動的再構成可能プロセッサVulcan2 とそのソフトウェア開発環境ISAcc に関する研究

平木, 哲夫
九州大学大学院システム情報科学府

門内, 伸吾
九州大学大学院システム情報科学府

山崎, 陽介
九州大学大学院システム情報科学府

神戸, 隆行
福岡知的クラスター研究所

他

出版情報：電子情報通信学会技術研究報告, RECONF2007-13. 107 (41), pp.73-78, 2007-05.
IEICE(RECONF)
バージョン：
権利関係：

動的再構成可能プロセッサ Vulcan2 と そのソフトウェア開発環境 ISAcc に関する研究

平木　哲夫[†]　門内　伸吾[†]　山崎　陽介[†]

神戸　隆行[††]　LovicGAUTHIER[††]　VictorMAURO GOULART FERREIRA[††]

AntoineTROUVE[†††]　井上　弘士[††††]　村上　和彰[††††]

† 九州大学大学院システム情報科学府　〒 819–0395 福岡市西区元岡 744

†† 福岡知的クラスター研究所　〒 814–0001 福岡市早良区百道浜 3–8–33 福岡システム LSI 総合開発センター

††† 財団法人九州システム情報技術研究所　〒 814–0001 福岡市早良区百道浜 2–1–22 福岡 SRP センタービル 7F

†††† 九州大学大学院システム情報科学研究院　〒 819–0395 福岡市西区元岡 744

あらまし　特定用途向けプロセッサとは，アプリケーションに特化した命令を実行することによって，汎用プロセッサに対して高性能を実現するものである．本稿では，特定用途向けプロセッサの実現方式として，データパスに動的再構成可能ハードウェアを用いたプロセッサ Vulcan2 とそのソフトウェア開発環境 ISAcc を提案する．また，実際に ISAcc を用いてアプリケーションを Vulcan2 シミュレータ上に実装した結果を解析し，Vulcan2 及び ISAcc の評価を行った．

キーワード　動的再構成可能プロセッサ，特定用途向けプロセッサ，Vulcan2，ISAcc

A Study of the Dynamic Reconfigurable Processor Vulcan2 and Its Development Tool ISAcc

Tetsuo HIRAKI[†], Shingo KADOUCHI[†], Yosuke YAMAZAKI[†],

Takayuki KANDO[††], Lovic GAUTHIER[††], Victor MAURO GOULART FERREIRA[††],

Antoine TROUVE[†††], Koji INOUE[††††], and Kazuaki MURAKAMI[††††]

† Department of Informatics, Graduate School of Information Science and Electrical Engineering, Kyushu University　Motoka, 744, Nishi-ku, Fukuoka 819–0395 Japan

†† Fukuoka Laboratory for Emerging and Enabling Technology of SoC　Institute of System LSI Design Industry, Fukuoka, 3–8–33, Momochihama, Sawara-ku, Fukuoka, 814–0001, Japan

††† Institute of Systems and Information Technologies/KYUSHU　Fukuoka SRP Center Building 7F, 2-1-22, Momochihama, Sawara-ku, Fukuoka, 814–0001, Japan

†††† Dvision of Informatics, Graduate School of Information Science and Electrical Engineering, Kyushu University　Motoka, 744, Nishi-ku, Fukuoka 819–0395, Japan

Abstract　Application specific extensions of a processor provide higher performance. In this paper, the authors propose“Vulcan2” the Application specific processor with dynamically reconfigurable datapath and “ISAcc” Vulcan2’s development tool, and demonstrate the efficiency of the proposed processor.

