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

2003－ＳＬＤＭ－111（25）
2003／10／24

［特別招待論文］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

天野　英晴†

† 慶應義塾大学大学院理工学研究科　〒223-8522 神奈川県横浜市港北区日吉3-14-1
E-mail: †hunga@am.ics.keio.ac.jp

あらまし　ダイナミックリコンフィギャラブルプロセッサは、データパス構造を短期間に切り替えることにより、高い面積効率と柔軟性と高速処理を実現する。本稿では、これらのシステムをサーベイし、登場の背景、特徴、位置づけをまとめる。
キーワード　ダイナミックリコンフィギャラブルプロセッサ

A Survey on Dynamic Reconfigurable Processors

Hideharu AMANO†

† Dept. of Information and Computer Science, Keio Univ.
3-14-1, Hiyoshi, Kohokuku, Yokohama, 223-8522, Japan
E-mail: †hunga@am.ics.keio.ac.jp

Abstract　Dynamic reconfigurable processors achieve a high degree of area efficiency, flexibility and performance by changing their data-path structure quickly. Recent trends of such processors are surveyed.
Key words　Dynamic Reconfigurable Processors

1. はじめに

単一シリコン上に簡単なCPUと様々な専用ハードウェアを混載するSoC (System-on-a-Chip)は、メディア処理、通信制御等広い分野で用いられ、DRAMに代って日本の半導体産業の主力となりつつある。SoCは、C言語レベル記述に基づくソフトウェア／ハードウェア協調設計を用いて、ハードウェア化する処理を適切に分離することで短い開発期間で低コスト、高性能、低消費電力のASICをそれぞれ目的用途に応じた市場に供給する。

しかし、SoCの対象とする分野の広がりと急速な技術的発展に加え、半導体プロセス自体の発展によって以下の問題が表面化しつつある。

- 対象分野が多様化するに伴い、様々な種類のASICを分野に応じて作成する必要が生じた。これにより、比較的生産量が少ないチップを多種類開発する必要が生じている。
- 対象分野の発展が急速で、新しい規格、方式が次々に開発されるため、ソフトウェア／ハードウェア協調設計を効率的に用いたとしても、適切な時期に市場に投入することが難しくなっている。
- 半導体プロセスの進歩に伴い配線遅延の増大が表面化し、フロアプラン、レイアウト、クロック配線等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、上流設計を短期間で行なうことによる、チップ化までの時間とコストの削減効果が小さくなってきた。

これらの問題点を解決するアプローチの一つとして、ダイナミックリコンフィギャラブルプロセッサが注目されている。ダイナミックリコンフィギャラブルプロセッサは、8bit-32bitの演算器、シフタ、レジスタ、分散メモリから成る処理ユニットを多数搭載し、これら処理ユニットの構成および接続を短時間で切り替えることにより、柔軟性と高速性を優れた面積効率で実現することができる。昨年以来、NECのDRP [4]、IPFlexのDAP/DNAチップ [6]、QuicksilverのACM [5]などのそれぞれの特徴を持つチップが開発されている。

本稿では、ダイナミックリコンフィギャラブルプロセッサが登場した背景、特徴と位置づけ、問題点および今後の発展について概観する。

2. なぜダイナミックリコンフィギャラブルなのか？

2.1 登場の背景：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、コード化された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。これに対応して、SoCでは、演算能力を要しないが複雑な処理を担当する低コストの組み込み用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ないため、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。ところが、最近のプロセスは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、特に下流設計に多大な時間を要する。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(注1)。

専用ハードウェアに代わる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、開発コストを削減することができる。さらに、単一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、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。

2.2 CPU+FPGAではなぜだめなのか？

それでは、柔軟性を持ったハードウェアとして、汎用のプログラム可能なデバイスであるFPGA(Field Programmable Gate Array)やCPLD(Complex Programmable Logic Device)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、Altera社のStratix [2]など性能面でASICに迫る製品や、Xilinx社のSpartan [1]など価格面で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ARMとFPGAを混載したExcalibur [2]やPowerPCを混載したVirtex-Pro [1]は既に市場に登場しつつあり、CPU+FPGAのアプローチは既に現実のものとなっている。

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Product term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。

- 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]。
- 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。
- 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(注2)。

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、おいそれとは手を出すことができない領域である。

現状では、FPGAベンダによる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特殊な用途に限らており、この状況はしばらく変化はないと考えられる。

