システムＬＳＩ設計技術　93－13
（1999. 11. 27）

Communicating Logicによる汎用情報処理機構の実現

塩澤恒道、Norbert Imlig、永見康一、小栗清

NTT未来ねっと研究所

〒239-0847　神奈川県横須賀市光の丘1－1

shiozawa@exa.onlab.ntt.co.jp

あらまし　本稿では、自律再構成可能なハードウェア・アーキテクチャであるプラスティック・セル・アーキテクチャ(PCA)上に汎用情報処理機構を実現する方法について述べる。具体的には、スループット指向の汎用情報処理機構を実現するためのオブジェクト構成および幾つかの基本となるオブジェクトの構成と実装サイズについて検討した結果について述べている。さらに、複数のオブジェクトが共有するデータを格納するパイプラインメモリの構成について述べ、シミュレーション評価を行った。

キーワード　自律再構成可能LSI、汎用情報処理機構、PLD、FPGA、非ノイマン型計算機、論理回路

A Implementation of General-purpose Computing Mechanism on the Reconfigurable LSI Architecture

Tsunemichi SHIOZAWA, Norbert IMLIG, Kouichi NAGAMI, Kiyoshi OGURI

NTT Network Innovation Laboratories

1-1 Hikari-no-oka, Yokosuka-shi, Kanagawa 239-0847 Japan

shiozawa@exa.onlab.ntt.co.jp

Abstract

In this paper, we report on the implementation of the general-purpose computing mechanism by using reconfigurable computer architecture called Plastic Cell Architecture (PCA). The method and design policy of composing objects to achieve the throughput oriented computing are described. The circuit of some basic objects ware implemented and the size of the object were evaluated. In order to show that PCA make it possible to construct the real scalable general-purpose computing mechanism, the composition of the shared memory is shown, and evaluate throughput by the simulation.

key words　reconfigurable computing, general-purpose computing mechanism, PLD, FPGA,Non von Neumann computer, wired-logic

1.はじめに

LSIの集積度の向上によりシステム全体(または大部分)をシリコンチップ上で実現するシステムオンシリコンが可能となりつつある。従来、LSI の外部入出力端子を経由して接続されていたプロセッサ、メモリおよび入出力等の機能ブロックが1つのシリコンチップ上に搭載可能となることで、チップ内部と外部との時間のギャップや入出力端子数ネック等の物理的な制約が緩和される。しかし、従来のアーキテクチャをそのままシリコンチップ上に実現しても解決しない問題も多い。たとえば、シリコンチップ上に多数のプロセッサを実装しても、複数のプロセッサを効率よく動作させ、台数に比例した性能向上を得ること(スケーラビリティ)はプロセッサ台数が増加するに従い困難となる。

一方、集積度の向上により、設計方法に関しても従来の延長ではない新たな手法が要求されてきている。この例としては設計対象を LSI を単位として考えるのではなく、既存の設計結果である機能ブロック(IP:Intellectual Property)に新しい設計を追加してシステムを設計する手法が試みられている。このような手法はソフトウェアでは当然のこととして用いられてきたが、LSI 設計では設計資産の再利用はソフトウェア設計に比べて遅れている。LSI 設計で設計資産の再利用が困難なことの要因としては記述方法やハードウェア本来のもつ同時並列性等さまざまな要因があるが、過去の設計資産に LSI の物理的な特性(レイアウト情報や素子の遅延)が含まれることにある。このような物理的な特性を含まない機能レベルでの設計資産を再利用しようとすると機能部分の設計は不要となるが、論理合成およびレイアウトが再度必要となり再利用によるメリットが減少することになる。

集積度の向上と設計資産の再利用の観点から、シリコンチップ上にスケーラブルで均質なリソースの階層を新たに設け、ソフトウェアと同様に物理階層と論理階層とを分離することで設計資産の利用を図る提案[3]がなされている。我々は、このような階層の候補の一つとして再構成可能ハードウェアを位置づけ、プラスティック・セル・アーキテクチャ[1-3]を提案している。

プラスティック・セル・アーキテクチャは当初からその均質なリソース階層の上に汎用情報処理機構の実現を目指したアーキテクチャであり、ソフトウェアの機能の一部を切り出して FPGA (Field Programmable Gate Array) 上で実行させることで性能をアクセラレートするものとは異なる特徴を持っている。

本稿は、自律再構成可能なハードウェアであるプラスティック・セル・アーキテクチャ上に Communicating Logic と呼ぶ手法によって汎用情報処理機構を実現する方法について述べるものである。

