リコンフィギャラブルシステム最新情報
Recent activites on Reconfigurable Systems

天野 英晴 †
†慶應義塾大学理工学部

Hideharu AMANO†
†Dept. of Information and Computer Science, Keio University

1　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システムとは，FPGA(Field Programmable Gate Array)，CPLD(Complex Programmable Logic Device) を代表とする大規模なプログラマブルデバイス上で，問題の解法を直接実行することにより，柔軟性と高速性を共に実現するシステムである．

リコンフィギャラブルシステム [4] は，汎用 FPGA と CPLD が本格的に発達した 90 年代初期から研究が本格化した. 初期のリコンフィギャラブルシステムは，多数の FPGA を接続し，特定の応用分野の高速化を目標とした．米国計算センターにより 1992 年に開発された SPLASH-II[5] は，DNA の塩基配列間の距離を解析し，スーパーコンピュータ CRAY-II の 300 倍の性能を発揮して話題となった. 国内でも RM シリーズ [6] などが大学で試作され，三菱電器からは大規模な商用機として RASH[7] が開発された. 90 年代の後半には，FPGA を搭載したカードが一般に普及し，この上で，論理シミュレーション，故障シミュレーション，暗号化，復号化，画像処理，パターンマッチング，プロダクションシステム，離散系シミュレーション，グラフ解析，GA，ニューラルネットワーク，バイオインフォマティクス等のアプリケーションにおいて効果が確認された．さらに，NTT による YARDS[8] など，ネットワークコントローラ，プロトコル変換さらにはソフトウェア無線などへの応用が試みられた.2000 年代になり，汎用 FPGA，CPLD がさらに発展を遂げると共に，このようなリコンフィギャラブルシステムは，ますますその応用範囲を広げ，その研究領域も多様化している.

また，汎用プロセッサにリコンフィギャラブル機構を組み込む研究も 90 年代後半に進み，リコンフィギャラブル部とキャッシュ共有する Garp[9] やリコンフィギャラブル命令を持つ DISC[10]，CHIMAERA[11] などが登場した．さらに，2000 年に Chamelleon 社からダイナミックリコンフィギャラブルプロセッサ CS2112[12] が商品化され，その後，次々と専用デバイスが登場している．これらのデバイスは，粗粒度の構成要素を用いると共に，面積効率を改善するため，高速動的再構成機能を持っている [32].

これらの，ダイナミックリコンフィギャラブルプロセッサは，汎用 FPGA を用いたリコンフィギャラブルシステムの発展形というよりは，SoC (System-on-a-Chip) のハードウェア部あるいは DSP 部の代替製品として，より広い分野での応用を狙っている．本稿は，この後者の最近の動向に関して主に紹介する．

2　従来型リコンフィギャラブルシステムの動向

汎用 FPGA，CPLD を用いた従来型のリコンフィギャラブルシステムは，特定用途における高速化，基地局通信制御，ネットワークインタフェース等，柔軟性を必要とする少数生産機器で一定のシェアを確立している．最近，特に注目されているのは，以下の動向である.

大規模 SoPD を利用した新応用分野への進出:　乗算器，高速リンク，CPU，大容量分散メモリ等を組み込んだ大規模 FPGA(Xilinx 社 Virtex-II Pro[1], Altera 社 Excalibur[2]) が登場した．これらは，システムをまるごとプログラマブルデバイス上に実装可能なことから SoPD(System on Programmable Device) と呼ばれる．これらのデバイスの利用により，通信，画像処理，バイオインフォマティクス分野など従来からリコンフィギャラブルシステムが得意とする分野において，さらに強力なアクセラレータを開発可能となった．さらに，組み込み演算器や Altera 社の Stratix などの高速 FPGA を利用し，従来リコンフィギャラ

ブルシステムには不向きであると考えられてきた数値計算や大量のデータをアクセスする応用分野への進出も始まっている．特に，ようやく浮動小数演算が汎用 CPU に太刀打ちできる性能に達したことから，科学技術演算への本格的応用が始まっている．2004 年の FCCM では，浮動小数演算の実装手法と科学技術計算への応用についての発表がもっとも多くのセッションで行われ，注目を集めた [20].

CPU-Reconfigurable System 協調設計: CPU 内蔵 FPGA を用いることで，ソフトウェアとリコンフィギャラブルシステム上のハードウェアとの協調作業を単一チップ内で行なうことができる．通常のハードウェアと異なり，リコンフィギャラブルシステムはランタイムの構成変更が可能であるので，状況に応じて構成を変更して行く技術，組み込み OS で構成変更を管理する技術，C 言語レベルの協調設計技術，遠隔再構成技術，動的再構成技術等の開発が進んでいる．

組み込みシステムへの本格的応用: FPGA は ASIC に比べてどうしても高価で電力消費量が大きくなりやすい．このため，リコンフィギャラブルシステムは組み込みシステムでの応用が期待されながら，実際の利用は遅れていた．しかし，最近 Xilinx 社の Spartan 等低価格な FPGA が普及し，低電圧化による低電力化が進むと共に，実際の組み込みシステムにおいて構成変更によるメリットを生かす試みが実用化されはじめている．

3 ダイナミックリコンフィギャラブルプロセッサ

3.1 SoC の問題点

メディア処理や通信制御では，MPEG や JPEG などのコード圧縮および復号，DES, AES などにより暗号化されたコード，Viterbi，Turbo などのエラー修正用コードの取り扱いなど，強力な演算能力を必要とされる処理を含んでいる．一方で，システム管理，ユーザインタフェース機能も必要であり，データ処理の一部にも比較的演算能力を要しないが複雑な処理が混在している．SoC は，このような性質に適合するように，演算能力は必要としないが複雑度の高い処理を担当する組み込み用 CPU と，強力な演算能力を要する処理をハードウェア化した部分を同一チップ上に混載することにより，低コスト，高性能，低消費電力の ASIC を実現している．

