特殊な機能をもつ PE を一定の割合でアレイ中に混ぜるヘテロロジーニアスな構成を採る場合もある。また, PE によっては, レジスタファイルではなく, 演算器の入力または出力に単一のレジスタをもたせる場合もある。

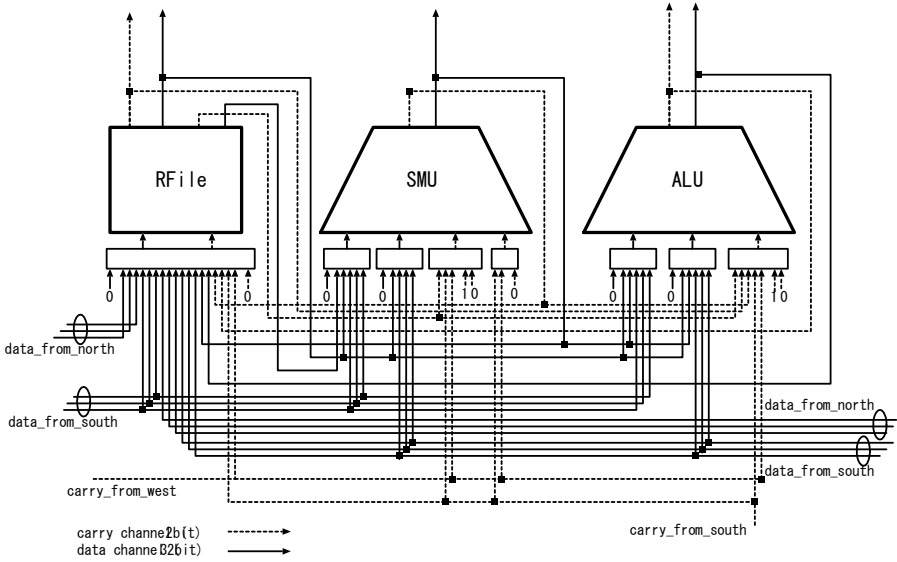


図 6・9 PE の構成例

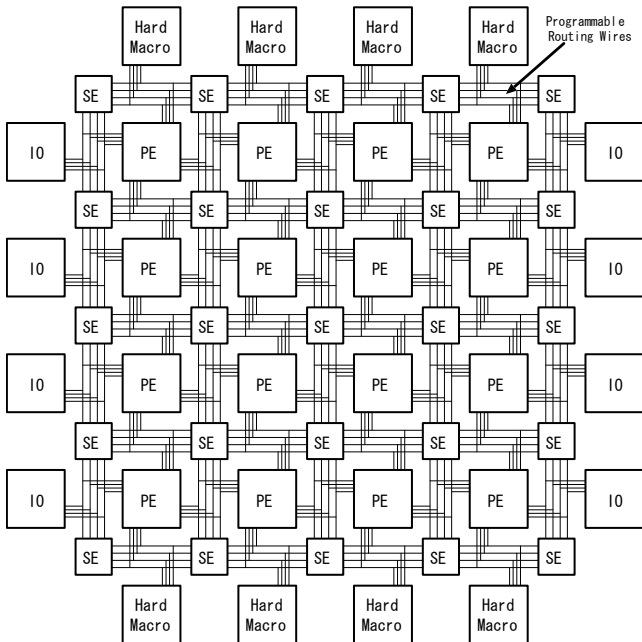


図 6・10 PE アレイの構成例

図 6・10 に PE アレイの構成例を示す⁴⁾。この構成は、FPGA 同様に PE と PE の間に配線領域を設け、交点にスイッチを置く方式で、アイランドスタイルと呼ばれている。PE と配線間を接続するコネクシオンブロックとスイッチの接続設定は、PE 内と同じ構成情報によって決められる。一方、PE 同士を直接リンクで接続する直結方式も広く用いられており、これはむしろマルチコアのプロセッサ接続に近い方式である。アイランドスタイル、直結型ともに、接続形態はメッシュを基本とした構造あるいは直線状が多い。