以下2. では従来のメモリ内蔵型計算機で実現されている汎用情報処理機構について述べ、3. では自律再構成可能アーキテクチャであるプラスティック・セル・アーキテクチャおよび Communicating Logic について述べる。4. ではプラスティック・セル・アーキテクチャでスループット指向の処理を実行するための基本的な構成要素について述べる。

2. 汎用情報処理機構

ここでは、メモリ内蔵型計算機の問題点とスループット指向のアプリケーションが再構成可能ハードウェアに適している理由について述べる。

2. 1 メモリとプロセッサ

メモリ内蔵型計算機は書き換え可能なメモリとこのメモリに逐次アクセスして演算等の処理を実行するプロセッサによって構成される。従来から指摘されているようにメモリ内蔵型計算機の性能はメモリアクセスのバンド幅によって制限される。この傾向は近年マイクロプロセッサの動作クロックが向上するに従って益々顕著になってきている。メモリアクセスのバンド幅を増加させる有効な手段として、メモリアクセスの局所性を利用したキャッシュメモリが多くのプロセッサで採用されており、最近のマイクロプロセッサでは50%～90%以上のトランジスタがキャッシュメモリのために使用[4]されている。プロセッサ内で行われる投機的実行 (Speculative Execution) や VLIW (Very Long Instruction Word) アーキテクチャはさらにメモリのバンド幅に対する要求を拡大すると考えられる。

書き換え可能なメモリにプログラムとデータを格納することによって得られるプログラマビリティは多種多様な問題をメモリ内蔵型計算機を用いて解くことができることを示し、広く社会の基幹システムとして用いられている。

2. 2 スループットとレイテンシ

プロセッサの高性能化の要求はさまざまな分野から求められており、プロセッサ単体での性能向上はほぼ従来の割合で続いている。プログラム内蔵型計算機に対する性能向上要求は通信ノードシステムに関しては以下の理由によるものと考えられる。

- 少ない通信ノードで多数のタスクを処理可能とすることでシステムの設備コストおよび運用コストを低減できる。
- 他のタスクに比べて優先度の高いタスクの応答時間を短縮することができる。

一般に通信ノードにおいてタスクの処理時間であるレイテンシ(latency)を短縮することは必ずしも最重要な要求条件ではない。これは、通信ノードで実行されるタスクのトリガとなる通信網の通過時間が数十ミリ秒であるのに対して、処理待ち時間を除いたプロセッサでの実行時間は数ミリ秒以下であるからである。このように多数のタスクを処理するようなシステムをメモリ内

蔵型計算機で構成する場合には以下のような問題が発生する。

(a)タスクスイッチによりプロセッサで実行中のタスクが切り替わり、メモリアクセスに関するタスク毎の局所性の性質が利用できなくなる。

(b)タスクのスケジューリングやプリエンプションに関して複雑な設計と詳細な評価が必要となる。

これらのことは、本来スループット(throughput)が要求されるシステムに対し、レイテンシを短縮することで要求を満足させようとする難しさが問題の解決を困難にしていると言える。

2．3 並列処理と粒度

スループットを指向した汎用計算機構として、メモリの共有／結合方法、プロセッサ間の通信方法等に着目したマルチプロセッサの研究[5]が多数なされている。これらの研究では固定した機能を持つプロセッサを相互に接続する接続網のアーキテクチャを変えることによって如何にスケーラビリティを実現するかということに重点を置いた評価が行われている。

上述したようにメモリ内蔵型計算機ではタスクスイッチによってメモリアクセスに関する局所性の性質が利用できなくなり、このことがスループットを向上させるための大きな問題となっている。タスクスイッチの原因となるマルチタスクはプロセッサが入出力待ち等で空きとなる時間を有効に活用するために導入された方法であるが、別の見方をすれば、タスクの粒度がプロセッサの粒度に比べて小さすぎる(タスクの粒度に比べてプロセッサの粒度が大きすぎる)ことに原因があるとも言える。理想的には並列に処理するタスクの数だけプロセッサを用意しプロセッサの機能および性能は固定ではなくタスクを実行する上で最適化されているかまたは各タスクに対して要求されるレイテンシを充分満足可能な性能であればよいはずである。

しかし、現実にはシステム設計の段階で並列に処理するタスクの数を把握することは困難であり、さらに、それらの数のプロセッサを準備することは逆にコストを増加させてしまう。