しかし，専用ハードウェアは，特定の処理に特化してしまうため，他の用途に転用することができず，用途別に様々な構成のハードウェアを混載した ASIC 開発が必要となる．しかも，JPEG2000，Turbo コードなど新しく複雑なコード化方式が次々に登場し，これに合わせて次々と新たな専用ハードウェアを混載する必要が生じる．もちろん，このような SoC を短期間で開発するため，C レベル記述，協調設計などの設計技術が発達しており，上流の設計時間は従来に比べて短縮されている．ところが，最近のプロセスは素子の遅延に比べて配線遅延が支配的になり，性能面で新しいプロセスの恩恵を受けるためには，配置配線以降の下流設計に多大な時間を要するようになった．このため，ASIC 設計自体は，プロセスが進むにつれて，むしろ困難になる傾向にある．これらの状況から，専用ハードウェアを搭載した新しい SoC を次々に市場に投入するための負担が，ベンダーにとって大きなものとなっている

ここで，専用ハードウェアに代わる効率の良い柔軟性を持ったハードウェアがあれば，一種類のチップを様々な応用分野に用いることができ，新しい技術も簡単に採り入れることが可能である．開発コストを削減すると共に，単一のチップを大量に生産すれば良いことからチップ自体のコストの削減も期待できる．柔軟性の持ったハードウェア自体の構造が決まっていれば，新しいプロセスにもいち早く対応することができる．また，場合によっては製品の配布後もハードウェアの構造を変更することで，発生した問題点や新たな機能付加に対処することができる．

3.2 汎用 FPGA の問題点

それでは，柔軟性を持ったハードウェアとして，汎用の FPGA や CPLD を用いてはどうであろうか？最近の FPGA や CPLD 技術の進展は目ざましく，性能面で ASIC に迫る製品や，価格面で ASIC の市場に進出している製品が出現している．さらに，実際に CPU と FPGA を混載したシステムは既に市場に出回っている．

これらの FPGA/CPLD は 4 入力 2 出力程度の LUT(Look Up Table) または AND-OR 接続のプロダクトターム方式に Flip Flop を設けた構成の基本構成要素を用いている．基本構成要素のハードウェア量が小さいことから，これらを細粒度のリコンフィギャラブルデバイスと呼ぶ．細粒度の構成は，演算回路，制御回路など様々な種類の論理回路を無駄な

く実現できる点で柔軟性は高いが，以下の問題点がある．(1) メディア処理で用いられる演算器を実現した場合，専用ハードウェアの数倍程度の面積を要し，動作速度の点でも不利である [3]．(2) 基本構成要素に効率良く論理回路をマッピングする配置配線用 CAD が複雑なものとなる．現在の FPGA ベンダーはこれに関して膨大なノウハウを持っており，これに対抗するのは難しい．(3) 細粒度の構成に関しては，米国 FPGA ベンダーが網羅的に特許を持っているため，商品化にはライセンス契約が必要である[1]．

以上のように，CPU+FPGA/CPLD のアプローチは純粋に技術的に考えても，SoC のハードウェア部に柔軟性を与える解法としては困難な点が多く，それ以外の情勢も考えると，特に新規ベンダーがおいそれとは手を出すことができない領域である．実際，現状では，CPU と FPGA の混載チップは高価格であり，その利用は，プロトタイピングやリコンフィギャラブルシステムとしての用途に限られている．

3.3 粗粒度構成要素の利用

細粒度のリコンフィギャラブルシステムの問題点は，構成ユニットを LUT や AND-OR アレイから，8bit-10bit の演算器レベルに大きくすることによって，ある程度解決することができる．もちろん，演算器だけでは柔軟性に欠くため，レジスタ，マルチプレクサ，シフタ等と組み合わせた構成を取る．図 1に，典型例として Chameleon CS2112 の構成ユニットである DPU を示す[2]．32bit の演算器，レジスタ，シフタ，マルチプレクサから構成され，個々の動作および構成要素間の接続を変更可能にすることにより様々な構成を実現する．CS2112 ではこの DPU を 96 個用いて並列処理を行なうことができる．

一方，NEC 社の DRP-1[13] の場合，図 2に示す 8bit のプロセッシングエレメントの構成を持つ．ここで，DMU は，任意のシフト，マスク，データセレクト，簡単な論理演算を行なうユニットであり，論理演算，算術演算を行なう ALU，フリップフロップおよびレジスタファイルとの組合わせで多様な機能を実現する．DRP は 8bit 構成の小さい PE を多数用いる方式であり，DRP-1 には総計 512 個の PE が搭載されている．

これらの方式を粗粒度の構成と呼ぶ．粗粒度の構成は，細粒度の構成に比べて柔軟性は低いが，メディア処理などで高速化が必要な演算処理を実行する場合には，面積効率，実行速度の点で有利である．一般に 4bit や 8bit などビット幅が小さい程，面積効率と柔軟性は高いが，大きいビット幅に対する演算速度の点で不利である．逆に 32bit などの大きなビット幅を用いると，対象アプリケーションのビット幅とうまく適合すれば高速かつ面積効率が良く，構成情報量も少なくて済む．しかし，対象とうまく合わない場合には，無駄が大きくなる．

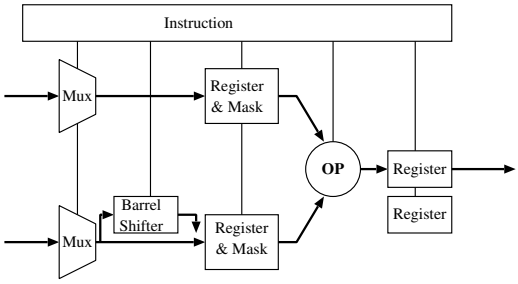

図 1: CS2112 の構成要素 DPU

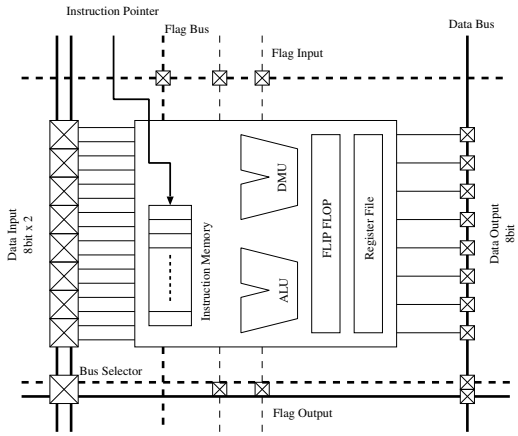

図 2: DRP の Processing Element

3.4 ダイナミックリコンフィギュレーション

粗粒度構成のプロセッシングエレメントを多数用いて並列処理を行なえば，場合によっては専用ハードウェアにそれほど劣らない性能を実現することができ，細粒度構成の FPGA に比べれば面積も小さくて済む．とはいえ，専用ハードウェアで実現するアプリケーションを粗粒度構成のリコンフィギャラブルデバイス上に実現した場合，どうしても面積の点では

