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

2006－AL－109 (3)
2006／11／21

[招待講演] ストリーム処理と動的再構成可能アーキテクチャPCA

小栗 清†

† 長崎大学工学部
〒852-8521 長崎県長崎市文教町 1-14
E-mail: †oguri@cis.nagasaki-u.ac.jp

あらまし さまざまな技術も最終的には物理的な実装によってその最終的な性能が決まり，有用度が定まる．データを一旦メモリ LSI 上に配置して，このデータを論理 LSI 上に設けた演算回路を使って加工する方式，たとえばコンピュータのソフトウェアによる処理方式ではデータをメモリ LSI と論理 LSI の間で何度も往復させる必要がある．一方，FPGA (Field Programmable Gate Array) 等の再構成可能論理 LSI を前提とすると，データを発生元から直接，演算回路群に投入し，処理のままに配置された演算回路群内を流すことにより処理を行うこと（以下ストリーム処理と呼ぶ）が可能となる．ストリーム処理ではそのパイプライン動作のために非常に高い性能が得られ，またメモリと論理ブロックという全体が大きく 2 分された場合の転送路に比べ，隣接演算器間の転送路は短いため消費電力を大きく削減できるという特徴がある．本稿ではこのストリーム処理とストリーム処理を目指して開発された LSI アーキテクチャである PCA (Plastic Cell Architecture) について述べる．

キーワード ストリーム処理，動的再構成可能アーキテクチャ，PCA

Dynamic Re-configurable Architecture PCA for Stream Processing

Kiyoshi OGURI†

† Faculty of Engineering, Nagasaki University
Bunkyo-Match 1-14, Nagasaki, 852-8521 Japan
E-mail: †oguri@cis.nagasaki-u.ac.jp

Abstract In conventional processing, the data is located in the memory at the first, then the processing unit, for example CPU, operates on it. A number of times, pieces of the data may go and return between the memory and CPU. This reciprocating processing leads to lots of power consumption and low performance. Instead of being located in the memory, the data could be flowed in the operation circuits. We call this style of processing a stream processing. This enabels low power consumption and high performance by its pipeline processing. However, in order to achive the versatility that CPU has, reconfigurable architecture is necessary for the stream processing. Dynamic reconfigurable architecture is preferable for being comparable with the CPU's versatility. If the reconfigurable computer is to evolve into such a general purpose computing platform that is superior to the von Neumann computer architecture, which consists of CPU and the memory, we believe that the reconfigurable hardware must offer the features as described below. The Plastic Cell Architecture (PCA), the general purpose dynamically reconfigurable computing architecture proposed by the author, fulfills the these requirements.

Key words Stream Processing, Dynamic Re-configurable Computing, Plastic Cell Architecture

1. 背 景

パターン認識，画像情報処理，音声情報処理，コンピュータビジョンなどの分野に於ける膨大な研究蓄積にもかかわらず，これらの技術をロボットや自動車などの自動制御に適用するためには，その精度・確実度において更なる研究が必要であると考えられている．

一方，これらの技術の前提を考えるとき，情報を一旦コンピュータ上に取り込んで認識処理等を行うということが暗黙のうちに仮定されており，例えばコンピュータビジョンの研究では「カメラから取り込んだ 2 次元画像情報から 3 次元情報を復元することをコンピュータビジョンといい，失われた 1 次元分の情報を推定しなければならないところにこの技術の困難性がある」との定義がまず第 1 に挙げられていることからもこの暗

黙の仮定の支配を感じることができる．

カメラからの信号をコンピュータに取り込もうと思えば2次元のピクセルに分解してこれをメモリ上に展開するということになり，ここからカメラからの情報は2次元情報であるということになったわけである．

ところが左右のカメラを特徴点の抽出と左右で同一の点を見ているかどうかの判断だけに使うことにし，その点の3次元としての位置はカメラの角度から求めることにして，次々と次の特徴点へカメラを動かして，物体の3次元位置を検出して行くことにすれば，これはもう2次元画像情報から3次元情報を復元する処理ではない．最初から3次元の処理である．このように一旦取り込むという呪縛から逃れることができれば，これまでにない視点で研究を進めることができる．