2．4 再構成可能ハードウェアの利用

スループット指向のアプリケーションを再構成可能ハードウェア上で実行することのメリットを以下に示す。

(a) タスクが個別に使用するデータを実行しているタスクの近傍に配置することでデータアクセスに関する無用な競合の回避と局所性の性質が利用できる。

(b) 予め並列に処理するタスクの数が与えられなくとも実装されているハードウェアリソースの範囲で数を増減させリソースの再利用が可能。

(c) プロセッサに相当する処理の実行主体をタスクの粒度に合わせて生成することが可能。

我々は、このような再構成可能なハードウェアの特徴を生かし、従来レイテンシを短縮することで高速化を実現していた組み合わせ最適化問題をスループットの向上によって高速化する手法を提案[6]してきた。また、再構成可能なハードウェアを用いたルーチングスイッチの構成方法と再構成可能ハードウェア上で自律負荷分散を行うメカニズムの実現性について検討[7]してきた。

3．プラスティック・セル・アーキテクチャ

プラスティック・セル・アーキテクチャの概要とその上に機能を実現するための設計手法であるCommunicating Logicについて述べる。

3．1 PCAの構成

プラスティック・セル・アーキテクチャ[1-3] (Plastic Cell Architecture: 以下PCAと略記)の構成を図3.1に示す。

PCAは相互に接続された均一なセル(cell)で構成され、各セルは組み込み部(bp部)とプラスティック部(pp部)とから成る。組み込み部は予め定義された固定機能からなり、隣接セル間の組み込み部と相互に接続されている。プラスティック部はセル単位での書き換えが可能なプログラマブル論理であり、隣接したセルのプラスティック部と相互に接続され、論理ゲートや記憶素子として使用可能な論理回路である。

PCAは以下の特徴を持つ。

(a) 組み込み部はセルが固定的に持つ機能を提供し、プラスティック部の機能を書き換え可能。

(b) プラスティック部は組み込み部を介して直接接続されていないセルのプラスティック部および組み込み部と通信することが可能。

(c) 各セルは空間的に一様な構造を持ち、単純な空間の拡張によりスケーラブル。

PCAが従来の多くの再構成可能ハードウェアと異なるのは、プラスティック部の書き換えがPCA自身によって自律的かつ並列に行われる点にある。従来の再構成可能ハードウェアは論理回路の全部または一部の書き換えはプロセッサ等を介して外部から行っていたため、回路が変化するといった動的なふるまいの実現にはプロセッサが介在する必要があった。PCAではPCAの内部に生成された回路が組み込み部の機能を用いて自律的に回路の書き換えを行うため、一般のソフトウェアで実現可能なプログラマビリティ(データ構造の変更／追加、命令の書き換え)をPCA内に閉じて実現することができる。これにより、プロセッサの介在による書き換えのオーバヘッドが削減され、PCAの内部で書き換えが並列に実現できる。

以下現在試作中のチップをベースに説明する。プラスティック部は4つの4入力1出力のLUTで構成されるベーシックセルの集合であり、LUTに構成情報書き込むことによって配線プリミティブとしての機能、状態を保持する記憶(FF)機能、入力端子からの任意の

論理演算を出力する論理ゲートプリミティブとしての機能が実現される。さらに、プラスティック部はデータを記憶するメモリとして使用することもできる。

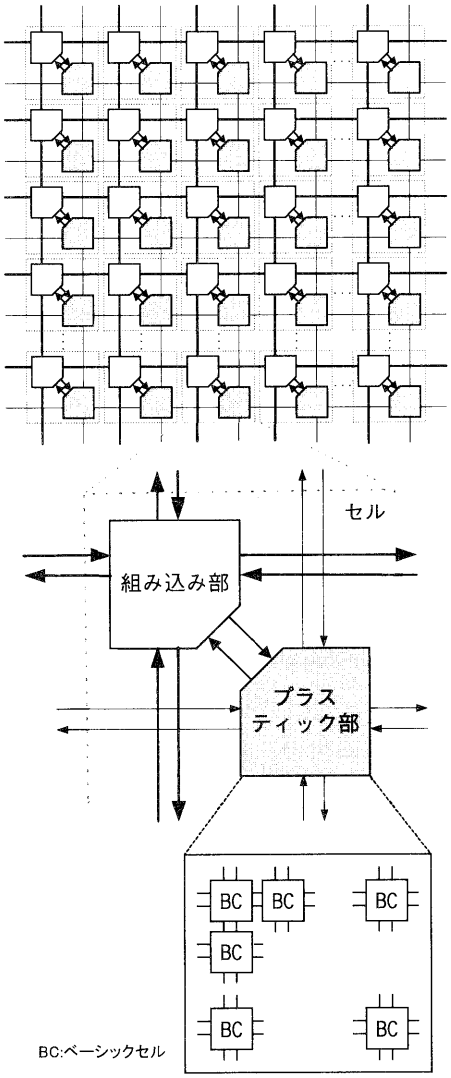


図3.1 PCA の構成

組み込み部は表3.1に示す 12 個の命令に対して予め定義された処理を実行し、これらの命令を組み合わせることによってプラスティック部への情報の書き込み、プラスティック部からの情報の読み出し、メッセージ通信が可能となる。PCA チップのチップ間では組み込み部のインタフェースのみが接続される。

