計算機アーキテクチャ　140－8
（2000．11．30）

PARSプログラミングモデルとPARSアーキテクチャの提案

谷川 一哉　弘中 哲夫　吉田 典可

広島市立大学大学院 情報科学研究科

本稿では再構成型コンピュータを利用し並列実行を前提とする，PARSプログラミングモデルを提案する．PARSプログラミングモデルでは，アルゴリズムがもつ並列性を損なうことなく，ハードウェアで利用できる．またPARSプログラミングモデルに基づくPARSアーキテクチャの検討を行う．PARSアーキテクチャの検討ではハードウェアの再構成にかかる時間を短縮することに重点をおく．PARSアーキテクチャでは機能を提供する素子をALUを基に設計し，素子間の結線方式に複数の結線で1つの接続情報を共有するBus方式を採用することにより，再構成に必要な情報を減少させ，再構成時間の短縮を目指す．

PARS Programming Model and PARS Architecture

Kazuya Tanigawa　Tetuo Hironaka　Noriyoshi Yoshida

Graduate School of Information Sciences, Hiroshima City University

A PARS(PARallel Structure) programming model suitable for parallel execution in reconfigurable computers is proposed. By using the model, it will be easer for the parallelism available in the algorithm to be directly exposed to the hardware. We also propose a PARS architecture supporting the PARS programming model. On the study of PARS architecture, our interest was focused on reducing the time of reconfiguration, to change the hardware structure. The PARS architecture reduce configuration data needed for reconfiguration, by applying a function unit organization based on ALU, and by applying bus interconnecting mechanism which share some connection configuration data with several connections, to reduce configuration time and data.

1　はじめに

近年，処理速度を向上させる手法の1つとして並列処理が広く使用されている．並列処理においてはアプリケーション毎に最適な並列処理方式が異なる．そのため使用するコンピュータの構成が固定であれば，アプリケーションの性能を最大限に引き出すことができないといった問題がある．そこでこの問題点に対処するため，アプリケーション毎にコンピュータの構成が変更できる再構成型コンピュータに注目した．

再構成型コンピュータではアプリケーションに特化した並列化手法をハードウェアで実現できる．並列化されたアルゴリズムの並列性を損なうことなく，再構成型コンピュータで利用するためには，並列処理を前提とする言語が必要になる．また，ここでは再構成型コンピュータを利用することを前提としているため，再構成型コンピュータの特性を反映した言語が望ましい．

本稿では上記の要求を満たす言語として並列中間言語を提案し，並列中間言語を用いたプログラミングモデルとしてPARS(PARallel Structure)プログラミングモデルを提案する．PARSプログラミングモデルでは，プログラミング言語とハードウェアで実行可能なオブジェクトコードの間に並列中間言語を設けることにより，1) プログラマはハードウェアに依存しないでソフトウェア開発ができる，2) 異なるハードウェア間で並列中間言語で記述されたプログラムの互換性を保てる，という利点もある．

また本稿ではPARSプログラミングモデルに基づくPARSアーキテクチャを提案する．PARSアーキテクチャではハードウェアの構成を変更する際のオーバーヘッドを低減することに重点を置く．

以降，本稿は次のように構成されている．まず，次節で並列中間言語の詳細について述べる．3節ではPARSプログラミングモデルをサポートするPARSアーキテクチャを提案する．また，PARSアーキテクチャの構成に関する検討も行う．4節ではPARSアーキテクチャの構成を検討するために行った評価と評価結果について述べる．5節で関連研究について説明し，最後に6節でまとめる．

2　PARSプログラミングモデル

2.1　並列中間言語に必要な特徴

並列中間言語は，プログラミング言語と再構成型コンピュータの間の橋渡しをする，インターフェースのようなものである．そのため，並列中間言語は両方の特徴を備え持つ必要がある．