[1]PowerPC と FPGA の混載は IBM と Xilinx がクロスライセンス契約により互いのハードウェア資産を利用可能にしている．

[2]CS2112 は現在では販売を終了している

不利となる．粗粒度構成といえども，FPGA や PLD 同様，様々なアプリケーションに対する柔軟性を実現するためには，機能や接続を可変にする必要があり，構成要素，配線面積，配線用スイッチなどあらゆる点で専用ハードウェアに対してハンディキャップを負うことになる．

そこで，構成要素の機能や接続を変更可能である点，つまりリコンフィギャラブルである点を利用し，面積効率の改善を図る方法がダイナミックリコンフィギュレーションである．多くのメディア処理では，入力されたストリームに対して，順に一連のタスクを施していく場合が多い [16]. ここで，ストリームの入力間隔にある程度の余裕がある場合，それぞれのタスクに相当する構成を用意しておき，一つのタスクが終了した後に，構成を動的に次のタスク用に変更して処理を進めていくことができる. このような処理中に動的に構成変更することをダイナミックリコンフィギュレーションと呼ぶ. この手法により，全てのタスクに対応するハードウェアを持たせておく場合に比べて，面積効率を大幅に改善することができる.

ダイナミックリコンフィギュレーションの実現方式は，内部メモリからの命令／構成情報配送方式とマルチコンテキスト方式に大別される.

3.4.1 命令/構成情報配送方式

ダイナミックリコンフィギュレーションは，FPGA や CPLD などの汎用リコンフィギャラブルデバイスでも可能である. しかし，汎用のデバイスは細粒度構成を取ってるため，構成情報の量が多く，また，チップ外のメモリから転送されるため，再構成に多大な時間を要する.

一方，粗粒度構成の場合，特に 32bit など，構成要素の粒度が大きい場合，ALU やシフタなどに対する設定情報は，FPGA の構成情報 (Configuration) よりはるかに少量であり，命令 (Command/Instruction) と呼ぶ方が相応しい．要素プロセッサに対する命令，および構成要素間，要素プロセッサ間の配線等の構成情報をチップ内の複数のメモリモジュール内に格納しておき，これらを専用のバスにより一斉に各プロセッサや接続スイッチに配送することにより，高速なダイナミックリコンフィギュレーションが可能となる．この方式をここでは命令／構成情報配送方式と呼ぶ. PACT Xpp[23] および Elixent DFA1000[24] は命令／構成情報配送方式で再構成を行なっている．この方法では，命令 (構成情報) メモリはいくつかの場所で集中的に実装されるため，面積効率の点で有利であり，多数の命令 (構成情報) を保持することができ，外部とのやり取りも容易である．通常，再構成に要する時間は μsec オーダーに抑えることができ，汎用 FPGA が時に数十 $msec$ オーダーの再構成時間がかかることに比べると，はるかに高速である. 問題は，再構成中には処理を停止せざるを得ないことである. ダイナミックリコンフィギャブルプロセッサは，CPU と同一チップに実装される場合が多いため，再構成中に，CPU で別の小さなタスクを実行することができれば，この間は隠蔽することができる. これができない場合は，再構成時間は純粋なオーバヘッドとなってしまう. したがって，この方式は，再構成の間隔があまり頻繁ではない用途に適している.

3.4.2 マルチコンテキスト方式

マルチコンテキスト型リコンフィギャラブルデバイスは，主として細粒度のリコンフィギャラブルデバイスを対象として，早い時期から提案され [17][18]，NEC により 1999 年に開発された DRL[19] により，実用レベルに達していた．この方式では，FPGA の構成情報を記憶する構成情報メモリを複数セット持ち，これをマルチプレクサを介して切り替えることで，一瞬で FPGA の構成を変更する．一般にこの方式を Xilinx 社の用語 [18] 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情報をコンテキスト，構成情報を入れ替えることをコンテキストスイッチと呼ぶ．例えば，NEC の DRP-1 は，16 コンテキストを保持しており，これらを 1 クロックで切り替えることができる.

マルチコンテキスト方式は，粗粒度構成にも適用可能であり，細粒度構成に比べて構成情報の量ははるかに少なくすることができる点で有利である．粗粒度の場合，要素プロセッサの機能設定データは命令に近いので，この方式は各要素プロセッサがローカルに命令を持っていて，これを全体で同期を取って，プロセッシングエレメント間の接続と共に切り替えるという考え方も可能である．マルチコンテキスト方式の場合，あるコンテキストでタスクを実行している間に，利用していないコンテキストに対して次のタスクの命令／構成情報を設定しておき，タスクの実行終了時に 1 クロックで切り替えることが可能である．このことで，命令／構成情報分配方式で生じた再構成のためのオーバヘッドを無くすことができる. マルチコンテキスト方式の中で，コンテキスト数が少ないものは主としてこのような使い方をする. この場合，コンテキスト切り替えの頻度はタス

ク単位であり，あまり頻繁ではない．

さらに，マルチコンテキスト方式を用いることで，単一の大きなタスクの時分割実行も可能である．一般的に複雑度の高い処理をハードウェア化した場合，全体が常に動作しているわけではなく，動作しているのはその一部に限られる場合が多い．そこで，全体をいくつかのコンテキストに分解してこれを次々に切り替えながら，処理を進めていくことにより，少ない面積でタスク全体に対するハードウェアと等価な処理を実現可能にする．もちろん，時分割多重をする分，性能面では不利になる．単純に考えるとn時分割多重をすれば面積が1/nになる一方で，性能も1/nになってしまう．しかし，分割前の回路でも全ての演算器がフルに動いているわけではないので，分割方法を工夫して，演算器の利用効率を高めれば，面積は小さくした上，性能低下をかなりの程度に抑えることが可能である．マルチコンテキスト方式の中で，多数のコンテキスト数を持つものは，この種の使い方が可能で，この場合，コンテキストスイッチは極めて頻繁になる．コンテキストの制御は，WASMIIではデータ駆動型を採用しているが，多くのアプリケーションでは動的にコンテキストを選択する必要は少なく，DRPでは状態遷移を用いている．