現在設計中の PCA チップは以下のような物理仕様となっている。詳細については文献[8,9]を参照のこと。

(a) 組み込み部の回路およびプラスティック部に生成される回路は全て非同期回路である。

(b) プラスティック部は隣接したベーシックセルが相互に接続された 8×8 個のベーシックセルで構成されている。

(c) 組み込み部は4bitのデータ幅で隣接したセルの組み込み部および自セルのプラスティック部と双方向に接続されており、送信元と送信先が未使用であれば設定されている経路とは独立に経路を設定可能である。

表3.1 組み込み部(bp部)の命令

命令	機能
Clear	メッセージの最後
Pp_open	pp部との接続オープン
Pp_close	pp部との接続クローズ
Config_in_function	構成データ入力(機能)
Config_in_memory	構成データ入力(メモリ)
Config_out	構成データ出力
Config_out_with_in	CI付き構成データ出力
West	以後のデータを西へ
North	以後のデータを北へ
East	以後のデータを東へ
South	以後のデータを南へ
pp_out	以後のデータをpp部へ

3. 2 オブジェクトとメッセージ

プラスティック部の書き換えによって生成／削除される単位をオブジェクト(Object)と呼ぶ。図3.2に示すように、オブジェクト間の通信は、組み込み部に対して送信元オブジェクトから経路情報に引き続き送信するデータや制御情報が格納されたメッセージを出力することで行われる。経路情報は各セルでの進行方向を示す情報(*west, north, east, south, pp_out*)の列で構成され、途中のセルを通過する度に先頭部分が消費(削除)され、消費される情報の内容に従って隣接した4つのセルから次に進むセルまたは可変部への経路が設定される。組み込み部に設定された経路上に Clear が

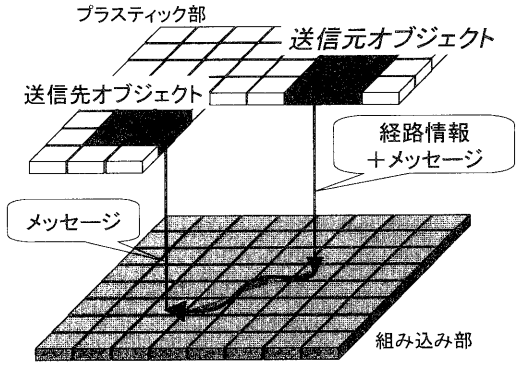


図3.2 オブジェクト間でのメッセージ送受信

転送されることでその経路は解除される。メッセージの中に *config_in_function*、*config_in_memory*、*config_out*、*pp_open* および *pp_close* 命令を適宜含めることによってオブジェクト構成情報の書き込み、読み出し、オブジェクトの生成および消滅が行われる[8]。

このように PCA では組み込み部に予め用意された機能を使ってメッセージの送受信およびオブジェクトの生成／消滅を自律的に行い、生成されたオブジェクトは独立に動作可能なハードウェア機能ブロックである。

3．3 Communicating Logic

PCA の各セルが組み込み部とプラスティック部と呼ばれる性質の異なる2つの部分から構成されていることに着目し、プラスティック部に生成されるオブジェクト間の通信機能を利用した設計手法として Communicating Logic を提案する。Communicating Logic は以下の特徴を持つ。

(a) オブジェクト内で行うデータ処理とオブジェクトエクト間の制御を分離[10]する。

(b) 処理に固有の変数をメッセージに埋め込み、オブジェクト間の通信機能を配線としてでなく一時記憶用のバッファとして利用する。

上記(a)によってオブジェクトの生成（コンパイル）と PCA 上へのオブジェクトの配置（リンク）とを分離することができる。さらに、オブジェクト間のリンクは送信元オブジェクトが組み込み部に出力する経路情報を変更することによって実現可能となるので、オブジェクト間の制御（順序や依存関係）を柔軟に変更することが可能となる。上記(b)は変数の局所性に対する最適化、高スループット化およびオブジェクトの再利用を実現するものである。これは組み込み部が単なる通信路として利用されるのではなく、プラスティック部に生成されるオブジェクトの機能の一部として利用されることを意味する。