一般的に再構成型コンピュータにおいては機能を提供する構成要素と素子間の結線を提供する配線資源から構成される．この構成要素で提供される機能と配線資源の構造はプロセッサアーキテクチャに強く依存する．しかし，プログラミング言語では記述できる処理内容や処理間のデータの受け渡しに制限はない．

これらのことから並列中間言語が兼ね備えなければならない特徴は以下のようになる．

- 構成要素で提供される機能の中で最も基本的な機能だけが並列中間言語で定義される．そ

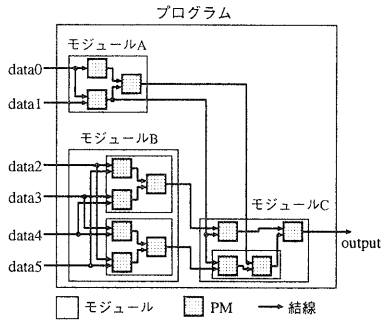


図 1: PARS-PIL 記述の回路図表現

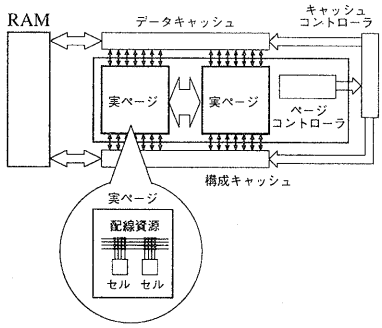


図 2: PARS アーキテクチャの構成図

の他の機能はそれらの基本機能を組み合わすことにより提供される.

- 並列中間言語では構成要素間の結線を自由に行える. これは並列中間言語で記述できる処理の制限を少なくするためである. つまり, 並列中間言語において配線資源の構造は定義されない.

以上の特徴を持つ並列中間言語を用いたプログラミングモデルがPARS プログラミングモデルである.

2.2 並列中間言語 PARS-PIL

本項では前節で述べた特徴を持つ並列中間言語 PARS-PIL (PARS - Parallel Intermediate Language) を提案する.

PARS-PIL は, Verilog-HDL で記述されたプログラムを論理合成して作成した, ネットリストのようなものである. つまり, 基本的な構成要素とそれらを結合する結線からなる.

PARS-PIL は以下のような構造を持つ.

- PARS-PIL はモジュールと結線の記述から構成される. モジュールは機能を提供し, 結線はモジュール間のデータの受け渡しを表現する.
- モジュールは以下に述べる PM やさらに小さなモジュールと (以後「サブモジュール」と呼ぶ), それらを結ぶ結線から構成される.
- PM(Primitive Module) は必要最小限の機能を提供する基本モジュールである. PM は 1 ビットの機能を提供し, その機能は加算, 補数処理, 一致判定, AND, OR, XOR, NOR, データの記憶, データの選択の 9 つの機能である. なお, 加算, 補数処理, 一致判定の機能は複数の PM を組み合わせて使用することにより多ビットの機能に対応する.

図 1に PARS-PIL の記述を回路図表現した図を示す. 同図においては PARS-PIL の記述を回路図表現しているが, 実際の PARS-PIL の記述はネットリストのように記述される.

図 1ではモジュールは長方形で表し, 結線は実線の矢印で表している. PM は網目をかけた正方形で表す. 例えばモジュール B は 2 つのサブモジュールから構成されている.

図 1のように PARS-PIL のモジュールは階層構造を持ち, 最上位のモジュールは中間言語で記述されたプログラム自身である. 最小のモジュールは PM である. このように PARS-PIL のプログラムは最終的に PM と結線に展開される.

3 PARS アーキテクチャ

3.1 並列中間言語のマッピング

並列中間言語では, 大きさに制限のない仮想ハードウェアを対象として, 目的の処理が記述される. PARS アーキテクチャでは, その並列中間言語を固定の大きさを持つページに分割することにより, 実ハードウェアで実行できる形式となる. そのページの大きさは実際のプロセッサに依存する.