3.5 C レベル設計との結び付き

粗粒度の構成要素は，レジスタ，シフタ，マスク，演算器から構成されており，簡単なプロセッサのデータパスに相当する．このため，粗粒度リコンフィギャラブルシステムはCレベルのハードウェア記述の割り付けが容易である．この場合，通常のC言語における複雑な演算，動的なポインタ，２次元配列などの利用を制限する一方，データ幅やレジスタに対する代入などを明示的に表示する記述を加える必要がある．逆に，一般のVerilogなどのHDLを用いる場合，粗粒度のデータパス要素をライブラリで定義されたモジュールとして扱う必要が生じる．

さらに，マルチコンテキスト方式で多数のコンテキストを持つ場合，コンテキストを切り替えることにより，データパス全体の構造を１クロックで変更することが可能になる．後に触れるように，この方式のダイナミックリコンフィギャラブルシステムは，FPGAとVLIW型のプロセッサの中間的な性質を持っている．この点を利用すれば，C言語で記述されたハードウェアを単一コンテキストに比べて効率良く変換することができる．

4 ダイナミックリコンフィギャラブルプロセッサの実例

ここでは，いくつかのダイナミックリコンフィギャラブルプロセッサを紹介する． 他の様々な提案[17][27][29] [28][30]については，紙面の関係で割愛せざるを得なかった．文献を参照されたい．

4.1 命令/構成情報配送方式

4.1.1 PACT 社 Xpp

PACT Informations Technologie 社の Xpp[23] は，典型的な命令配送方式のダイナミックリコンフィギャラブルデバイスである．

Xppの最小構成チップXPU48は，24bitのALUを2セットと転送用レジスタセットから成るALU PAE (Processing Array Element) 48セットを6×8の2次元のアレイ状に接続した構造を持つ．図3に示すように，各PAEは双方向の専用線で縦方向に接続され，バスを介して横方向に接続される．このバスを介してPAEは両側配置されているデータ格納用RAM PAEをアクセスする．コンフィギュレーション用RAMはPAEアレイの下方向に装備されており，このRAMからコンフィギュレーション情報を流し込むことによって再構成を実現する．このことにより，μsecオーダで再構成を実現することができる．コンフィギュレーションRAMは，キャッシュを伴っており，外部からの転送も可能で，全体として多数のコンテキストの切り替えが可能である．

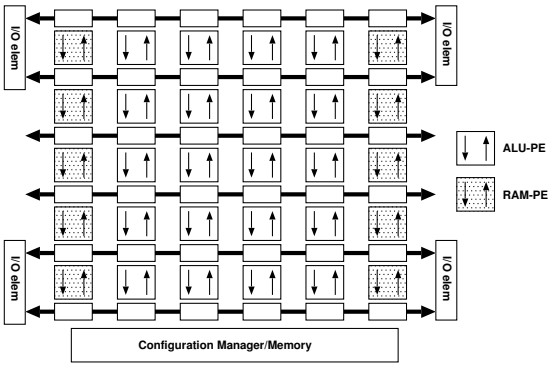

図 3: PACT Xpp のアレイ構造

4.1.2 Elixent 社 DFA1000

Elixent 社の DFA1000[24] は，4bit 構成の ALU を 128 から 2048 個，チェス盤状に並べた構成を持つ D-Fabrix に入出力用バッファと I/O を接続した構成を

持つ．D-Fabrics は，図 4に示すように 4bitALU およびレジスタから成るユニットと RAM ベースのスイッチングマトリクスが互い違いに配置される．Xpp 同様数十 μsec オーダの再構成が可能である．

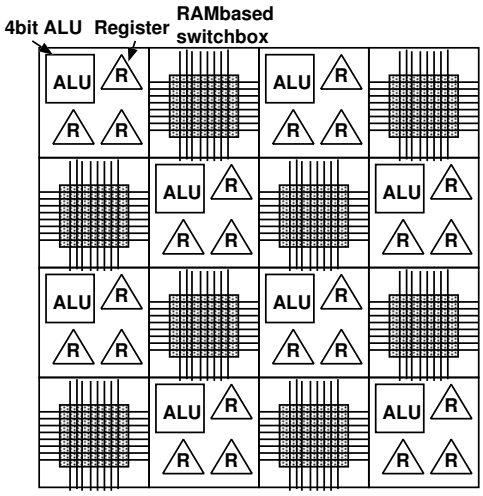

図 4: Elixent 社 DFA1000 の D-Fabrix

4.1.3 NTT PCA-2

PCA(Plastic Cell Architecture) は構成情報配送方式に分類してはいるが，他とは全く異なった思想に基づくユニークなダイナミックリコンフィギャラブルデバイスである．PCA(Plastic Cell Architecture) は，モジュール化された論理回路やメモリ等の処理単位である「オブジェクト」とそれらの「メッセージ」通信による協調作業により論理回路の自律再構成と並列処理を実現するアーキテクチャである．PCA-2[21] は 2000 年に開発されたプロトタイプ 1 号機の後を継いで，2003 年に発表されたプロトタイプである．可変構造のプラスチックパートと，プラスチックパートの制御と，周辺との交信を受け持つビルトインパートから成る PCA セルを 144 個持つ．Sea-of-LUT と呼ぶ細粒度構成を持ち，しかも全体が非同期で動作する点がとりわけユニークな特色である．非同期動作と交信のメッセージ化により，ハードウェアのプロセスを自己生成することが可能である．PCA-Chip2[?] など，PCA に基づく様々な構成が検討されている．