4．オブジェクト構成

PCA 上に汎用情報処理機構を実現する上で基本要素となる制御、メッセージ通信およびデータ格納を行うオブジェクトについて述べる。

4．1 機能オブジェクトの構成

オブジェクトは基本的にメッセージの受信をトリガとして動作し、メッセージに含まれる値およびオブジェクト内に格納されているデータに基づいて動作する。一般にオブジェクトは有限状態機械として構成されるので、メッセージとその時点のオブジェクトの状態との組み合わせによって動作が決定されるが、オブジェクトの動作を決定する状態情報もメッセージに含めることによってオブジェクトに到着するメッセージの時刻（タイミング）とオブジェクトの動作とが独立になるように設計し、メッセージ単位でのオブジェクトの再利用を図る。

この例としては繰り返し処理を制御する繰り返し制御オブジェクトと繰り返し変数との関係が挙げられる。図4.1(a)のように繰り返し変数 *i* をオブジェクト内に状態として持つ場合、その繰り返し制御オブジェクトは繰り返し処理が完了するまで他の繰り返し処理を制御することができない。しかし、図4.1(b)のように繰り返し変数をメッセージに含めることによって繰り返し制御オブジェクトは受信したメッセージ毎に繰り返し変数のインクリメント（またはデクリメント）と終了条件を判定することとなり、複数の繰り返し処理を独立に処理することが可能となる。

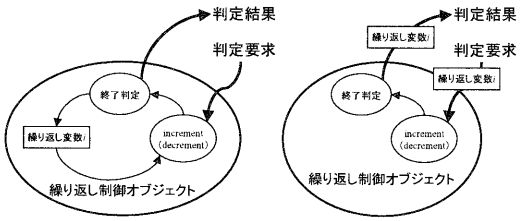


図4.1 繰り返し処理の例

4．2 ローカルメモリオブジェクトの構成

PCA の特徴として、セル単位でプラスティック部をメモリまたは機能の何れかとして使用可能な点がある。これを利用することで、PCA 上に生成されるオブジェクトが利用するデータをそのオブジェクトが配置されたセルの近傍に配置することができる。

PCA ではオブジェクト間のメッセージ通信のために通信先オブジェクトまでの経路情報が必要であり、この情報を格納するためにオブジェクトの近傍に配置されたメモリオブジェクト（ローカルメモリオブジェクト）が利用される。このように経路情報を機能オブジェクトと独立なメモリオブジェクトとして実現することは以下の利点がある。

i. メモリオブジェクトの内容を書き換えることでオブジェクト間の順序や関係を変更することができる。

ii. 機能部分の設計の共通化を図ることができる。

図4.2に示すようにローカルメモリオブジェクトと簡単

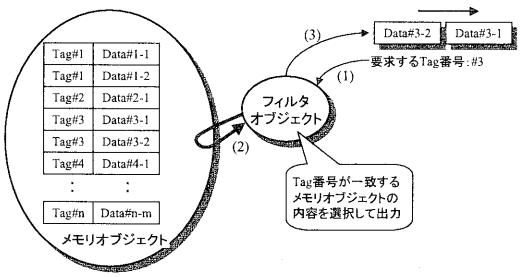


図4.2 ローカルメモリオブジェクトの使用例

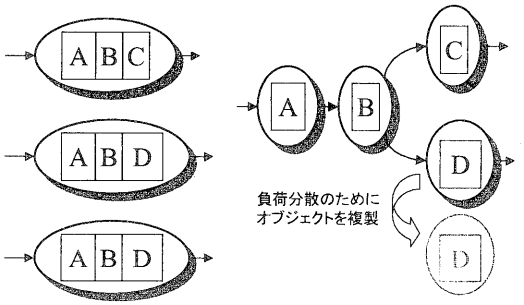
なオブジェクト(フィルタオブジェクト)とをみ合わせることでメッセージ送信先の変更やデータの選択を実現することができる。ローカルメモリオブジェクトに予めTag 番号とデータとを組みにしたデータを書き込んでおく。フィルタオブジェクトは Tag 番号を指定してデータを要求されると *config_out* 命令によりメモリオブジェクトの内容を順次読み出して指定された Tag 番号と組みになっているデータのみを要求元に応答する。メッセージ送信先を選択する条件をTag番号とし、経路情報をデータとしてメモリオブジェクトに書き込んでおくことで条件分岐、switch 文に相当する制御を実現することができる。また、コードを Tag 番号とし、デコード結果をデータとすることでデコーダを実現できる。