このように並列中間言語で記述された処理をページに分割し, プロセッサで提供されている機能を使用した記述に変換したプログラムがオブジェクトコードである.

3.2 PARS アーキテクチャの構成

図 2に PARS アーキテクチャの構成図を示す. 同図において矢印はデータパスを表す. 以下に PARS アーキテクチャを構成する各ユニットについて説明する.

- RAM にはオブジェクトコード (再構成情報) とオブジェクトコードで使用するデータが格納される.
- 構成キャッシュはオブジェクトコードのページを格納するキャッシュである.
- データキャッシュはオブジェクトコードで使用するデータを格納するキャッシュである.
- 実ページ (Real Page; RP) はセルと配線資源から構成される. なお, 図 2では 2 つの実ページからなる 2RP 構成を示しているが, PARS

アーキテクチャではプロセッサの実ページ数を規定しない．

- セルは機能を提供し，セルの構成情報により機能が指定される．
- 配線資源はセル間の結線を提供し，結線の接続情報によりセル間の結線が指定される．
- ページコントローラは動的にページの実行順序を制御する．そのため，例えばデータ駆動的にページの実行を制御することもできる．
- キャッシュコントローラはページコントローラの情報を基にページやデータのプリフェッチをする．

3.3 オブジェクトコードの実行とページの再構成

PARSアーキテクチャでは以下のようにオブジェクトコードが実行される．オブジェクトコードはRAMに格納され，必要に応じてページコントローラがページを実ページにロードする．実ページではロードされたページの実行が開始する．実ページはロードされたページの実行が終了すると，ページコントローラに次に実行するページの要求を出す．これが繰り返されることによりオブジェクトコードが実行される．

このようにPARSアーキテクチャでは，実ページで実行するページが次々と切り替えられることにより，オブジェクトコードの実行が進行していく．このページの切り替えのことを再構成と呼ぶ．再構成にかかる時間はオブジェクトコードの実行から見ればオーバヘッドである．そのため，PARSアーキテクチャでは再構成にかかる時間を短縮することが重要な問題となる．

3.4 再構成時間を短縮するための構成方式の検討

本項では，PARSアーキテクチャでの再構成時間を短縮するために，1つのRP(実ページ)が必要とする情報の減少に焦点を当て，構成方式を検討した．実ページが必要とする情報はセルの構成情報と結線の接続情報である．

3.4.1 セルの構成情報の減少

セルの構成情報を減少させるために2つの観点から検討する．1つは，セルで必要とする構成情報そのものを減少させるように，セルの構成法を工夫する方法である．もう1つはセルが提供する演算のビット幅を調整する方法である．本稿ではこのセルの演算が何ビットで行われるかを表したものを演算ビット幅と呼ぶ．

(1) セルの構成法 セル構成法に関してLUT(Look Up Table)またはALUのどちらを基にして構成するか検討した．

LUT LUTを基にした構成ではセルで実現できる機能に柔軟性を持たせることができる．しかし，LUTを基にすると4入力1出力LUTの場合，16ビットの構成情報が必要になる．

ALU ALUを基にした構成において，最低限必要な演算数が8種類～16種類であると仮定すると，構成情報は3～4ビットで実現できる．そのため，このセル構成では大幅に構成情報を削減できる．

そこでPARSアーキテクチャのセル構成ではALUを基にしてセルを構成する．

(2) セルの演算ビット幅 セルの演算ビット幅の決定においてはセルの構成情報とセルの占有面積に関するトレードオフが存在する．

セルの構成情報においては，演算ビット幅を大きくするにつれ，必要な構成情報は減少する．例えば，4ビット演算をセルを用いて実現する場合，演算ビット幅が1ビットのセルではセル4つ分の構成情報が必要になる．演算ビット幅が4ビットのセルではセル1つ分の構成情報でよい．

また，演算ビット幅がプログラムで使用するデータ幅以上の場合，必要な構成情報は等しくなる．つまり構成情報を最小にするためには演算ビット幅をできるだけ大きくすればよい．