4.1.4 命令構成情報配送方式のまとめ

これらのダイナミックリコンフィギャラブルプロセッサは，再構成の時間が長いことから，一定の機能を複数のコンテキストで実現することは困難である．この点で，単一の機能の実現に対して，データパスの構成を切り替えつつ問題を解いていくのが自然な DRP-1 に比べると，むしろ，汎用 FPGA に近い使い方が想定される．すなわち，一つのコンテキストにまとまった機能を実装し，動的再構成により機能を高速に切り替えることで，高い面積効率を実現する方式である．マルチコンテキスト型も含めたこれらのデバイスの位置づけを図 5に示す．

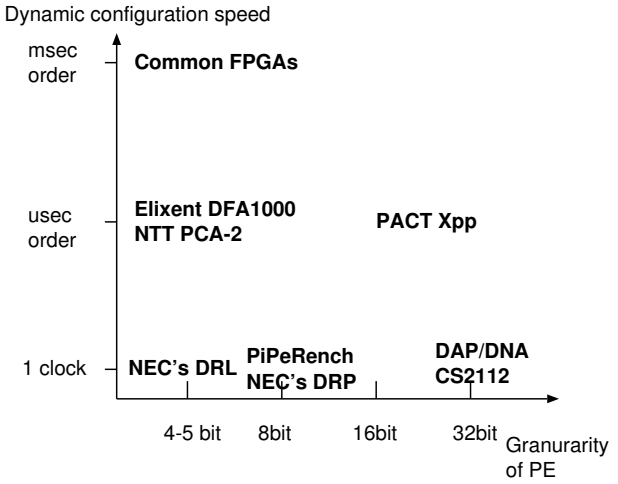

図 5: 命令/構成配送方式の位置づけ

4.2 マルチコンテキスト方式

4.2.1 NEC DRP-1

2002 年に NEC により発表された Dynamically Reconfigurable Processor[13] は，粗粒度のマルチコンテキストデバイスであり，そのプロトタイプチップ DRP-1 は PCI インタフェース，乗算器，DRAM インタフェースなどを周囲に持つシステム LSI である．

Core 部分には Tile と呼ばれるリコンフィギャラブルユニットが 4×2 個配置されており，中央にチップ全体の状態遷移を制御するためのシーケンサである Centeral State Transition Controller (CSTC) が配置されている．各 Tile は図 6は，8×8 の Processing Element (PE) アレイ，状態制御を行なうシーケンサである State Transition Controller (STC) から構成される．また，8bit×256 エントリのメモリ 8 セットとこれらを制御するメモリコントローラ 2 セットを両側に持ち，8bit×8192 エントリのメモリ 4 セットとメモリコントローラを Tile の上部もしくは下部に持つ．DRP 全体では 8bit×256 エントリのメモリを合計 80 セット，8bit×8192 エントリのメモリを合計

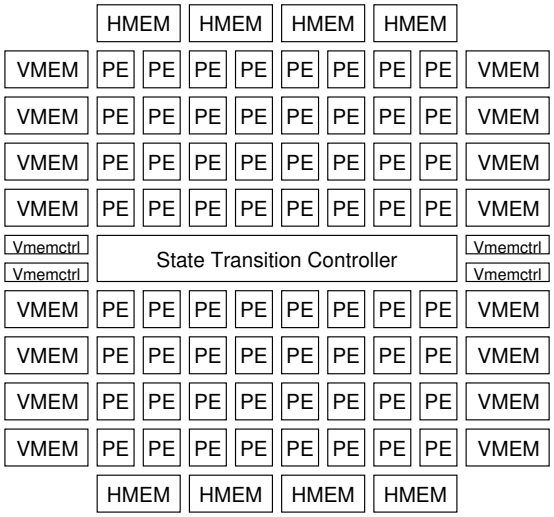

図 6: Tile の構成

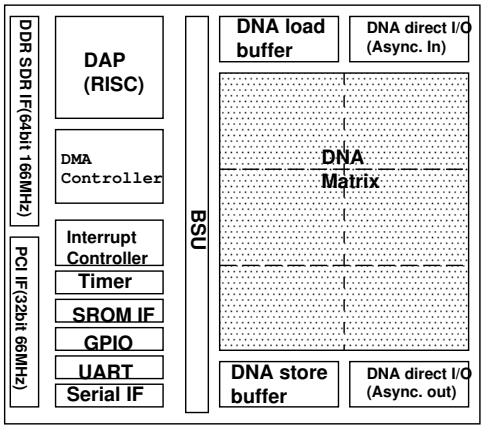

図 7: DAP/DNA の全体構成

32 セット持つ.

PE の構造は前節に紹介した通り，図 2に示す 8bit 構成であり，全ての PE が同一構造のホモジーニアスな構成となっている.

DRP は内部のメモリに最大 16 コンテキスト分の情報を蓄えることができ，最大で 5 コンテキストの中から次のコンテキストを選択し，動的に再構成をすることができる. 次のコンテキストの選択は STC に信号を送ることで行なう. さらに，動作中に，切り替えの対象外のコンテキストに対してコンフィギュレーション ロードが可能である.

4.2.2 IPFlex DAP/DNA

IPFlex 社が 2002 年に発表した DAP/DNA[15] は図 7 に示すように，32bit の専用 RISC DAP とデータプロセシングエレメントを実装した並列実行エンジン DNA マトリクス，および周辺インタフェースから構成される. プロトタイプである DAP/DNA-1 の経験に基づき，並列演算ユニットを 368 個に増やした商用チップ DAP/DNA-2 の開発が終了している.

32bit RISC である DAP は，条件付き実行命令などを含む 128 種類の命令を持つコアであり，命令，データそれぞれ 8K バイトのキャッシュを装備している. DNA とのバッファを用いて高速にデータ転送を行なう機構，同期命令や非同期実行機構を持ち，効率の良い処理の分担を実現している.