Key words　Dynamic Reconfigurable Processor，Application Specific Instruction-set Processor, Vulcan2，ISAcc

1. はじめに

近年，携帯電話やデジタルカメラ，PDA などの急速な普及に伴い，それらの製品に用いられるシステム LSI には高性能や柔軟性，短 Turn Around Time(TAT) が求められている．これらの要求に対して，従来は汎用プロセッサと専用の Application Specific Integrated Circuit(ASIC) を搭載したシステム LSI を用いることで高性能化の要求を満たしてきた．しかし，この手

法では ASIC のレジスタ転送レベル設計や ASIC を並列に動作させるための並列プログラミングに開発工数がかかり，短 TAT を実現することが困難になってきている．そこで，アプリケーションに特化した命令セットを備えた特定用途向けプロセッサをアプリケーション毎に生成することによって，高性能と設計期間の短縮を実現しようとする手法が研究されてきた．特に最近では，汎用プロセッサに機能ユニットやコプロセッサを追加することによって特定用途向けプロセッサを実現するシステムが注目されている．これらは構成可能プロセッサと呼ばれ，汎用プロセッサに新規命令を追加するための機能ユニットやコプロセッサを新規に設計，あるいはデータベースから選択し，ハードウェアとして実現するという特徴がある．これらはプロセッサの高速化という面に関しては強力な手法であるが，一方で命令を追加するための機能ユニットをハードウェアとして実現するため，一度生成したプロセッサは単一のアプリケーションに対してのみ有効であるということや，頻繁な規格変更があった場合は，また一からハードウェアを作り直さなければならないという問題点がある．そこで筆者らは，再構成可能ハードウェアで実現されたデータパスを搭載した特定用途向けプロセッサをを用いることで，複数のアプリケーションに対応可能かつ，チップ製造後にもハードウェアを変更可能なプロセッサが実現可能なのではないかと考え，その実現可能性について研究を行っている [1] [2] [4].

本稿では，2 章で再構成可能データパスを搭載した特定用途向けプロセッサ Vulcan2 の紹介を行い，3 章では動的再構成可能プロセッサ用のソフトウェア開発環境である ISAcc について紹介する．4 章では，ISAcc 及び Vulcan2 命令セットシミュレータを用いた実装実験の結果とその考察を述べ，最後に 5 章でまとめる．

2. 動的再構成可能データパスを搭載した特定用途向けプロセッサ Vulcan2

Vulcan2 とは，動的再構成可能ハードウェアをデータパスに持つことでアプリケーション毎に命令セットを再定義することが可能なプロセッサである．Vulcan2 に搭載されている動的再構成可能データパスは RDP(Reconfigurable Data Path) と呼ばれ，この RDP に読み込む構成情報により RDP にて行われる命令の処理機能が決定される．以下に，Vulcan2 の命令セットアーキテクチャとマイクロアーキテクチャそれぞれについて紹介を行い，Vulcan2 プロセッサのプログラム実行フローの説明を行う．また，動的再構成可能プロセッサの問題点である再構成時間などの再構成オーバーヘッドに対して，Vulcan2 が導入している手法について説明を行う．