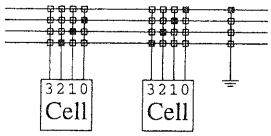
例えばあるプログラムで使用するデータ幅が5ビットであった場合，1ビットで演算するセル（1ビットセル）であれば5つのセルで1つの演算ができ，4ビットで演算するセル（4ビットセル）であれば2つのセルで1つの演算ができる．この場合4ビットセルの方がセル数が少ないので構成情報が少ない．

しかし，演算ビット幅が大きくなるにつれ，アプリケーション毎に異なるデータ幅に対して冗長な面積を含みやすいという問題もある．例えば前述の例で4ビットセルの場合，2つのセルを使用し8ビット分の演算機能で5ビットの演算を実現している．

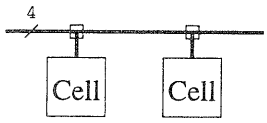
一方，1ビットセルでは5つのセルを使用しているが，5ビット分の演算機能で5ビットの演算を実現している．そのため4ビットセルは必要以上のビット幅の演算機能を提供していることになり，その冗長な演算機能に占有されている面積は有効に利用されていない．ただし，ここでは配線資源に関して考慮していない．

ハードウェアを有効に利用することを考えると，セルの構成情報とセルの占有面積に関するトレードオフが存在することになる．このことはPARSアーキテクチャで実行対象とするアプリケーションを限定していない場合に言える．アプリケーションを特定した場合，最適な演算ビット幅はアプリケーションの特性に依存する．

このようなトレードオフが存在するため，次節でnビットのデータ幅を持つ演算を実現するのに必要なセルの占有面積に関する予備評価を行う．



(a) Wire 方式



(b) Bus 方式

図 3: 結線方式

3.4.2 接続情報の減少

次に接続情報を減少するための結線方式について検討する．ここで結線方式とはセル間で必要な結線の接続情報の保持方式を表す．本稿では Wire 方式と Bus 方式について検討する．それぞれの方式を図 3に示す．Wire 方式と Bus 方式では同じアプリケーションでも必要なセル数，結線数が異なる．

Wire 方式 この方式は 1 ビットの結線毎に接続情報を設定する方式である．そのため，結線の自由度が大きく，結線だけでシフト演算が実現できる．例えば，図 3(a) において左のセルの 4 ビット出力 out[3:0] を 1 ビット左シフトし，右のセルの 4 ビット入力 in[3:0] とする場合を考える．同図において，黒の四角は接続されていることを示し，白抜きの四角は接続されていないことを表す．この場合 out[2:0] が in[3:1] に接続され，in[0] には定数 0 が入力されるため，1 ビット左シフトしたのと同じ効果が得られる．このように Wire 方式におけるシフト演算は結線を工夫することにより実現できる．ただし，Wire 方式の欠点は接続情報が多くなることである．

Bus 方式 この方式は数ビットの結線で 1 つの接続情報を共有する方法である．そのため Bus 方式は接続情報を少なくすることができる．Bus 方式の欠点は，結線の自由度が小さく，セルでシフト演算をサポートする必要があることである．

このように Wire 方式と Bus 方式ではシフト演算の実現方法に違いがある．そのため同じアプリケーションでも必要なセル数，結線数が異なる．よって次節で結線方式に関して予備評価を行う．なお，シフト演算以外にも結線だけで実現できる演算が他にも存在するが，本稿では最もよく使用されるシフト演算に着目する．

4 予備評価

本節では PARS アーキテクチャの構成を検討するために予備評価を行う．今回は n ビットの演算で使用する面積の評価と結線方式に関する評価を行った．

4.1 評価内容

今回の評価において評価対象はセルの演算ビット幅が 1 ビット，2 ビット，4 ビット，8 ビット，16 ビット，32 ビットのセルとした．そのセルにおいては PARS-PIL の PM で提供している機能と同じ機能を実現した．