一方，DNA マトリックスは，図 8に示すように 7 種類のプロセッシングエレメントを 2 次元アレイ状に接続した構成を持つ. プロセッシングエレメントには，算術論理演算を実行する EXE エレメント，データパス間の遅延調整用の DLE エレメント，データ保持用の RAM エレメント，アドレス生成用の C16E, C32E エレメントなどがあり，これらが一定の間隔で配置されている. すなわち，DNA マトリックスはヘテロジーニアスな構造を持つ. PE 間には，32 ビットのデータバスとキャリーバスが張り巡らされており，どのように PE を接続しても 166MHz の動作周波数が保証される. DNA コンフィギュレーションメモリは 3 バンクで構成され，1 クロックで変更可能な情報を 3 面分保持することができる. また，コンフィギュレーションデータは DMA 転送を用いてバックグラウンドで DNA コンフィギュレーションメモリへ高速転送が可能である.

4.2.3 CMU PipeRench

1997 年から CMU で続けられているプロジェクト [22] で 2002 年にチップが完成し，商用化が検討されている.PiperRench は，8bit 構成の演算器，レジスタからなる PE を 16 セット並べた Stripe と称する帯状の構造を基本とし，これらをインターコネクションを介して直線状に並べた構造を持つ. 末端のインターコネクションは最初の Stripe にフィードバックされ，全体でループ構造を実現する.

PipeRench の再構成は，Stripe 単位に行なわれ，パイプライン処理の流れに沿って最後に用いた Stripe が新しい構成に入れ替わることにより，仮想的に長大な構造のパイプラインを実現する.2002 年の開発されたチップでは実 Stripe を 16 持ち，これを用いて最大 256 の仮想 Stripe を実現可能である.

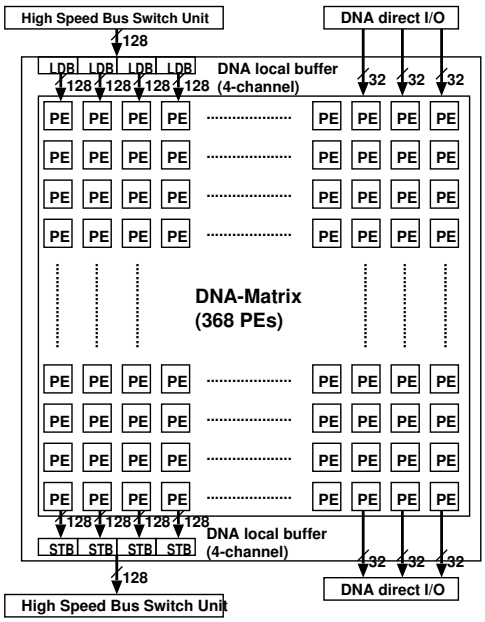

図 8: DNA Matrix の構成

4.2.4 マルチコンテキスト方式のまとめ

IPFlex 社の DAP/DNA は，マルチコンテキスト方式の採用により，実行中に次のタスクの構成情報を設定することにより，タスク間の待ち時間なしのコンテキストスイッチが可能である. このため，命令/構成情報転送方式で問題であったタスク間の切替えオーバヘッドを無くすることができる. しかし，コンテキスト数が少ないため，単一タスクを複数コンテキストで実現することは難しい.

これに対して，DRP は，16 コンテキストを頻繁に切替えることにより，単一タスクを複数コンテキストで処理することが可能である. すなわち，各プロセッシングエレメントの機能とスイッチの接続情報を長大な命令と考えると巨大で再構成可能なデータパスを有する VLIW(Very Long Instruction Word) 型コンピュータとして見ることができる．しかし，もちろん通常の VLIW 型に比べて実行可能な命令数 (コンテキスト数) が限られているため，同一コンテキスト内の並列処理と繰り返し処理を十分利用する必要がある.

構成ユニットの粒度 (Granurarity)，数 (Parallelism)，細粒度時間多重度 (Fine grain time multiplexing) により，マルチコンテキスト方式のダイナミックリコンフィギャラブルプロセッサ，VLIW 型，汎用 FPGA を位置づけ，図 9に示す．あると考えることができる.

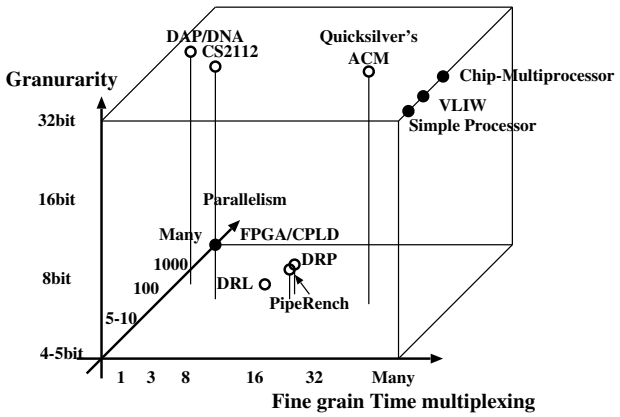

図 9: マルチコンテキスト方式の位置づけ

4.3 オンチップマルチプロセッサとの境界タイプ

今まで紹介してきたダイナミックリコンフィギャラブルデバイスの構成要素は，プロセッシングエレメント (PE) ではあるが，それ自体が独立に動作するわけではなく，他の PE と組み合わせて全体として巨大なデータパスを構成する. ところが，構成要素を自律的に動作可能なプロセッサとし，プロセッサ同士の接続，メモリとの接続などをリコンフィギャラブルにしたチップも登場している．これらのチップは厳密にはリコンフィギャラブルデバイスというよりチップマルチプロセッサである. しかし，これらのデバイスは，従来の高性能汎用チップマルチプロセッサよりも，以下の点で，今まで紹介してきたダイナミックリコンフィギャラブルプロセッサに近い.

- 対象とするアプリケーションが組み込み用途であり，専用ハードウェア，DSP の代替を狙っている. このため，それぞれのプロセッサで実行するプログラムは極めて限定された機能を実現するに過ぎない.
- 簡単なプロセッサを多数用いたアレイ構造を取る.
- 特定用途での性能価格比をあげるため，ヘテロジーニアスな構成を取る場合が多い.
- 全体の処理と接続を切替える「コンテキスト」の概念が存在する.