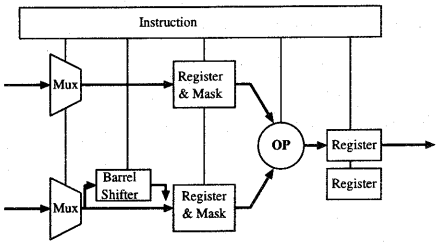

図1 CS2112の構成要素DPU

2.3 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、Chameleon CS2112の構成ユニットであるDPUを示す。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。図3のz軸方向に各ダイナミックリコンフィ

(注1)：多種類のASICを開発することが、その分ベンダーの利益に繋がれば「どんどんいろいろ作らなければならないこと」によりベンダーは幸福になるのだが、現実はそうなってはいない。

(注2)：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。

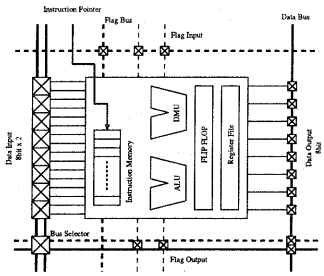

図2　DRPのProcessing Element

ギャラブルデバイスの構成要素のビット幅を示す。ビット幅が小さい程、面積効率と柔軟性は高いが、ビット幅が大きくなった場合の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、後に述べるように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。

2.4　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うことにより、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不利となる。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。メディア処理に限らず、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、チップ内部に複数の回路構成情報(コンフィギュレーション)を用意しておき、これを次々に切り替えながら、つまり動的に再構成しながら処理を進めていくことにより、少ない面積で大きなハードウェアと等価な処理を実現可能にする。すなわち、全体の処理を行なう回路を、時分割多重化して実装することで少ない面積で処理を行なうことができる。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、面積は小さくして上で、性能低下をかなりの程度に抑えることが可能である。これがダイナミックリコンフィギュレーションによる面積効率増加の原理である。

ダイナミックリコンフィギュレーションの具体的な実現方式には以下の考え方がある。

○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[8][9]、NECからは1999年にDRL[11]として実現された。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[9]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。この方式では、複数セットの構成情報格納用メモリが、FPGA上の構成要素ブロックの近傍に置かれ、動作前にあらかじめ設定されていると考える。

マルチコンテキスト方式は、粗粒度構成にも適用可能した場合、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[10]でむしろ有利である。特に32bitなどの粒度が大きい構成ユニットの場合、ALUやシフタなどに対する設定情報は、ハードウェア構成情報(Configuration)と呼ぶよりは、命令(Command/Instruction)と呼ぶ方が相応しい場合もある。この場合、プロセッシングエレメント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、接続と共に切り替えるという考え方も可能である。Chameleon CS2112などでは、この考え方を取っており、NEC社のDRPでも同様にPEの構成設定情報のことを命令と呼んでいる(図2)。ただし、DRPは、PEの粒度が小さく柔軟性が高いため、命令コードというよりはハードウェアの構成情報に近いため、それぞれの構成のことをハードウェアコンテキストと呼んでいる。DRP-1は、16コンテキストを保持し1クロックでコンテキストスイッチ可能である。

○命令配送方式

32bit等の粒度が大きく、比較的単純な構造の構成ユニットを結合網で接続した構成を持つ場合、構成情報と呼ぶよりは、プロセッシングエレメントに対する命令と、それらの接続経路の設定命令と考えられる。これらの命令をチップ内の特定のメモリ内に格納しておき、これらを一定のタイミングで各プロセッサや接続スイッチに配送することにより、ダイナミックリコンフィギュレーションが可能となる。この手法は、マルチコンテキスト方式に比べて、命令を集中して管理し、読み出して、配送するという点で、よりプログラム格納型に近い考え方となる。PACT Xpp[12]はこれに近い方法で再構成を行なっている。この方法では、命令(構成情報)メモリは集中実装されるため、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。しかし、この方法では1クロックでの構成変更は困難で、再構成には数クロックを要することになり、命令の配送に要する配線を確保する必要がある。

上記二つの方式は、一見相当雰囲気は違うが、実際は、どの程度ロジックの近傍に構成情報(命令)を保存するメモリを分散するか、という実装上の問題に他ならない。ただし、マルチコンテキスト構成は、粒度に依らず用いることができるが、命令