2.1 命令セットアーキテクチャ

Vulcan2 は命令セットとして基本命令とアプリケーションに特化したカスタム命令を持つ．それぞれの命令の種類については以下の通りである．

- 基本命令

32bit の標準的 RISC 準拠の命令セット．

- カスタム命令

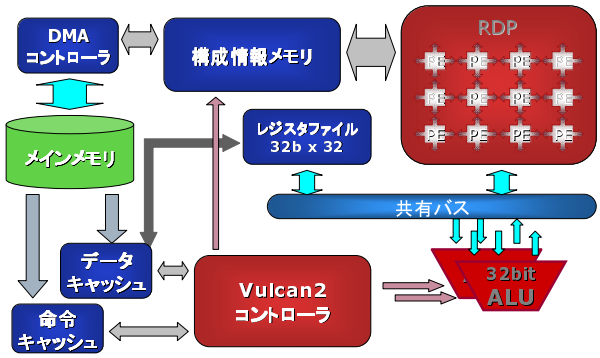

図 1　Vulcan2 アーキテクチャ

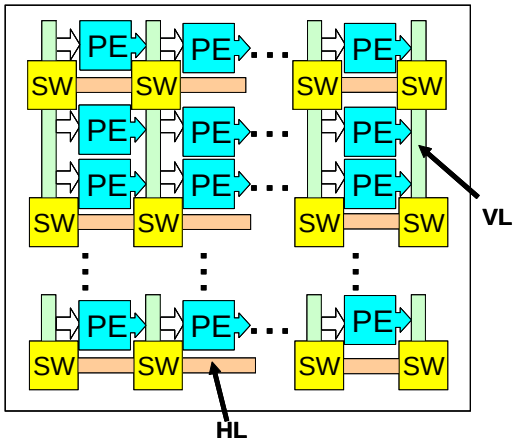

図 2　RDP の内部構造

対象アプリケーションに特化したカスタム命令を実行するために用いられる命令．RDP を再構成し実行する．カスタム命令は次の 2 つに分割される．

– prcf 命令

指定されたカスタム命令に対応する RDP の構成情報を構成情報メモリから RDP に読み込み，RDP の再構成の実行を指示する命令．再構成に複数クロックサイクル必要な場合もあるが，命令の完了を待たずに基本命令を実行することが可能である．

– ci 命令

RDP を用いてカスタム命令の演算の実行を指示する命令．処理内容は先に再構成された RDP の構成により決定される．4 つのソースオペランドと 2 つのディスティネーションオペランドを指定する．

2.2 マイクロアーキテクチャ

Vulcan2 アーキテクチャの概観を図 1 に示す．Vulcan2 は Vulcan2 コントローラ，RDP，ALU，構成情報メモリ，レジスタファイル，DMA(Direct Memory Access) コントローラ，メインメモリ，命令/データキャッシュから構成されている．以下に各ユニットついての説明を行う．

- Vulcan2 コントローラ

基本命令およびカスタム命令をデコードし，各ユニットへの制御信号を作成して全体の実行管理を行う．

- RDP

RDP の内部構造を図 2 に示す．RDP とはカスタム命令の演算

処理を行うデータパスであり，構成要素である PE(Programable Element) が 2 次元アレイ状に配置されている．PE 数や配置，配線リソース等はまだ決定されていない．PE は 6 入力 2 出力の LUT(Look Up Table) で構成されており，LUT 内に保持するデータによって入力に対して任意の論理関数を出力することができる．各 PE 間は，垂直方向の VL(Vertical Line) と水平方向の HL(Horizontal Line) で接続されている．VL が 1 列の各 PE に接続しており，各 VL は HL で接続されている．VL から HL，あるいは HL から VL への信号の受け渡しはスイッチ部が制御している．PE の LUT に保持するデータと PE の配線スイッチの状態を構成情報に基づいて変更することにより，任意の機能を実現することを可能としている．

- ALU

基本命令の演算処理を行う．RDP の再構成と並列に動作させることができる．

- 構成情報メモリ

各カスタム命令に対応する RDP の構成情報を格納するメモリ．Vulcan2 コントローラからの指示により RDP に構成情報をロードする．

- レジスタファイル

32bit の汎用レジスタを 32 個持つ．共有バスを介して RDP と ALU とで共有されている．このため，RDP と ALU でデータを並列に処理することはできない．

- DMA コントローラ

プログラム実行前にメインメモリに格納されている構成情報を，直接構成情報メモリに転送する．

- メインメモリ

基本命令，カスタム命令，各カスタム命令に対応する RDP の構成情報，そしてデータが格納されている．

- 命令/データキャッシュ

命令キャッシュには基本命令およびカスタム命令が格納されている．データキャッシュにはデータが格納されている．

2.3 プログラム実行フロー

Vulcan2 のプログラムの実行フローについて説明する．Vulcan2 では，命令を 5 つのステージから構成されるパイプラインで処理される．各ステージの処理について，説明する．