データを一旦メモリLSI上に配置して，このデータを論理LSI上に設けた演算回路を使って加工する方式，たとえばコンピュータのソフトウェアによる処理方式では，データをメモリLSIと論理LSIの間で何度も往復させる必要があり，全体を二分するメモリLSIと論理LSI間のこの転送はその大きな静電容量のため，転送速度を上げる場合には電力消費が著しい．一方，データを発生元から直接，演算回路群に投入し，処理のままに配置された演算回路群内を流すことにより処理を行う方式（ストリーム処理）では，そのパイプライン動作のために非常に高い性能が得られ，またメモリLSIと論理LSIという全体が大きく二分された場合の転送路に比べ，隣接演算器間の転送路は短く静電容量が小さいため消費電力を大きく削減できるという特徴がある．

しかしながらこのような利点を持つストリーム処理は，必要とされる処理に応じて演算回路とその間の転送路を配置する必要があるため，FPGA（Field Programmable Gate Array）等の再構成可能論理LSIを前提として始めて成り立つ方式である．

2. PCAの概念

プラスティックセルアーキテクチャ（PCA：Plastic Cell Architecture）は，FPGAよりもさらに汎用性を向上させ，布線論理の世界でCPUとメモリによるコンピュータに匹敵する汎用性を持たせることを目的として開発されたアーキテクチャである．PCAはルーティングネットワークのなかに記憶回路と論理回路をちりばめたアーキテクチャで，処理の本質である「データの記憶と加工・転送およびその制御」を直接かつ並列的に行い，動作中に回路を増減させることができる．NTT，長崎大学，京都大学，会津大学などで研究が進められており，具体的な実現例としては，NTTで開発されたPCA-1，PCA-2，京都大学で開発されたPCA-Chip2，長崎大学で開発中のビットシリアルPCAなどがある．

FPGA（Field Programmable Gate Array）の本質はその名前の通り製造後に布線論理，すなわちハードウェアをプログラムできるところにある．当初，プログラマブルとするためのオーバヘッドが大きいため，FPGAは集積量や性能の点でゲートアレイなどをリプレースするものではなく，積極的にプログラマビリティを使う可変構造システムがFPGA応用の本命であると考えられた．しかし，可変構造システムは面白いけれどもキラーアプリケーションがないと言われているうちに，FPGAはその少品種性から徹底した最適化が行われ，数十万ゲート規模以下のLSIや入出力端子がネックとなるLSIでは量産品であってもゲートアレイの代わりに使われるようになってきている．この事実は，可変構造がもつ汎用性が，可変であることそのものより重要であったことを示しているのではないだろうか．この汎用とは何であるかの分析により得られたアーキテクチャがPCAであるので，ここから議論を始める．

2.1 汎用であることの重要性

汎用であるものの代表であるコンピュータの発展の鍵は，その構成要素がハードウェアとソフトウェアから成り，ソフトウェアを取り替えることにより，任意の機能を実現できるところ，まさに汎用性にある．この結果，機能を階層的に，たとえばOS，ミドルウェア，アプリケーションプログラムというように整備することが可能になり，多くの努力が互いを効率化する形で集約できることとなった．機能を決めて装置を開発するほうがより良いものを作れるはずとの見方もあるが，実際はそうではなかった．たとえば20年ほど前にLISPという言語を専用に処理する装置（LISPマシン）が種々開発されたが，コストはもちろんのこととして性能でも汎用コンピュータにすぐに負けてしまった．汎用であるために多くの人の努力を集約できたということが重要であるわけである．