分配方式は、8bitなどの比較的粒度が細かい構成を取る場合、構成情報の量が大きいために実装が困難である。

2.5 VLIW型との関係

ダイナミックリコンフィギャラブルプロセッサの構成ユニット(プロセッシングエレメント)の粒度を32bitにとり、構成を簡単化すると共に、構成ユニット自体の数を減らしてさらに時間多重度を高めた場合、構成情報が簡単化される一方、一つのコンテキストで実行可能な作業はきわめて制限される。このため、複雑な処理を実行するためには、単純な構造のコンテキストを、短時間でどんどん切り替えて実行する必要が生じる。これを命令配送方式で実現した場合、この実行方式は、限りなく「VLIW(Very LongInstruction Word)型コンピュータ」に近づく。さらに要素を減らして、構成ユニットの数を1にすると、これはもうただのプログラム格納型コンピュータになってしまう。

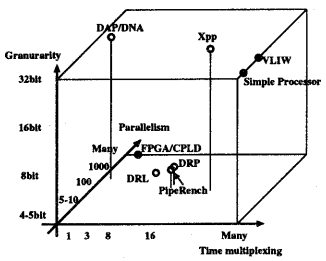

図3 ダイナミックリコンフィギャラブルデバイスの位置づけ

すなわち、構成ユニットの粒度(Granularity)、数(Parallelism)、時間多重度(Time multiplexing)により、図3のような位置づけを考えることができる。すなわち、構成ユニットの粒度を32bit、数を1、時間多重度を極大に取れば単純なプログラム格納型コンピュータとなり、粒度を4-5bit、数を極大、時間多重度を1とすると通常のFPGAとなる。ダイナミックリコンフィギャラブルプロセッサは、この中間的な位置をカバーすることになり、用途とコスト、性能を考えて、適切な位置を選ぶことが重要である。どちらかの極に近すぎる構成は、確固たる地位を築いている方式にその特徴が近づくことになり、競争に勝つことが難しくなる。例えば、時間多重度を高めすぎて構成ユニットの数が少なすぎる構成は、命令フェッチやメモリとの接続が洗練されたVLIW方式に対してメリットを見い出すことが難しい。逆にコンテキスト数が少なすぎる場合、演算器や分散メモリを混載したFPGAとの競争に敗れる可能性がある。

2.6 ダイナミックリコンフィギュレーション関連技術

ダイナミックリコンフィギュレーションは、以下の技術と組み合わせることでさらにその可能性を広げることができる。

○ランタイムリコンフィギュレーション:

ランタイムリコンフィギュレーションあるいはオンザフライリコンフィギュレーションとは、処理中に外部から構成データを読み込む手法である。一般的なFPGAは、単一のコンテキストであるので、全体をいくつかのブロックに区切って一部を動作させながら、動作させていない部分に対して構成データを読み込む。この方法を部分再構成(パーシャルリコンフィギュレーション)と呼ぶ。しかし、マルチコンテキスト構成であれば、利用していないコンテキストに対して、動作中に新たな構成情報データを読み込むことで、ランタイムリコンフィギュレーションが可能である。「命令を分配する」と考える場合は、動作中に新たな命令を外部からプリフェッチあるいはキャッシュするという考え方になる。

この手法を利用すれば、チップ内の構成情報メモリに入らない場合に、コンテキストを外部から先行フェッチすることで、仮想的に巨大な対象回路を実行することが可能になる。この方式は、仮想ハードウェア(Virtual Hardware)[8]と呼ばれ、論理エミュレータでの時分割実行方式と類似している。しかし、ダイナミックリコンフィギュレーションは、コンテキストを短期間で切り替える場合が多いことから、構成情報メモリへのフェッチ時間がボトルネックになり、効率の良い実行方式はまだ研究段階である。この機能により多数の設計から、状況に応じて適当なものを選択する動的適応型[7]の手法も利用可能となる。

○クラスタリング/部分コンテキストスイッチ:

コンテキストスイッチは、チップ全体に渡って行なうと、柔軟性を妨げる可能性がある。そこで、全体をいくつかのブロックに区切り、この単位でコンテキストスイッチを制御できる機能が便利である。例えばNEC社のDRPではこの単位をTileと呼び、DRP-1では8つのTileがそれぞれ独立にコンテキストスイッチ可能であり、独立にランタイムコンフィギュレーション可能である。この部分コンテキストスイッチにより、チップ上にいくつかの独立したモジュールを設けることができ、柔軟な処理が可能になる。PACT XppのPAC(Processor Array Cluster)も同様なブロック化構造である。CMUのPipeRench[13]は、コンテキストスイッチの単位をStrideという直線構造にしている点で独自の特徴を持つ。