（1） IF(Instruction Fetch) ステージ

Vulcan2 コントローラが命令キャッシュから命令を読み出す．命令がキャッシュになければメインメモリから命令を読み出す．

（2） ID(Instruction Decode) ステージ

Vulcan2 コントローラへ命令をデコードし，レジスタファイルからレジスタの値を読み出す．prcf 命令の場合は再構成を行うカスタム命令の構成情報が格納されているアドレスを構成情報メモリに指定する．

（3） EXE(EXEcution) ステージ，または RC(ReConfiguration) ステージ

読み込まれた命令が基本命令および ci 命令の場合は EXE ステージ，prcf 命令の場合は RC ステージを実行する．

- EXE ステージ

演算を行う．ci 命令の場合は RDP を用い，基本命令の場合は ALU を用いて実行する．

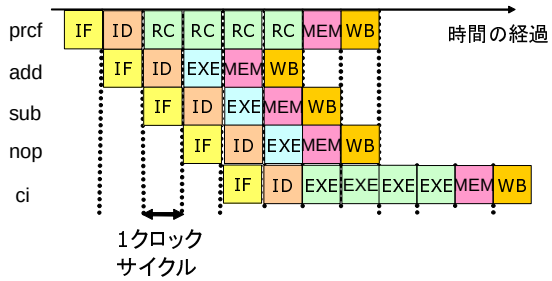

図 3　Vulcan2 のプログラム実行フローの例

- RC ステージ

指定されたカスタム命令の構成情報を構成情報メモリから RDP にロードし，RDP の再構成を行う．

（4） MEM(MEMory access) ステージ

メモリアクセスが発生した場合にメモリアクセスを行う．ロード命令の場合，データがデータキャッシュになければメインメモリからデータをロードする．

（5） WB(Write Back) ステージ

レジスタにデータを書き込む．

図 3 に，Vulcan2 での命令実行フローの例を示す．prcf 命令の RC ステージは RDP の再構成を行い，再構成が終了するまでこのステージを繰り返す．図 3 の例では，再構成が完了するまでに 4 クロックサイクルが必要であると仮定している．また，図 3 のように prcf 命令の RC ステージは ci 命令の EXE ステージ以外のステージとは並列に実行することが可能である．ci 命令が EXE ステージに遷移する際，RC ステージの実行が完了していなかったらストールする．ci 命令は入力として 4 つのレジスタと，出力として 2 つのレジスタを指定し，prcf 命令の RC ステージと同様に演算が終了するまで EXE ステージを繰り返す．ci 命令の EXE ステージと基本命令の EXE ステージを並列に実行することは出来ないため，ci 命令の EXE ステージが終了するまで次の命令はストールする．

2.4 再構成オーバーヘッド削減手法

次に Vulcan2 における再構成オーバーヘッドの削減手法について紹介を行う．動的再構成可能プロセッサにおける問題点の一つとして，データパス部の再構成に必要な時間がオーバーヘッドとなり性能に影響を与えることが挙げられる．Vulcan2 においては，以下の 2 つの方法により，再構成オーバーヘッドの削減を試みている．

（1） Vulcan2 では，カスタム命令の再構成と実行を異なる命令（prcf 命令，ci 命令）で処理を行う方法を用いている．このため，同じ種類のカスタム命令が連続して実行される場合においては一度再構成を行った後は RDP の構成を再利用するとで prcf 命令を実行する必要なく ci 命令を繰り返し行うだけでよい．このような方法をとることで，一つの命令で再構成と実行を行う場合に比べて無駄な再構成を行う必要がなく，再構成時間の削減が期待される．