また n ビットの演算で使用する面積の評価ではセルを Verilog-HDL を用いて論理設計を行った．使用した設計ツールは Synopsys Design Analyzer の Version 2000.05 である．使用したライブラリは 0.35μm 東大版 EXD 社製ライブラリである．

また結線方式に関する評価では DCT 関数を使用した．DCT 関数は IJG(Independent JPEG Group) が配布している JPEG プログラム [1] の jpeg_fdct_islow 関数のフェーズ 1 の部分を利用した．そのフェーズ 1 では入力値のデータ幅を 8 ビットと仮定した．また，フェーズ 1 内で 16 ビット以内の定数値と乗算されているため，その定数値乗算処理部分，及びそれ以後の処理では 24 ビットのデータ幅が必要であるとした．

4.1.1 n ビットの演算で使用する面積の評価

この評価の目的は n ビットの演算を評価対象のセルを用いて実現した場合に，その実現に必要なセルで占める面積を評価することである．評価対象のセルをそれぞれ設計し，今回の評価においては n を 1 ～ 32 ビットとした．

また今回の評価では結線方式に Wire 方式と Bus 方式を用いた場合に対してそれぞれ評価を行った．ただし，Bus 方式ではセルでシフト演算をサポートする必要があるため，Bus 方式のみセル内にシフタを実装した．

4.1.2 結線方式に関する評価

この評価の目的は Bus 方式がセルの構成情報と結線の接続情報に与える影響を評価することである．評価対象のセルに対して DCT 関数を実行する場合に必要なページの情報を評価した．

セルの構成情報は

(DCT 関数で必要なセル数)×(1 つのセルの構成に必要なビット数)

で求めた．必要なセル数はデータ幅も考慮して計測した．具体的には演算ビット幅 8 ビットのセル構成を用い，16 ビットのデータ幅をもつ演算を 1 つする場合には，必要なセル数は 2 つとした．

結線の接続情報は DCT 関数で必要な結線数を基に求めた．結線方式が Wire 方式の接続情報は DCT

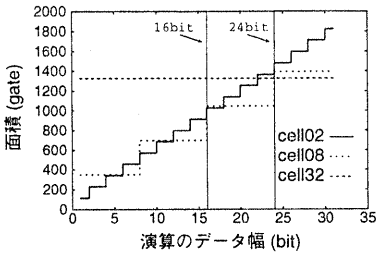


図 4: n ビットの演算で使用する面積の推移（Wire 方式）

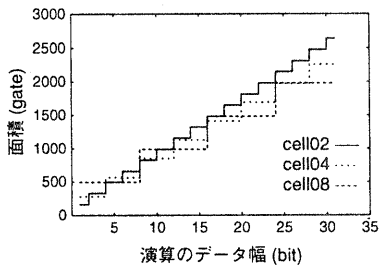


図 5: n ビットの演算で使用する面積の推移（Bus 方式）

関数で必要な結線数を 1 ビット 1 本として計測した．結線方式が Bus 方式の場合，接続情報は DCT 関数で必要な結線数を k ビット 1 本として計測した．ここで，k は Bus 方式で 1 つの接続情報を共有する結線のビット数である．

今回の評価において結線方式に Bus 方式を使用する場合，セルの演算ビット幅と同じビット数で 1 つのバスを構成するものとした．つまり，データ幅が 16 ビットの結線がセル間に 1 本必要な場合，Wire 方式では結線数を 16 本として計測した．Bus 方式では，セルの演算ビット幅に依存して結線数が異なり，仮にセルの演算ビット幅が 8 ビットであるとすると，必要な結線数は 2 本として計測した．

今回の評価においてはこのようにして計測された結線数を 1 本当り 2 ビットの接続情報が必要であるとした．この 2 ビットにした理由は 1 本の結線を表すためには最低 2 つのスイッチを on にする必要があるからである (図 3(a) 参照)．