2.2 決定を後回しにする能力

それでは汎用であることの意味をもう少し考えてみよう．図1は機能表現の効率化が実は決定を後回しにする機能の進歩であったことを説明するものである．

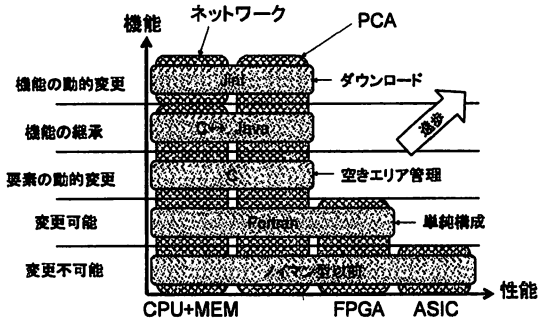

図1 決定を後回しにできる能力

横軸はハードウェアとしての実現手段の例を表し，縦軸は決定を後回しにできる機能の程度を表している．まず最初は機能が変更できない段階である．これはプログラム格納型計算機が現れる前の段階に相当する．次はプログラムを変更すれば機能を変更できる段階である．この段階を象徴するものとしてプログラミング言語のFORTRANがある．初期の頃のFORTRANでプログラムを書く場合には，たとえばどのような大きさの配列がいくつ必要であるかをあらかじめ見積っておく必要があった．実際に動作させるとその見積りが間違っていることがわかっ

て度々プログラムを変更しなければならなかった．そこで次の段階では，C言語を例にあげると，データを必要になった時点で割り付け（malloc），不要になったら回収する（free）という機能が考案され追加された（メモリのヒープ管理）．この結果，データについてはあらかじめ必要量を予測するということが不要となった．しかしながら機能の変更に対しては，C言語では広範囲の関数を見直す必要が生じるため，機能についてはあらかじめ十分な検討をして決定しておく必要があった．そこで次の段階のオブジェクト指向言語（C++やJava）では，機能の変更や追加を行う機能（クラスと継承性）が考案され追加された．すなわち機能の決定を遅らすことが可能になった．最近では，さらに次の段階，すなわち設計時に考慮されていなかった機能を動作時に追加できるようにさえなってきている．これらは機能や動作状況をあらかじめ正しく見積ることが如何に困難であるかを示している．技術の進歩がますます速くなる今日において，この決定を後回しにできる機能はより重要になってくるものと思われる．

2.3 布線論理の汎用性はどの程度か

以上説明した汎用性は，現在のところハードウェアとしてはCPU＋メモリという構造が担っている．しかしながら，CPU＋メモリを前提としたプログラム（program logic，プログラム論理）は機能を実現する一手段にすぎない．異なる手段として，論理回路（wired logic，布線論理によっても任意の機能を実現することができる．プログラム論理は機能を時間方向に展開したものであり，布線論理は機能を空間方向に展開したものである．十分な汎用性があれば布線論理の高速性，低消費電力性など，そのメリットは非常に大きい．布線論理にCPU＋メモリのような汎用性を持たせることはできないのだろうか．図1の横軸に示したASICでは，機能は製造時に決められてしまうため全く汎用性はない．FPGAは従来の回路は固定的なものであるというイメージを打ち壊す画期的なデバイスであるが，その汎用性はFORTRANレベルである．それでも十分なインパクトがあり，ゲートアレイをリプレースしようとしているわけである．しかしCPU＋メモリが実現している汎用性にはまだまだ遠く及ばない．

2.4 CPU＋メモリ＋FPGAではどうか

それではCPU＋メモリ＋FPGAという組合せで布線論理の高速性とプログラム論理の汎用性を両立させるというのはどうだろう．しかしFPGA上に構成された専用回路がメモリを使うのであればCPU並みの性能となってしまう．アドレスによる読出し／書込みで特徴づけられるメモリは非常に単純な構成・機能ながら柔軟性に富み，これこそがノイマン型アーキテクチャの本質である．高性能CPUはそのメモリを最大限有効に使えるように作られた専用マシンであるので，メモリを使っている限り，問題毎に専用回路を使用してもCPUを大幅に越えることはできない．それでは性能や柔軟性においてCPU＋メモリを越えるためには，どのような構造を考えればよいのであろうか．CPU＋メモリによる柔軟性がどのように実現されているかを布線論理の場合と比べてみよう．