（2） Vulcan2 では，基本命令と再構成命令の並列実行が可

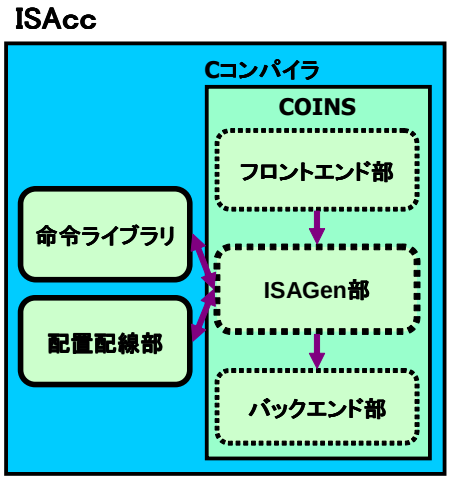

図 4　ISAcc の構成

能である．このため，再構成中であってもパイプラインをストールせずに基本命令を実行することが可能である．これにより再構成時間が基本命令の実行時間に隠蔽されることになり，再構成オーバーヘッドの削減が期待される．

3.　Vulcan2 専用ソフトウェア開発環境 ISAcc

Vulcan2 用のアセンブリコード，及び構成情報の生成ツールである ISAcc(Instruction Set Architecture Cross Compiler) についての紹介を行う．ISAcc は動的再構成可能プロセッサでカスタム命令を実現するための RDP の構成情報とそのカスタム命令を用いるオブジェクトコードの両方を，C 言語で記述されたアプリケーションのソースコードから生成する C コンパイラである [3]．ISAcc の構成を図 4 に示す．ISAcc は C コンパイラとカスタム命令生成の補助を行う命令ライブラリ，配置配線部からなっている．C コンパイラを構築するに当たっては COINS コンパイラインフラストラクチャを利用している．COINS とは研究目的でコンパイラを作成することを補助するフレームワークである．コンパイラ製作者は COINS の提供する枠組みに対し必要な部分のみを適宜置き換え，あるいは追加することで目指すコンパイラを得ることができる．また，COINS は Java で記述されており，様々なプラットフォームで動作する．COINS 内では処理の内容に基づき，フロントエンド部，ISAGen，配置配線部，バックエンド部の 4 つに分けられる．

3.1　フロントエンド部

C パーサであり，ハードウェア独立の最適化を提供する．殆どは COINS の標準のものを利用するが, 後段の ISAGen 部のために未使用コード除去（余分なカスタム命令の生成を抑制する）, ループ展開, 関数のインライン展開（定数伝播の効果を高める）, 定数伝播をより強化したものに置き換え, ビット幅解析と小さな if 文について基本ブロックを融合しデータフロー化する最適化を新たに追加している．

3.2　ISAGen 部

フロントエンド部から内部表現を受け取ってカスタム命令の利用による最適化が可能ならばカスタム命令に書き換え，カスタム命令を実現するために RDP 部の構成情報のネットリストを生成する．ISAGen の処理は基本ブロック内で局所的に行われるため，基本ブロックを跨いだカスタム命令は生成されない．ISAcc の心臓部であり, 以下のように動作する．

（1）　認識ライブラリを参照し, C 言語の構文の構造に対するパターンマッチングで書き換え可能な内部表現の項に注釈を付加する．

（2）　評価ライブラリを参照し，要素命令のスコアを計算する．

（3）　命令の組み立てスコアの結果を見ながら, 要素命令をグループ化し, どの要素命令をどのカスタム命令に含めるか（あるいはカスタム命令化しないものはどれか）を判定する．

（4）　生成グループに含まれる要素命令に対応する部分的なネットリストをライブラリから取り出す．

（5）　合成部分的なネットリストからカスタム命令のネットリストを合成して配置配線部を呼び出して渡す．失敗したカスタム命令のシグニチャ情報を記録し, 次からは配置配線を試みずに済ますようにする．

（6）　置換カスタム命令に置き換える部分の内部表現をカスタム命令を表す内部表現に書き換え, 必要な prcf 命令を対応するカスタム命令の直前に出力する．結果はバックエンド部へ送られる．

3.3　配置配線部

ISAgen 部から呼び出され, ネットリストを受け取ってターゲットに合わせ Independence/PathFinder アルゴリズムによって配置配線を行う．この際 ISAcc 独自の処理として, 必要ならば空いている PE を配線リソースとして用いるフォワーディングも行う．このフォワーディングは Independence アルゴリズムを若干拡張し, PE の配置時にルーティング・リソースを表す有向グラフのデータを書き換えることで実現している．