4.2 評価結果

4.2.1 n ビットの演算で使用する面積の評価結果

図 4に Wire 方式のセルの評価結果を，図 5に Bus 方式のセルの評価結果を示す．両図において横軸は 1 つの演算のデータ幅を表し，縦軸は占有面積を NAND 換算のゲート数で表す．図が煩雑になるのを避けるため，1 ～ 32 ビットのデータ幅に対して占有面積が小さかった順に順位を付け，一番多く一位になった順に 3 つまでの評価対象の結果を示した．

Wire 方式　Wire 方式の結果の場合，演算ビット幅が 2 ビット，8 ビット，32 ビットのセルの評価結果を示す．図 4において cell02，cell08，cell32 はそれぞれ演算ビット幅が 2 ビット，8 ビット，32 ビットのセルの評価結果である．

同図において演算のデータ幅が 24 ビット以上の場合，演算ビット幅が 32 ビットのセルが面積最小となっている．しかし，演算のデータ幅が 16 ビット以下の場合に，演算ビット幅が 2 ビット，8 ビットのセルと比較して面積が非常に大きい．そのため演算ビット幅が 32 ビットのセルを使用し，演算のデータ幅が 16 ビット以下の場合，多くの面積が有効に利用されていないことが分かる．

また，演算ビット幅が 2 ビットと 8 ビットのセルを比較すると，演算ビット幅が 8 ビットのセルの方が，1 ～ 32 ビットの範囲内において，多くの演算のデータ幅に対してよい結果を示している．よって，Wire 方式における今回の評価においては演算ビット幅が 8 ビットのセルが最も有効にハードウェアの面積を利用していると考えられる．

Bus 方式　Bus 方式の結果の場合，演算ビット幅が 2 ビット，4 ビット，8 ビットの評価結果を図 5に示す．ここで演算ビット幅 8 ビットのセルが，1 ～ 32 ビットの範囲内において，多くの演算のデータ幅に対してよい結果を示している．よって Bus 方式における今回の評価においては演算ビット幅が 8 ビットのセルが最も有効にハードウェアの面積を利用していると考えられる．

4.2.2 結線方式に関する評価結果

図 6に結線方式の違いによる情報量の評価結果を示す．同図において横軸はセルの演算ビット数を表し，縦軸は全体の情報量を表す．全体の情報量は構成情報と接続情報の和で表す．

図 6において演算ビット幅が 1 ビットのセルでは Wire 方式と Bus 方式で全体の情報量は同じである．これは演算ビット幅が 1 ビットのセルにおいて，Wire 方式と Bus 方式に違いがないためである．

また，Wire 方式と Bus 方式のセルの構成情報を比較すると，Bus 方式のセルの構成情報の方が多い．これは，Wire 方式においてシフト演算を結線で実現しているのに対して，Bus 方式ではセルの演算で対応していることに起因する．そのためシフト演算の個数に応じて Bus 方式で使用するセルが増えている．特に今回使用した DCT 関数においては比較的シフト演算の割合が大きいので，Bus 方式のセルの構成情報は Wire 方式のセルの構成情報の約 2 倍となっている．

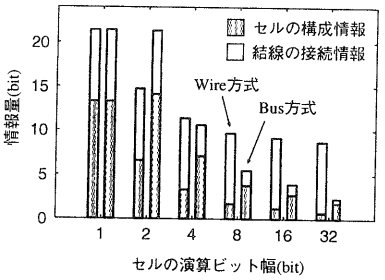


図 6: 結線方式の違いによる情報量の推移

このようにBus方式においてはセルの構成情報が多くなるといった欠点があるにも関わらず，結線の接続情報の減少により，演算ビット幅が4ビット以降でBus方式の接続情報はWire方式の接続情報よりも少なくなっている．よって今回のDCT関数を用いた評価においてはBus方式による接続情報の減少の効果は大きいと言える．