Quicksilver 社から発表された ACM[14]，MorphTech 社の rDSP[26]，picoChip 社の PC101[25] な

どが，これらの分類に入る．ここでは，ACM のみ簡単に紹介する．

4.3.1 Quicksilver ACM

米国 Quicksliver 社から 2002 年に発表された ACM は [14]，算術計算用，ビット処理用，状態マシン，通常の RISC プロセッサなど様々な種類のプロセッシングノードを H-tree 状のネットワークにより接続した構成を持つ. ノードの一例を下記に紹介する．

- AXN : 16 個の 8bit 乗算器，4 個の 16bit ALU，8 個のアキュムレータ，64 個のレジスタ，データキャッシュ等で構成された強力な並列演算ユニット．接続構成および制御がプログラム可能である．
- AN2： 単純な汎用信号処理用のノード.AXN ほど並列処理ができない用途に用いる．
- DAN : FFT/IFFT, DCT/IDCT に特化されたノード
- DBN : Bit 処理用ノード.Viterbi コード，Turbo コード等のエンコード／デコード用ノード
- DFN : FIR Filter 専用ノード
- PSN : 汎用 32bit RISC

これらのノードは，MIN(Matrix Interconnect Network) と呼ばれるオンチップネットワークにより接続される. データは，1 フリットのパケットの形でノード間を転送され，各ノードの入力 FIFO に格納される．ストリーム処理に特化しており，各タスクをそれぞれのノードにマップして並列，パイプライン処理を行う．ノードの組合わせは，アプリケーションに異存してさまざまに選ぶことができ (もちろん一度チップを作ってしまえば機能は変えられない)，その意味でこのシステムはコンフィギャラブルなアプローチをとっている．

それぞれのノードは，プログラマブルであるが，このプログラムを一斉に，あるいは非同期に切り替えることが可能である. この切り替えがコンテキストスイッチであり，ACM は 31 コンテキスト分を保持する命令メモリを持つ. コンテキスト切り替えによって，それぞれのノードで実行するタスクの切り替えを行う. この辺は，ダイナミックリコンフィギャラブルプロセッサと類似している点である．

4.4 プログラム開発環境

ダイナミックリコンフィギャラブルプロセッサは，開発の最初の段階では，それぞれのノードの PE の機能や接続をそのまま RTL で記述したり，図形エディタ等で記述する方法が取られている．しかし，これらの方法は，チップ依存性が強く，生産性が悪いため，普及するためには，高級言語でのプログラミングが不可欠であり，各チップ共，C 言語での設計環境を開発中である．ダイナミックリコンフィギャラブルプロセッサは，VLIW 型汎用プロセッサと FPGA の中間的な性質を持つが，どちらかというと FPGA 側に近いものが多い. このため，コンパイラのベースとなる技術は，一般的な計算機用のコンパイラというよりも，ハードウェアの合成技術に基づくものとなる．

一例として，NEC の DRP-1 用に開発された DRP コンパイラ [33] は，C 言語を拡張した BDL をフロントエンドとし，ASIC 用の動作エンジンをベースにした動作合成部により，制御回路と複数コンテキスト用データパス回路を発生する. さらにバックエンド合成部で，マッピングおよび配置配線を行い，コンテキスト制御用の STC コードと PE アレイコードを生成する．BDL は通常の C コンパイラでコンパイル可能であり，設計の最初の段階で，アルゴリズムを C 言語レベルで検証する. 次に，合成の各段階で，対応する Verilog コードが生成されるため，通常の論理シミュレーションツールと組み合わせて，動作確認が可能である．最終的には，DRP の PE の詳細レベルのシミュレーションが可能であり，このレベルで動作すれば，ほぼ実チップ上での動作が保証される．

この合成ツールを組み込んだ統合設計環境 Musketeer は実チップのデバッグ機能をも内蔵しており，実チップ上で複雑な動作を行った際の，コンテキスト，レジスタ，メモリ内容の表示等が可能である．

5 おわりに： リコンフィギュラブルな世界へのお誘い

ダイナミックリコンフィギャラブルプロセッサの中でも、特にマルチコンテキスト型の一部は、今までのアーキテクチャの枠からはみだす計算機構である。この計算機構では、単一コンテキスト内でデータをパイプライン処理、SIMD 型並列処理、シストリック型並列処理を行うことができる点で、FPGA 的である一方、次々にコンテキストを切り替えることで

データパスの構造自体を変えることができることから、コンテキストという巨大な命令を持つ VLIW 型プロセッサとして考えることもできる。時空間的に三次元的で、しかも柔軟性の高いあらゆる種類の並列処理を実現することができる。

現状で、ある程度アプリケーションの実装経験が溜まり [34]、方向性も見えては来ているが、依然としてアーキテクチャが固まっているとはいえず、逆に考えるとあらゆる方向に発展の余地が残されている。PE アーキテクチャの最適化、コンテキストの制御手法、ネットワークスイッチ、ストリーム入出力制御等をどのように行えばよいかがはっきりせず、抽象化、評価モデル化も不足している。マルチコンテキスト型でないものも、ヘテロジーニアスなオンマルチプロセッサの境界上のシステムはアーキテクチャ的な研究がほとんど行われておらず、計算機アーキテクチャの研究対象として、広大かつ豊穣な舞台であるといえる。

この世界では、従来の汎用プロセッサアーキテクチャの世界では受け入れられそうもない技術がまじめに研究されている。仮想ハードウェア [17][36] は既に、実現レベルの技術となっており、自動的に構成自体を最適化する動的最適化手法 [35][37] なども実際に試されている。PCA の思想では、ハードウェア自律的に fork をし、自己増殖を行いながら処理をしていく。その他あらゆる Fancy なアイディア [38] の舞台となり得る。

SACSIS に参加されている計算機アーキテクトの皆様、どうか **汎用 CPU でない**とか **怪しげ**とか馬鹿にせず、この分野へ参入されてはいかがであろうか。最近の汎用プロセッサアーキテクチャと違って、この分野ではアーキテクトのやるべき膨大な仕事が残されている。また、日本の技術が世界に先行している分野でもあり、アイディアが実際に製品化されて世の中に使われるチャンスも多い。また、コンパイラ技術者の皆様、この種のダイナミックリコンフィギャラブルプロセッサを対象とするのはいかがだろうか？さらに、ハイパフォーマンス技術者の皆様、とりあえず汎用 FPGA を用いたリコンフィギャラブルシステムは今まさに数値計算の沸騰期を迎えつつある。また、ダイナミックリコンフィギャラブルプロセッサは、現在のところ、面積効率を狙ったアプローチだが、もちろんハイパフォーマンスなダイナミックリコンフィギャラブルも考えられる。この辺を研究対象としてはいかがだろうか？