Tag 番号が 16 種類(4bit 幅)の場合、このようなフィルタオブジェクトは 2x2 個のセルで実現することができる。

4. 3 パイプライン化

上記2.4で述べたように、再構成可能ハードウェアである PCA を用いることで処理単位に合わせて独立に動作するオブジェクトを生成することが可能となる。一方、PCA はセルサイズに対してスケーラビリティを実現するアーキテクチャではあるが処理毎に全く独立なオブジェクトを生成した場合、リソースの使用効率が低下してしまう。このような使用効率の低下に対応するため、各処理が使用する一連のオブジェクトを分割してパイプラインのステージと見なし、処理間でオブジェクトの共有化を図る。

PCA 上でパイプラインを構成する方法の例を図4.3に示す。図4.3(a)では、3つの処理がそれぞれ独立に動作するオブジェクトで実行され、オブジェクトは A～D の4種類の機能で構成されている。図4.3(b)では各機能を独立なオブジェクトで構成し、各オブジェクトでの処理結果はメッセージとして次のオブジェクトに引き渡される。



(a)並列化　　(b)パイプライン化

図4.3 並列とパイプライン化

一般に、このようなパイプラインを構成するためにはオブジェクト間でメッセージの送受信を確認するための制御信号のやりとりが必要となるが、現在試作中の PCA チップは全て非同期回路を前提とした回路構成となっているため、オブジェクト間で、相手側オブジェクトの構成を意識した設計は不要である。オブジェクト間で単純にメッセージのやり取りをする回路は 1 個のセルで実現することができる。

非同期回路を前提とすることでオブジェクト間のインタフェースはオブジェクトに閉じた形で設計することが可能となったが、スループットを向上させるためには一連の処理でメッセージが通過するオブジェクト間の処理時間のばらつきが問題となる。一般にオブジェクトの処理時間の最小値は比較的容易に予測することが可能であるが、平均値や最大値を予め予測することは難しい。しかし、オブジェクトの機能を単純にすることで平均値や最大値の推定を容易にし、事前のシミュレーション等でボトルネック(過負荷)となる可能性の高いと判断されたオブジェクトは複数配置することで負荷分散を図る。また、オブジェクトでの処理時間のばらつきの影響を小さくするためには、通信路がメッセージをバッファリングする能力も重要である。通信先のオブジェクトが実行中状態で次のメッセージを受信できない場合でも送信元のオブジェクトは送信メッセージを通信路に出力することができれば後続のメッセージに対する処理を実行できるからである。通信路がメッセージをバッファリングする能力はオブジェクト間の経路の長さと組み込み部のバッファリング数の積で決定されるので、オブジェクトの配置と経路の選択を適切に行う必要がある。

また、ボトルネックとなるオブジェクトが時間的に変化するような場合には動作中にオブジェクトの複製を作成して負荷分散することも考えられる。この場合、ボトルネックとなっているオブジェクトを検出する方法[7]についてはオブジェクト毎に検討する必要があるが、16x16 個のセルから成るオブジェクトの複製を作成するための回路は 3 個のセルで実現することができる。

4.4 共有メモリの構成

再構成可能なハードウェアを用いることによって、処理の分散／並列化を図ることはできるが、各オブジェクトで実行する処理が共通にアクセスするデータを格納する共有メモリをスケーラブルに拡張可能なように構成する必要がある。

PCA では各セルが機能オブジェクトまたはメモリオブジェクトの何れとしても使用可能な構造を採用しているためので、複数のオブジェクトによって使用されるデータは、それらのオブジェクトが配置された近傍のメモリオブジェクトに格納することで共有メモリへの無用なアクセス競合を分散することができる。

ここでは、共有メモリの構成を検討するために、アレイ状に配置されたセルをアドレスで指定し、指定したセルにメッセージを送信する方法について検討した。共有メモリに大規模なデータを効率よく格納できるよう

に矩形のセルを使用し、アドレスによってセルの水平方向および垂直方向の位置を指定する。

図4.4にデータを格納するメモリアレイが 16x16 個のセルで構成される共有メモリ(16×16×1024×4bit)であるパイプラインメモリのブロック図を示す。ここで、網掛けの四角はオブジェクトを示し、オブジェクトの内部には受信したアドレスのビットパターンによってアドレスを水平方向または垂直方向に転送するスイッチおよびcolumnデコーダがある。columnデコーダはアドレスの列アドレス(この例の場合には下位4bit)で指定される水平方向のルート情報を組み込み部に出力するデコーダである。オブジェクト間の通信を行う垂直方向の矢印は組み込み部に設定される通信路である。