部分コンテキストスイッチ機能を持つダイナミックリコンフィギャラブルプロセッサは、独立した命令をそれぞれ実行可能なプロセッサを複数搭載したオンチップマルチプロセッサとして考えることもできるようになる。

3. サーベイ

本節では代表的なリコンフィギャラブルプロセッサアレイを簡単に紹介する。詳細および他の様々な提案[16][18][17][19]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。

a) NEC DRP

2002年にNECにより発表されたDRP[4]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8×8のPro-

cessing 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32セット持つ。

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。

b) IPFlex DAP/DNA

IPFlex社が2002年に発表したDAP/DNA [6]は32bitの専用RISC DAPと144個の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マトリクスから構成される。さらに並列演算ユニットを368個に増やしたDAP/DNA-2を開発中である。

32bit RISCであるDAPは、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、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。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。

c) PACT Xpp

ドイツのPACT Informations Technologie社が発表したXpp [12]は階層的な構造を持つ再構成可能なプロセッサアレイである。32bitのALU, 転送用レジスタセットから成るPAEを2次元のアレイ状に接続してクラスタを作る。このクラスタを複数スイッチにより接続することで階層構想を実現する。Xppはデータの他にもコンフィギュレーション用の2次元バスを持っており、このバスの末端にコンフィギュレーション用RAMを装備している。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。このため、DRPやDNAマトリックスと異なり1クロックでの再構成はできないが、マイクロ秒オーダで事実上数の制限なしに再構成を実現することができる。

d) Quicksilver ACM

米国Quicksliver社から2002年に発表されたACMは[5]、算術計算用、ビット処理用、状態マシン、スカラプロセッサの4種類のプロセッサによりクラスタを構成し、これらのクラスタを4つずつまとめて、ツリー構造(H-tree)で接続した構成を持つ。これらのプロセッサはそれぞれ再構成可能な構造になっており、例えば算術計算用のプロセッサは、その内部に複数の32bit演算装置を持ち、その演算内容と内部の接続を設定することができる。これらの構成情報は、オンチップのRAM上に格納され1クロックでの再構成が可能である。ACMは現状ではあまり詳細なデータが公表されていないが、実際の利用は既に始まっており、応用例の公開が待たれている。

e) CMU PipeRench

1997年からCMUで続けられているプロジェクト[13]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。

f) NTT PCA-2

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 [14]は2003年に発表された二つ目のプロトタイプであり、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でここで紹介した他のリコンフィギャラブルプロセッサアレイとは全く異なったユニークな構造を持つ。大学でもPCA-Chip2 [15]など、PCAに基づく様々な構成が検討されている。

4. C言語によるプログラミング

4.1 高位合成ツールの重要性

ダイナミックリコンフィギャラブルプロセッサが、ストリーム処理、メディア処理を対象とし、現在のSoCの専用ハードウェア部に対する面積効率と柔軟性の優れた置き換えを狙っていると考えると、このプログラミングの方法としてC言語エントリを備えているのは、きわめて自然である。まとめると以下の通りである。

- ストリーム処理、メディア処理では、複雑だが計算能力を有しない処理と強力な計算能力を必要とする処理が混在する場合が多いため、ハードウェア/ソフトウェア協調設計が必要である。このためにはC言語を用いた高位レベルの設計が有利である。

• HDLによる記述は、それぞれのコンテキスト内の構成の最適化には有利だが、コンテキストの切り替えによる実行を表現することが難しく、特に多数のコンテキストを次々に入れ替えながら実行する場合には記述がたいへんである。

したがって、ダイナミックリコンフィギャラブルプロセッサが成功するかどうかは、そのC言語レベルでの設計環境にかかっている。特にC言語で記述されたプログラムを、コンテキスト数、それぞれのコンテキストが実現できるハードウェア量、コンテキスト内の並列処理を考えて、高位合成してPEに割り付けるコンパイラが重要である。

このコンパイラは、VLIW型のプロセッサ用のコンパイラよりも、通常のSystem-CなどでASICを設計する場合の高位合成ツールに近い機能が必要である。これは、VLIW型のプロセッサのバックエンドは、単一メモリモデルに基づいていて、レジスタファイル間のデータに対して演算を行なうことが基本となっているのに対して、ダイナミックリコンフィギャラブルプロセッサの場合は、分散メモリ上あるいは入力からのストリームに対して並列に演算処理を行なう構成を生成する必要があるためである。