リコンフィギャラブルシステム研究会 [31] は、9 月に総会 (今年は 9/15-17 に長崎大)、11 月のデザインガイア、1 月の合同研究会でその成果を発表している。皆様の御参加を心よりお待ちしている。

参考文献

[1] http://www.xilinx.com

[2] http://www.altera.com

[3] 松本:”FPGA の進化と今後の FPGA 設計に求められるもの,” 信学報 VLD2002-128, 2003 年 1 月

[4] 末吉，飯田，:” リコンフィギャラブルコンピューティング” 情報処理 Vol.40, No.8, pp.777-782, 1999.

[5] J.Arnold, D.Buell, E.Davis, “SPLASH2,” Proc. of the ACM SYmposium on Parallel Algorithms and Architectures, pp.316–322, 1992.

[6] 沼昌宏:”FPGA を利用したアーキテクチャとシステム設計,” 情報処理 Vol.35, No.6, pp.511-518, 1992.

[7] 中島 他: “FPGA ベース並列マシン RASH,” 並列処理シンポジウム JSPP99, pp.222. 1999.

[8] A.Tsutsui, T.Miyazaki,”YARDS: FPGA/MPU Hybrid Architecture for Telecommunication Data processing,” Proc. of FPL (LNCS1482), pp.366–375, 1998.

[9] J.Hauser, J.Wawrzynek, “Garp: A MIPS processor with a reconfigurable coprocessor,” Proc. of FCCM, pp.12–21, 1997.

[10] H.Withlin, B.Hutchings, “Sequencing run-time reconfigured hardware with software,” Proc. of FPGA, pp.122–128, 1996.

[11] Z.Ye, et.al. “CHIMAERA: A high performance architecture with a tightly coupled reconfigurable functional unit,” Proc. of FCCM.pp.225-235, 2000.

[12] X.Tang, M.Aalsma, R.Jou, “A compiler directed approach to hiding cofiguration latency in Chameleon processors,” Proc. of FPL, pp.29–38, 2000.

[13] M.Motomura:”A Dynamically Reconfigurable Processor Architecture,” Microprocessor Forum, Oct. 2002.

[14] P.Master: ”The Age of Adaptive Computing Is Here,” Proc. of FCCM, pp.1-3 (2002).

[15] 佐藤:“アイピーフレックスのリコンフィギャラブルプロセッサデバイス,” 第 1 回リコンフィギャラブルシステム研究会予稿集, 2003 年 9 月

[16] U.J.Kapasi, et. al, “Programmable Stream Processors,” Computer, pp.54-62, Aug. 2003.

[17] X.-P. Ling, H. Amano, “WASMII: A Data Driven Computer on a Virtual Hardware” Proc. FCCM, pp. 33–42, (1993).

[18] S. Trimberger, D. Carberry, A. Johnson and J. Wong “A Time-Multiplexed FPGA” Proc. FCCM pp.22-28, (1997).

[19] Fujii, T., et.al.: “A Dynamically Reconfigurable Logic Engine with a Multi-Context/ Multi-Mode Unified-Cell Architecture,” Proc. Intl. Solid-State Circuits Conf., pp.360–361 (1999).

[20] Application I,II, Arithmetic I,II, Prof. FCCM 2004.

[21] 伊藤 他: “動的再構成可能論理 LSI PCA-2,” 第 1 回リコンフィギャラブルシステム研究会予稿集, 2003 年 9 月

[22] H.Schmit, et.al. “PipeRench: A Virtualized Programmable Datapath in 0.18 Minron Technology,” Proc. of IEEE CICC, (2002).

[23] http://www.pactcorp.com

[24] http://www.elixent.com

[25] http://www.picochip.com

[26] http://www.morphtech.com

[27] T.Miyamori and K.Olukotum, “REMARC: Reconfigurable Multimedia Array Coprocessor,” IEICE Trans. Inf. & Syst. , Vol.E82-D, No.2, pp.389–397, (1999).

[28] K.Tanigawa, et.al.: “PARS Architecture: A Reconfigurable Architecture with Generalized Execution Model,” IEICE Trans. Inf. & Syst. , Vol.E86-D, No.5, pp.830–840, (2003).

[29] H.Singh, et.al.: ”MorphoSys: An intergrated reconfigurbale sytem for data-parallel and computation intensive applications,” IEEE Trans. Computers, Vol.49, pp.465–481, (2000).

[30] Caspi, E. et al.: “Stream Computations Organized for Reconfigurable Execution (SCORE)”, Proc. of FPL2000, pp. 605–614 (2000).

[31] リコンフィギュラブルシステム研究会ホームページ http://slab.cis.nagasaki-u.ac.jp/reconf/

[32] 天野:” ダイナミックリコンフィギャラブルプロセッサの研究開発動向,” 信学報 DSP2003-133,2003.10.

[33] 粟島 他:” 動的再構成可能チップ DRP の C コンパイラ,” 信学報 VLD2003-118, 2004 年 1 月

[34] N.Suzuki, et.al.,”Implementing and Evaluating Stream Application on the Dynamically Reconfigurable Processor,” In Proc. of FCCM2004.

[35] H.Amano, A.Jouraku, K.Anjo, “A dynamically adaptive switching facrics on a multicontext reconfigurable device,” Proc. of FPL 2003.

[36] 天野、犬尾、紙,”DRP 上での仮想ハードウェア機構の検討:” 信学報 VLD2003-122. 2004 年 1 月

[37] 吉田、曽我、村上、林田:”DAP/DNA を用いた SysteMorph プロトタイピング,” 第 1 回リコンフィギャラブルシステム論文集, pp.221-226.

[38] 中島浩 :”Fancy Architecture の数打ちゃ当たる,” 情処アーキテクチャ研報 No.158,pp.55-56. (150 回記念パネル), 2004 年 4 月