2.5 相違点はヒープ管理にある

アプリケーションをプログラム論理に展開する場合と布線論理に展開する場合の共通点と相違点を図2にまとめた．

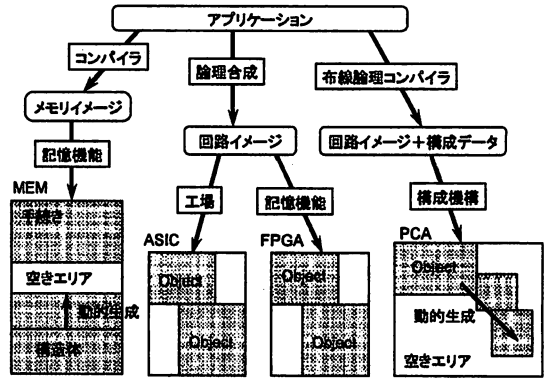

図2 動的な領域の利用

まず，コンパイラや論理合成系によって生成されたメモリイメージ，回路イメージを実際のメモリやASIC，FPGAに構成するところが，各々，記憶機能，工場，記憶機能と異なるのは当然であるが，ここで重要な相違点は，プログラム論理ではメモリ上のデータ表現やオブジェクトインスタンスを手続きの実行に伴って動的に構成できる点である．ASICやFPGAでは動作中に構造が変化することはない．複雑な情報処理はデータ表現やオブジェクトインスタンスを生成，操作することによって行われるため，この相違点は本質的である．柔軟性の源はこのヒープ管理と呼ばれるメモリの使い方にある．布線論理に十分な汎用性を持たせるためには布線論理の世界でこのヒープ管理を行える（図2の右端）仕組みを考えなければならない．FPGAが第2段階レベルの汎用性に留まっているのは，ヒープ管理をできるように作られていないからである．

2.6 汎用性を生み出す本質

先にコンピュータがハードウェアとソフトウェアという2つの部分から成るということが重要であると述べたが，そのハードウェアがCPUとメモリという2つの部分から成るということにも重要な意味がある．実際に動作しているのはハードウェアであるから，任意の機能や動的構造を実現するためには，このハードウェアの一部を可変構造としておく必要があり（**可変部**, PP : Plastic Part），またこの可変部を構成する機能と，構成することによってさまざまな機能を得るための基本機能とが，あらかじめ備わっている必要がある（**組込部**, BP : Built-in Part）．コンピュータの場合は可変部を「メモリの記憶」が担い，これに意味を与える組込部が「あらかじめ機能が組み込まれたCPUの命令セットやメモリのアドレスによるアクセス機構」であるわけである．まとめると「汎用システムはハードウェアとソフトウェアから成り，ハードウェアは可変部と組込部から成る」ということになる．これを**汎用システムの二重の二重構造**と呼ぼう．ここでソフトウェアとはプログラムというよりもハードウェアをどう構成するかという情報そのものを指す．

生物の成り立ちにもこの構造をみることができる．DNAの情報に従ってアミノ酸からたんぱく質を合成する機構を組込部とすると，たんぱく質が可変部であり，DNAの情報がソフトウェアとなる．この構造により多様な種が存在して環境に適合している．また人の社会において，人の集合をハードウェアと見ると，記憶力や思考能力，コミュニケーション能力が組込部，共有された記憶が可変部，そして文化がソフトウェアとなる．ソフトウェアの生成はそのコピーに比べて大変な時間がかかるという点も共通している．

2.7 処理とは

さてコンピュータによる処理とは何であろうか．これはCPUとメモリの動作を考えてみればわかるように，メモリ上のデータを適切に加工し移動することである．これを機能という観点から整理してみると，加工と移動は操作であるのでそのまま加工機能と移動機能を対応させればよい．データは機能ではないが，その保持に記憶機能が必要であるので，結局，処理は加工機能，移動機能，記憶機能と，その順序を決める制御機能から構成されることになる．ここで，機能の指定を命令で表し命令の実行順序を記憶機能のアドレス順に対応させた制御をプログラム論理というのであった．