図4.4に示すパイプラインメモリは、オブジェクトがアドレスから垂直方向の位置を決定する row デコーダ(モジュールから CD を除いた部分)と水平方向の位置を決定する column デコーダから構成されていると見ることもできる。

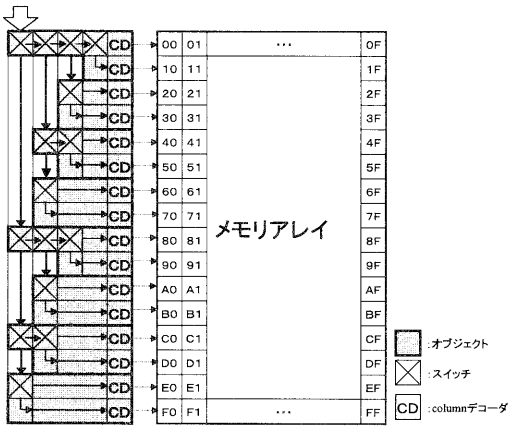


図4.4 パイプラインメモリのブロック図

メモリを構成するセルにアクセスするためのアドレスは左上のセルにメッセージとして入力され、順次各オブジェクト内の row デコーダでスイッチングされ何れかの column デコーダに到達する。各スイッチは受信したアドレスをパイプライン的に処理して次のスイッチに引き渡すことで、左上のセルからはスイッチングの周期でアドレスを入力することが可能となる。ここで、組み込み部を介して垂直方向のメッセージを転送する部分が問題となる。スイッチング処理毎に経路情報を生成してアドレスを転送すると経路情報を生成/出力するためのオーバヘットが発生してスループットが向上しない。他方、垂直方向の転送もプラスティック部に生成したデータパスで行う方法もあるが、組み込み部のデータ転送速度に比べて LUT を経由して行われるプラスティック部に生成したデータパスの転送速度は1/10以下である。垂直方向のデータパスの長さはメモリの高さに比例するため、この部分の遅延時間はメモリアクセスのレイテンシに影響する。このようなことから、垂直方向の経路情報はデコーダの生成時に予め設定し、アクセス毎に経路情報の出力を行わない方法を採った。これによって、メモリの row デコーダ上を垂直に通過するルートはオブジェクト間の通信路として使用できなくなる問題は生じるが、これは配置および通信ルート情報の生成側で解決することとした。

row デコーダを構成するスイッチはメモリアレイのサイズが実用的な範囲である 1024×1024 以下では 3×2 または 2×2 のセルで実現できる。また、4bit の分のアドレスを処理する column デコーダは 3x2 のセルで実現できる。このことから、サイズが $m \times n$のメモリアレイの row デコーダの幅は $2log_2n \sim 3log_2n$ となり、column デコーダの幅は $3(log_2n)/4$ となる。

パイプラインメモリはスイッチの動作速度によってそのスループットが制限されるが、図4.5(a)のようにデコーダを単純に追加することでスループットを増加させることができる。これはデコーダから出力する水平方向のアクセスが組み込み部を介したメッセージで行うようにパイプラインメモリが構成されているからである。

メモリのアクセスパターンによっては、特定の row アドレスにアクセスが集中する場合がある。また、メモリアクセスのレイテンシを短縮したい場合もある。これらのような場合には、図4.5の(b)に示すようにメモリアレイの高さを拡大する。これによって row アドレスへのアクセスを分散させることができる。また column アドレスでセルにアクセスするために生成するメッセージの数を削減され、レイテンシが短縮する。実際に要求スループットからパイプラインメモリの構成を決定する際には、

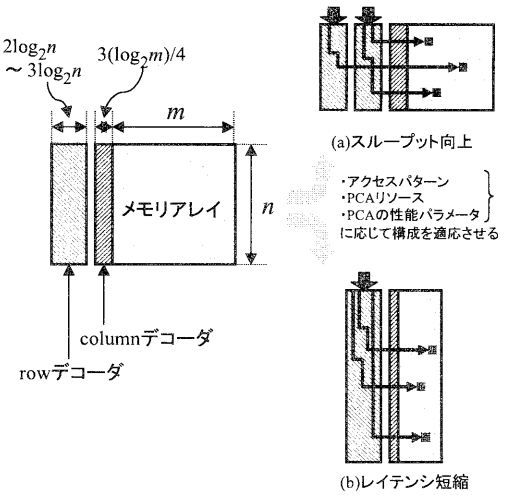


図4.5 要求性能への適応と構成の変更

各デコーダの動作速度および組み込み部のメッセージ転送速度等のチップ性能を考慮する必要がある。

このように、予め要求スループットを予測してメモリを構成するのではなく、変化する要求スループットに適応させて構成を変化させることでシステム全体のスケーラビリティを向上させることが可能となる。