しかし、一方でASIC用の高位合成ツールと比べると、ダイナミックリコンフィギャラブルプロセッサは、時間多重機構、すなわちコンテキストの切り替えが可能であることから、どの程度の処理を一つのコンテキストで実行し、どのタイミングでコンテキストを切り替えるかを判断する必要が生じる。現状ではコンテキスト内のハードウェア資源やコンテキストの最大数はさほど多くないため、高位合成ツールは、厳しい制限付きで一つ次元の増えたトレードオフを考慮しなければならないことになる(注3)。ただし、ASIC設計と異なり、チップ内に収まってしまえば、面積すなわち利用PE数をそれ以上小さくしても意味がないため、この点の制限は緩和される。コンテキストの切り替えのタイミングについては、ある程度プログラマからの指定も必要であろう。現在、DRPにおけるCyber-C [20]など、ASIC用の高位合成を基にした、優れた合成環境の開発が進められている。

4.2 データ構造の変換

SoCにおけるASIC設計同様、ダイナミックリコンフィギャラブルプロセッサにおいても、C言語で記述したプログラムがそのままの形で高速動作するわけではない。C言語で記述した通常のプログラムは、単一アドレス空間のメモリを想定しているが、ダイナミックリコンフィギャラブルプロセッサ上では、分散メモリあるいは入力バッファ中のストリームから並列にデータを取ってきて演算するようにデータ構造の変換が必要である。この作業を行なわない場合、メモリのボトルネックで並列処理が制限され、高速動作の実現が困難になる。

場合によっては、ストリームをダイナミックリコンフィギャラブルプロセッサ上のバッファやメモリに載せられる量で分割する必要が生じる。対象となるチップの構造、物理的制約を理解した上、設計者が上記の変換をうまく行なった上で、高位合成を行なえば、チップの性能を充分に発揮することができる。したがって、ダイナミックリコンフィギャラブルプロセッサのプログラマの主たる仕事は、上記のデータ構造の変換をC言語レベルで行なうこととなるだろう。

(注3)：PipeRenchのように直線状のパイプラインならばこのトレードオフは単純化される

5. おわりに

ダイナミックリコンフィギャラブルプロセッサはここに述べた以外にも消費電力についての問題、コンテキスト数の制限の緩和、NoC(Network On Chip)の導入によるブロック間ネットワークの改善、数値計算などの大きな粒度や、制御等の小さな粒度など適応範囲の拡大等、解決すべき問題点が多い。一方で、今後の技術開発により、さらなる発展が期待される分野といえる。

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] M.Motomura:"A Dynamically Reconfigurable Processor Architecture," Microprocessor Forum, Oct. 2002.
[5] P.Master: "The Age of Adaptive Computing Is Here," Proc. of FCCM, pp.1-3 (2002).
[6] 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
[7] H.Amano, A.Jouraku, K.Anjo, "A dynamically adaptive switching fabric on a multicontext reconfigurable device," Proc. of FPL2003, pp.160-170, Sept. (2003).
[8] X.-P. Ling, H. Amano, "WASMII: A Data Driven Computer on a Virtual Hardware" Proc. FCCM, pp. 33–42, (1993).
[9] S. Trimberger, D. Carberry, A. Johnson and J. Wong "A Time-Multiplexed FPGA" Proc. FCCM pp.22-28, (1997).
[10] T.Kitaoka, H.Amano, K.Anjo, "Reducing the configuration loading time of a coarse grain multicontext reconfigurable device," Proc. of FPL2003, pp.171–180, (2003).
[11] 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).
[12] http://www.pactcorp.com
[13] H.Schmit, et.al. "PipeRench: A Virtualized Programmable Datapath in 0.18 Minron Technology," Proc. of IEEE CICC, (2002).
[14] 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
[15] T.Okamoto, et.al. : "Design Tools and Trial Design for PCA-Chip2," IEICE Trans. Inf. & Syst. , Vol.E86-D, No.5, pp.868–871, (2003).
[16] T.Miyamori and K.Olukotum, "REMARC: Reconfigurable Multimedia Array Coprocessor," IEICE Trans. Inf. & Syst. , Vol.E82-D, No.2, pp.389–397, (1999).
[17] 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).
[18] 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).
[19] Caspi, E. et al.: "Stream Computations Organized for Reconfigurable Execution (SCORE)", Proc. of FPL2000, pp. 605–614 (2000).
[20] K.Wakabayashi, "Cyber: High Level Synthesis System from Software into ASIC," High-Level VLSI Synthesis, Kluwer Academic Publisher, 1991.