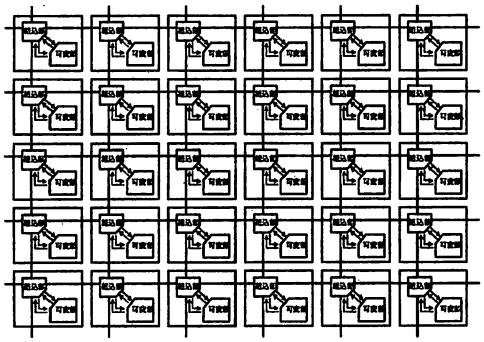

図3 PCAの基本構造

CPU＋メモリにおいて，処理を構成する加工機能，移動機能，記憶機能，制御機能が，汎用システムの二重の二重構造のどの部分に置かれているかを見てみると，加工機能はCPU内の演算回路が実現しているので組込部，移動機能はCPUとメモリのアドレスによるアクセス機構が実現しているので組込部，記憶機能はメモリの記憶が実現しているので可変部，制御機能のうちの命令取出しと解釈はCPU内の命令取出し解釈部とアドレスアクセス機構が実現しているので組込部，制御機能のうち命令の格納はメモリの記憶が実現しているので可変部ということになる．

2.8 布線論理を汎用とするための必要条件

汎用な布線論理を考える場合には，加工や制御を担うべき布線論理が汎用，すなわちさまざまな回路を用意できるようにするというわけであるから，加工機能と制御機能は可変部に，その構成を行う最小限の機能は組込部になければならない．記憶機能は記憶の本質から可変部になければならない．一方，移動機能は固定的な機能であるため組込部でかまわないが，加工や制御を行う回路が可変部にあってその数が増えることを考えると，複数の移動が同時に行えるネットワーク型でなければならない．さらに可変部に作られた回路が他の可変部を構成するはずであるので，構成機能はネットワークと強く連携した形で分散して存在していなければならない．これが布線論理を汎用とするための必要条件である．

2.9 PCA

論理素子が高速となり，通過するルータの段数よりも転送距離そのものが性能を決めるようになるとルータをメッシュ状に結合した単純なネットワーク構成が主流になると考えられる．すなわち移動機能を司るネットワークはルータのアレイという形態となる．そこでこのルータにもともと全体に配置しなければならない可変部をアレイ状に分割したものを加え，さらにその構成機能を加えて単位胞（セル）とすれば，これをアレイ状に配置しメッシュ結合することで布線論理を汎用とするための必要条件を満足することができる（図3）．

すなわち，セルの可変部をどう作るかやセルの可変部同士をどう接続するかに自由度が残るが，セルには組込部としてのルーティング機能と分散させるべき構成機能があり，そして可変部としての構成可能な加工機能，制御機能，記憶機能があって，このセルをアレイ状に配置することで移動機能が構成機能と強く連携された形でメッシュネットワークを構成し，そのなかに加工機能，制御機能，記憶機能をもつ可変部がちりばめられた形になるからである．この可変性があり，かつセル構造のアーキテクチャをプラスティックセルアーキテクチャ（PCA）と名づけた．ここでプラスティック（Plastic）という言葉は**可塑性のある**という意味で用いている．上記でセルと述べた構成単位を以下ではPCAセルと呼ぶ．PCAでは図4のように，まず初期ロードなどによりPCA上に存在している回路が，必要となった回路の構成情報をネットワークによって送りつけて別の回路を生成し，協調しながら処理を進めていくことができる．