パイプラインメモリのスループットに関して以下の条件でシミュレーション評価を行った。

- メモリアレイのサイズは$m \times n=2^{20}$とし、nを変化。
- アドレスはランダムパターン。
- 平均アクセス時間が待ち無しのアクセス時間の2倍以下となるスループットを測定。
- スイッチの動作速度は 4bit／25ns、column デコーダの動作速度はルート命令／20ns、組み込み部のメッセージ通信速度は 4bit／10ns とした。

図4.6にシミュレーション結果を示す。ここで、従来のメモリの実装密度は PCA の4倍とし、動作速度は4倍とした。

デコーダを増加させることによって、理想的にはスループットは比例して増加するはずであるが実際にはスループットの向上は比例しない。これは追加したデコーダとメモリアレイとの間でのルート情報の生成および row アドレスの競合によってアクセス時間が増加するためである。

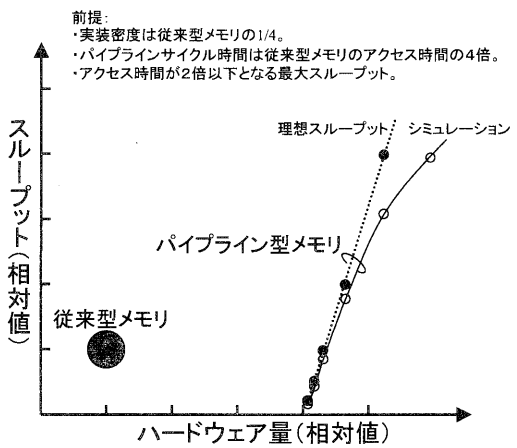


図4.6 共有メモリのハード量とスループット

5. おわりに

本稿ではPCA上に汎用情報処理機構を実現するためのオブジェクトの構成および幾つかの基本となるオブジェクトの構成と実装サイズについて述べた。さらに、複数のオブジェクトが共有するデータを格納するパイプラインメモリの構成について述べシミュレーション評価を行った。

今後は現在試作中のチップの性能パラメータに基づいた性能評価および高位記述言語からのオブジェクト合成／配置について検討していく。

謝辞

PCA のオブジェクトを記述するためのエディタを提供して頂いた京都大学大学院情報学研究科中村研究室の根本祐輔氏に感謝致します。

文献

[1]永見康一, 塩澤恒道, 小栗清, "自律再構成可能アーキテクチャ" , 設計自動化研究会, 87-3(1998).

[2]Kouichi Nagami, Kiyoshi Oguri, Tsunemichi Shiozawa, Hideyuki Ito and Ryusuke Konishi, "Plastic Cell Architecture: A Scalable Device Architecture for General-Purpose Reconfigurable Computing", IEICE Transactions on Electronics, vol. E81-C, No. 9, pp.1431-1437, September 1998.

[3]小栗清, "VLSI アーキテクチャと設計自動化技術の将来", デザインガイア'98, VLD98-38, ICD98-141, CPSY98-75, FTS98-65, 1998.

[4]Christoforos E. Kozyrakis and David A. Patterson, "A New Direction for Computer Architecture Research", IEEE Computer, pp.24-pp.32, 1998.

[5]天野英晴, "並列コンピュータ", 昭晃堂, 1996.

[6]塩澤恒道, 小栗清, 永見康一, 伊藤秀之, 小西隆介, "汎用計算機構を実現する再構成可能 LSI アーキテクチャ", VLD97-114, ICD97-219, 1998.

[7]塩澤恒道, Norbert Imlig, 小栗清, 佐野浩一, "PCA による通信アプリケーションの一実現方法", 第 13 回パルテノン研究会, pp.57-pp.66, 1998.

[8]伊藤秀之, 小栗清, 永見康一,小西隆介, 塩澤恒道, "動的再構成可能計算機構のためのプラスティックセルアーキテクチャ", 計算機アーキテクチャ研究会,129-6, 1998.

[9]伊藤秀之, 小栗清, 小西隆介, 中田広, "非同期式動的再構成論理 LSI の設計", デザインガイア 99, 1999.

[10]Norbert Imlig, Tsunemichi Shiozawa, Kouichi Nagami, Ryusuke Konishi and Kiyoshi Oguri, "Communicating Logic: Digital Circuit Compilation for the PCA Architecture", DA Symposium'99, pp101-pp.106, 1999.