図 6・10 中では PE アレイの周囲にメモリモジュールと特殊機能モジュールをもたせている。このように PE 自体はホモジニアスであっても、周辺に特殊なモジュールを置くことで、対象とするアプリケーションの要求を満足させる。特にメモリモジュールはほとんどのシステムで PE アレイの周辺に配置され、同時にアクセスされることで、同時に動作できる PE 数を増やして性能の向上を図っている。

6-4-3 動的再構成手法

動的リコンフィギャラブルシステムの最大の特徴は、PE、コネクシオンブロック、スイッチ内の構成情報を動作中に変更することで、様々な用途に PE アレイを利用することが可能な点である。この手法には大きく分けて二つある。一つはチップ内部に備えているコンフィギュレーションメモリから各 PE 及びスイッチに、順にコンフィギュレーションデータを配送する方法である。この方法は、データの送付に時間が掛かることから、全体の構成を変更するためには数マイクロ秒程度を要する。この方式を構成情報配送方式と呼ぶ。

一方、それぞれの PE 及びスイッチに構成情報を格納できるレジスタを複数セットもたせておけば、コントローラからポインタを配って、レジスタ番号を指示する（あるいはポインタに従ってレジスタファイルからデータを読み出す）ことにより、構成を一瞬にして変更することができる。この方式をマルチコンテキスト方式と呼び、それぞれの構成情報で実現される PE 上の構成のことをハードウェアコンテキスト、ポインタを配って構成情報を入れ替える作業をコンテキストスイッチと呼ぶ。コンテキストスイッチは、場合によっては 1 クロックで実行が可能であり、マルチコンテキスト方式の再構成時間は構成情報配送方式に比べてずっと短い。一方、この方式では、構成情報を格納するレジスタあるいはメモリ（コンテキストレジスタ、コンテキストメモリと呼ぶ）をそれぞれの PE でもつための面積上の損失が大きい。

6-4-4 まとめ

構成情報配送方式の動的リコンフィギャラブルシステムは、構成情報の変更時間が高速で、演算機能に特化した FPGA とも考えられる。一方で、マルチコンテキスト型の動的リコンフィギャラブルシステムは、巨大なデーパスをもつ代わりに命令メモリが制限られた VLIW (Very Long Instruction Word) 方式とも考えられる。このように動的リコンフィギャラブルシステムは、アーキテクチャ上の選択肢の幅が広い。表 6・1 中には、各システムが今まで紹介した方法のどれを用いているかを示した。それぞれのシステムが様々なパターンで各種方式を採用していることがわかる。最近、ストリーム処理用に構成をチューンした東芝の SAKE、カーチーナ用に特化した SANYO のアーキテクチャ、ワイヤレス通信処理に特化した富士通のシステムなど、ある程度対象アプリケーションに絞ら込むことにより、配線資源やコン

テキスト用のメモリなどに要する面積を削減している。また、目的別に作り込んだ小規模な PE アレイを複数用いるマルチコア方式で更に効率を改善する。しかし、構成を一定のアプリケーションに絞り込むことは、逆に柔軟性を失うことになるため、どの程度の範囲のアプリケーションを対象として専用化するかが問題となる。適切な範囲で設定することができれば、極めて性能、コスト、消費電力に優れたアクセラレータを構成することができる。

■参考文献

- 1) 末吉, 天野編, “リコンフィギュラブルシステム,” オーム社, 2005.
- 2) H.Amano, “A Survey on Dynamically Reconfigurable Processors,” IEICE Trans. Comm. Vol.E89-B, no.12, pp.3179-3187, 2006.
- 3) Y.Kurose, I.Kumata, M.Okabe, H.Hanaki, K.Seno, K.Hasegawa, H.Ozawa, S.Horiike, T.Wada, S.Arima, K.Taniguchi, K.Ono, H.Hokazono, T.Hiroi, T.Hirano, and S.Takashima, “A 90nm embedded DRAM single chip LSI with a 3D graphics, H.264 codec engine, and a reconfigurable processor,” Hot Chips 16, 2004.
- 4) H.Amano, Y.Hasegawa, S.Tsutsumi, T.Nakamura, T.Nishimura, V.Tunbunheng, A.Parimala, T.Sano, M.Kato, “MuCCRA Chips: Congurable Dynamically-Reconfigurable Processors,” Proc. ASSCC2007, pp.384-387, Nov. 2007.