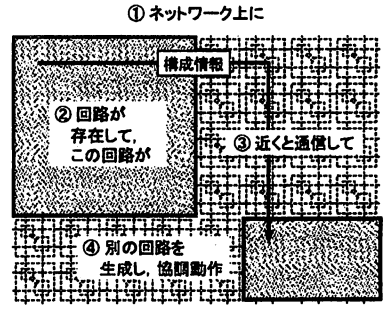

図4 PCAの動作イメージ

2.10 LSI高集積化技術からの要請

コンピュータやネットワーク技術の発展は微細加工技術の進歩に伴うVLSIの高集積化が原動力となっている．さらなるVLSIの高集積化を活かすには，まず第1に消費電力の問題に対処しなければならない．最近は漏れ電流の影響も無視できな

くなってきているが，CMOS 回路を前提とすると，消費電力は単位時間当たりの信号の遷移数と負荷容量に比例する．回路の接続関係が同じならば，負荷容量は面積に比例するので，高集積化によって面積と動作周波数当たりの消費電力は変化しない．しかし高速動作可能なトランジスタを当然高速動作させるので，面積当たりの消費電力は増大してしまう．そこで反省すべきは CPU＋メモリの構造である．この構造こそが電力を消費する原因である．なぜならば全体を CPU とメモリの二つで構成しその間で信号をやりとりするということはもっとも遠い距離（全体をたくさんの部分に分けて通信する場合と比較して）の通信が発生しており（負荷容量が大きい），かつ一回に一つの情報しか送れないために勢い通信のピッチを上げる必要が生じているためである（クロック周波数が高い）．したがってこれは VLSI の高集積化を活かしできるだけ CPU やメモリという構造を使わずに布線論理で処理を行い，必要な速度で各部分を動作させればよいということになる．もちろんたくさんの CPU＋メモリを置くという方法も存在する．

第 2 に高集積化によって製造可能となった大規模回路を効率良く設計しなくてはならない．設計規模の増大を単なる階層設計や設計の再利用でまかなうことはできない．たとえばこれまでのプリント基板上の回路は設計が階層的であるだけでなく製造もまた段階的であるため，一度に製造しなければならない VLSI には適用できないのである．動作するように製造された部品を買ってきて使うのと，その設計情報を買ってきて自分の責任で製造して動作させることには大きな違いがある．

第 3 に配線遅延の割合の増大に対処しなければならない．高集積化によって配線の単位長当たりの遅延が増大するが，回路の接続関係が同じならば，配線の長さも減少し結果として配線遅延は変化しない．しかしトランジスタはより高速に動作できるので配線遅延の割合は大きくなってしまう．回路の遅延のうちトランジスタの動作遅延による部分は論理設計に依存するのに対し，配線遅延による部分は論理設計とレイアウト設計に依存する．したがって，配線遅延の割合が小さい場合はまず論理設計を最適化してから次にそのレイアウト設計を最適化するということが可能であった．しかしながら配線遅延の割合が高い場合には論理設計とレイアウト設計を同時に最適化する必要が発生する．しかしこれは簡単なことではない．これらに対する解は 2 つある．第 1 は命令セットが設計をハードウェアとソフトウェアに分けたように，布線論理の均質な構成要素となり得るゲートよりも大きい何かを定義し，設計と製造をその上下に分けることである．その何かは最終的に実現されるものより大幅に簡略化されることから十分な最適化ができるということになる．第 2 はタイミングを抽象化して，すなわち非同期として，実時間に依存しない順序のみをインタフェースとする階層化や部品化を行うことである．これにより設計製造の困難さをソフトウェアによる階層化や部品化に近づけることができる．

PCA は布線論理を目指し，PCA は設計を分ける単位であることを目指し，そして PCA は非同期を目指す．

2.11 通信からの要請

次にネットワークからの要請を述べる．端末，伝送路，スイッチ，制御装置からなるネットワークには，電話網，インターネット，並列コンピュータなどがある．電話網の制御装置は交換機と呼ばれる特殊なコンピュータであり，インターネットの端末やスイッチ，制御装置の一部はコンピュータであり，並列コンピュータの端末はコンピュータである．すなわちコンピュータはネットワークに不可欠な構成要素となっている．そのコンピュータの CPU とメモリによる処理は，先に述べたように加工機能，移動機能，記憶機能，制御機能から成り，移動機能はネットワークによる情報の移動，すなわち通信と本質的には同じものである．