また今回の評価においては結線1本あたり2ビットの接続情報としたが，図3(a)のように結線されていないスイッチの情報も含めると，必要な接続情報はさらに多くなる．その場合には全体の情報量に対して接続情報が占める割合は多くなる．そのためBus方式による接続情報の減少の効果はさらに大きくなる．

5 関連研究

PARSアーキテクチャと同様にハードウェアとソフトウェアの融合を図ったアーキテクチャとしてPCA(Plastic Cell Architecture) [2] がある．PCAではオブジェクトと呼ばれる布線論理回路を必要に応じて生成・消去することにより処理を行う．しかしPCAでは，任意の大きさのオブジェクトを生成・消去するため，実ハードウェア上でのオブジェクトの管理が複雑になると考えられる．反対にPARSアーキテクチャでは，固定の大きさを持つページ毎に実行するため，ページの管理は簡単になると考えられる．

固定の大きさをもつページ単位で実行するアーキテクチャとしてHOSMII [3] がある．HOSMIIはDRAM型マルチコンテキストFPGAを利用しており，チップ内部に複数のページを蓄え，これらをデータ駆動的に実行することで処理を行う．PARSアーキテクチャとHOSMIIの相違点は再構成にかかる時間を減少させる方式のアプローチにある．HOSMIIではFPGAとメモリを混載させ，ページの転送にかかる時間を短縮している．PARSアーキテクチャではセルと結線の構成を工夫することによりページ情報を減らし，キャッシュメモリと組み合わせることでページの転送にかかる時間を短縮するアプローチをとる．

6 まとめと今後の課題

本稿では再構成型コンピュータを利用し，並列実行を前提とするPARSプログラミングモデルを提案した．PARSプログラミングモデルでは並列中間言語の導入により，アプリケーションが持っている並列性を損なうことなく，その並列性を再構成型コンピュータで利用できる．

このPARSプログラミングモデルに基づくPARSアーキテクチャを提案し，その構成に関して予備評価をした．まず実行対象とするアプリケーションを限定しない場合のPARSアーキテクチャにおいて，1～32ビットのデータ幅を持つ演算を実現した場合，何ビットで演算をするセルがハードウェア面積を効率よく利用するかを評価した．今回の評価では8ビットで演算をするセル構成がよい結果を示した．また結線方式による接続情報の減少の効果をDCT関数を用いて評価した．その結果，複数の結線で1つの接続情報を共有するBus方式は有効であることが分かった．

今回の構成要素に関する評価において論理設計のみで構成要素の面積を評価した．そのため構成要素の面積に配線面積が含まれていないので，配線面積を含めて構成要素の面積に関する評価をすることが今後の課題の1つである．その他の課題としてはPARSアーキテクチャの配線資源の構成方法に関しても検討すること，結線方式に関してシリアル転送タイプについても評価すること，などが挙げられる．

謝辞 本研究を進めるにあたり，貴重なご意見を頂いた広島市立大学大学院情報科学研究科の佐々木敬泰氏，木山真人氏に深く感謝します．また本稿での評価は東京大学大規模集積システム設計教育研究センターの協力で行われた．

参考文献

[1] ftp://ftp.uu.net/graphics/jpeg/jpegsrc.v6b.tar.gz

[2] N. Imlig, R. Konishi, T. Shiozawa, K. Oguri, K. Nagami, H. Ito, M. Inamori, H. Nakada, "Communicating Logic : An Alternative Embedded Stream Processing Paradigm", Proceedings of the Asia and South Pacific Design Automation Conference, pp.317-322, 2000.

[3] 柴田裕一郎, 宮崎英倫, 高山篤史, 凌暁萍, 天野英晴, 「DRAM混載FPGAを用いたデータ駆動型仮想ハードウェア」, 情報処理学会論文誌, Vol.40, No.5, pp.1935-1946, 1999年5月.