にもかかわらずコンピュータの CPU とメモリにおける通信はフォンノイマンボトルネックと言われるように，同時には 1 つのアドレスに対する読み書きしか行えず，これは並列かつ分散的なネットワークの通信と大きく異なっている．この結果，ネットワークのなかにこの範囲はコンピュータであるという境界をはっきりと作ってしまうことになっているわけである．もっと均質にならないのかという要請である．

PCA はネットワークとシームレスであることを目指す．

3. PCA の基本構造

PCA の基本構造は布線論理を汎用とするための必要条件と今後主流になる思われるメッシュ構造のネットワークを組み合わせることにより得られたものであり，その構造は先に述べた通りである（図 3）．ノイマン型アーキテクチャにさまざまな具体例が存在し，改良が進められてきたように，PCA にもさまざまな具体例が存在し得る．

3.1 ビットシリアル PCA

一般に構成単位を組み合わせてさまざまな機能を実現しようとするとき，構成単位のサイズ，これを粒度という，が小さければ多くの構成単位を組み合わせなければならず効率が悪い，これをオーバヘッドが大きいという．また構成単位のサイズが大きければその結果として構成単位の機能も大きくなり，実現しようとする機能に無関係な機能は使われないことになる，これを無駄が大きいという．オーバヘッドと無駄の削減は背反する関係にあるが，処理を特化することにより必要な機能の種類を減らせば，オーバヘッドを削減するために構成単位のサイズを大きくしても，それほど無駄を発生させないようにすることができるはずである．ビットシリアル PCA は処理をビットシリアル処理に特化することによりオーバヘッドと無駄を共に削減することをねらうものである．また処理を通信のビットシリアルな特性に合わせることにより通信と処理を融合させることをねらうものである．

3.1.1 設計のペトリネットによる表現

ハードウェアによる性能向上は関連するデータ群，これを処理対象ということにすると，1 つの処理対象に対して複数の処理を同時に行う並列処理，複数の処理対象に対して同じ処理を平行して行う並列処理，部分回路が縦列接続された処理部に複数の処理対象を次々と投入して処理を流れ作業的に行うパイプライン処理，そしてこれらを補助する先行制御と置いてきぼり制御により達成される．

これらの並列度の高い処理方式では部分回路が同時に動作していること，部分回路間の接続関係が処理の流れ（データフロー）と対応していることが特徴である．たとえば並列処理では接続関係に複数の経路が存在して合流し，パイプライン処理では複数の部分回路が縦列接続されているというようにである．

このような場合，部分回路の機能が単純であれば部分回路の接続関係が機能の適用順序を表すことになるため，部分回路の接続関係を表現する回路図は処理の内容そのものを表していることになり理解しやすい．

ところがハードウェアリソースが十分でないあるいはデータを十分に供給できない場合には，データフローに沿って部分回路を配置することができないあるいは配置する意味がないので，部分回路の機能を少しずつ変えて繰り返し使用するなどの工夫が行われる．

このような場合，回路図は処理の流れとは無関係な接続関係を多く含むこととなり理解しにくくなる．特に処理対象をメモリに置く場合には処理をしてはメモリに戻すという構成となるため，回路図と処理の流れとの対応はいっそう希薄となる．

このため回路図の意味モデルとして，処理の流れとは無関係な刺激反応モデルが広く使われることとなった．VHDL や Verilog，そして最近の SystemC などもこのモデルである．

しかしながらハードウェアによってできるだけの性能向上を目指す場合には，部分回路を同時動作させるように設計するべきであり，それを強制するものとして部分回路の接続関係が直接処理の流れに対応する表現形式を考えることができる．

ペトリネットは，待合わせ合流や選択分岐など，処理要求の流れを示すための表現形式であるので，これに処理の内容を記述できる形式を加えるとまさに接続関係が処理の流れに対応する表現形式となる．なおこの表現形式の追加は単なるシンタックスシュガーであり，ペトリネットのモデルを拡張するものではない．

ビットシリアル処理の場合には機能当たりの回路が小さいので，少ないハードウェアリソースを有効利用するために同じ部分回路を少しずつ機能を変えて繰り返し使用するなどの工夫を行う必要性が少なく，この処理の流れをそのまま表現するペトリネットの使用が向いていると思われる．

3.1.2 領域管理

プラスティックセルアーキテクチャは回路が必要に応じて新しい回路を構成しこれと協調して動作するということを可能にするためのアーキテクチャである．PCA-1 による動作実験により回路が自分の横に回路を作り出してこれと協調して動作するという設計が意図どおりに動作することを確認することができた．しかしこの設計では回路の横に空き領域があることが前提となっており，どのように領域を管理するかについては考慮されていない．領域を管理する回路を設けるという方法はすぐに思いつくことができるが，この方法では同時動作している沢山の回路からの領域割当て要求をどう処理するのかという問題が発生する．管理するためには情報の一元管理が必要でこれを行う回路の処理能力をいくらでも高められるはずはないからである．また必要な処理能力に応じて管理を分割するという改良を加えたとしても割当て要求をどこに送ればよいのかという問題が発生する．

そこで管理することを止めて，回路がそこにあるかないか，そのことがすべてであると考えることにより，細胞分裂や生物の増殖を真似た，圧力による空き領域確保というメカニズムを思いつくことができる．これは新しい回路の生成は隣接した空き領域に限ることとし，隣接部分のすべてに回路が存在する場合には，その隣接回路に圧力をかけて移動させ，空きを作るというアイディアである．この方法では複数の領域確保が同時に発生しても圧力をベクトルとして加算するだけでよく，また隣接した領域に新しい回路を作るため，協調動作を行うための接続が最短距離で行えることとなる．

領域管理は動的再構成可能アーキテクチャにとって大きな問題であったが，この圧力による領域管理は一つの解となっているものと思われる．また回路の移動に伴って回路間の接続も移動しなければならないので．圧力による領域管理は，回路間の接続関係が複雑でないビットシリアル方式との相性が良いものと考えられる．

3.1.3 動的構成機能の表現形式

CPU＋メモリを前提としたプログラミング言語では構造体やクラスの新しいインスタンスを確保するために malloc や new が使われる．そこでハードウェアの場合も回路の新しいインスタンスを作り出す表現形式として同様の形式を用意すればよいように思えるが困難が発生する．それは，構造体やクラスや回路の新しいインスタンスを作り出して使うということを表現するためには，

- その契機を示すこと
- 新しいインスタンスの内容を示すこと
- 既存部分と新しいインスタンスとの接続を示すこと

が必要であるのに対し，malloc や new では，新しく確保する契機やインスタンスの内容は表現されているものの，そのインスタンスがどう接続されるかはインスタンス内に用意されたポインタに値を代入することによって実現されているからである．ハードウェアの場合にはポインタではなく接続のための回路が新たに用意されなければならない．

そこで新しく作り出す回路内に新しい回路との接続を含んだ形式，すなわち再帰構造が必要となる．わかってみれば簡単なことであるが，動的な回路を表現する形式とは，ある回路種 A を定義する回路図の中に回路種 A を含んだ回路図ということになる．これだけでは内部に含まれた回路種 A を使う契機が示せないのでそのための形式も必要になる．この契機信号は内部の回路種 A を迂回または接続するスイッチの切替え信号にもなっている．ビットシリアル PCA では，この再帰構造は単なる回路図ではなくペトリネットをベースとする接続表現のなかに組み入れられている．

4. まとめ

PCA の技術背景と，概念，PCA アーキテクチャの具体的な実現例として長崎大学で開発中のビットシリアル PCA について述べた．