3.4　バックエンド部

ISAcc 独自の処理として ISAGen 部の出力した内部表現について prcf 命令をその関数内で可能な限り前方へ移動することでスケジューリングする．スケジューリング後は COINS の標準的なフローに従って内部表現をマシン語に置き換え，レジスタを割つけてアセンブリ・ソースを出力する．

4.　評価実験および考察

本実験では，ISAcc を用いて Vulcan2 シミュレータにアプリケーションを実装し，評価を行った．現在，FPGA 版の Vulcan2 の設計作業が進行中であり，まだ実ハードウェアへマッピングされていないため，基本命令の実行時間 (1 クロックサイクル) と prcf 命令と ci 命令の実行時間の比がまだ明らかになっていない．使用した Vulcan2 シミュレータは機能シミュレータであるが，再構成に必要なクロックサイクル数 (CC_{prcf})，カスタム命令の実行に必要なクロックサイクル数 (CPI_{ci}) をパラメータとして任意の値を入力することでプログラムの実行クロックサイクル数を見積もることが可能である．以下に実験の概要を示す．

4.1 実装アプリケーション

実装アプリケーションは DES(Data Encryption Standard) とした．DES [6] は米国商務省標準化局 (U.S.National Institute of Standards and Technology) で標準化されている暗号規格である．暗号化と復合化に同じ鍵を使用する共通鍵暗号で，1 ブロック 64bit の平文と 56bit の鍵を入力することによって，暗号文 64bit を出力する．本実験では DES 暗号化の実装は 56bit の鍵と 64bit の平文 16 ブロックを入力し，64bit × 16 ブロックの暗号文を出力させた．

4.2 実験方法

まず C 言語のソースコードを ISAcc を使用してコンパイルすることにより，Vulcan2 用のアセンブリコードと構成情報を取得する．Vulcan2 の RDP の PE の個数や配置はまだ決定されていないため，ISAcc でコンパイルする際には RDP の構造のパラメータの設定が可能である．本実験では，コンパイル時に RDP の PE 数の設定を 0，64，128，256，512 個と変更し，それぞれ取得したアセンブリコードを用いて Vulcan2 シミュレータにより実行命令数およびクロックサイクル数の計測を行う．各 PE 数の場合の RDP の構成は以下の通りである．

- PE 数:0

RDP 部を所有しておらず，基本命令の実行のみを行うアーキテクチャを想定．

- PE 数:64

PE が 16 列 4 行の配置で，128bit 幅の VL が 4 本，64bit 幅の HL が 4 本 (入力 2 本，出力 1 本，補助 1 本) の構成．

- PE 数:128

PE が 16 列 8 行の配置で，128bit 幅の VL が 8 本，64bit 幅の HL が 5 本 (入力 2 本，出力 1 本，補助 2 本) の構成．

- PE 数:256

PE が 16 列 16 行の配置で，128bit 幅の VL が 16 本，64bit 幅の HL が 7 本 (入力 2 本，出力 1 本，補助 4 本) の構成．

- PE 数:512

PE が 16 列 32 行の配置で，128bit 幅の VL が 32 本，64bit 幅の HL が 11 本 (入力 2 本，出力 1 本，補助 8 本) の構成．

4.3 プログラムの実装結果

DES 暗号化アプリケーションに対して ISAcc を用いて生成された Vulcan2 アセンブラの命令の内訳を図 5 に示す．図 5 では，ISAcc コンパイル時に設定する RDP の PE 数を 0，64，128，256，512 個と変更したときの命令数の内訳を示している．

図 5 から，使用可能な PE 数が増えると基本命令数が減少し，カスタム命令数が増加していることがわかる．このことから，PE 数が増えると一つのカスタム命令で実現可能な機能が高機能化し，カスタム命令化可能な処理が増えると推定する．しかしながら，PE 数 64 のように PE 数 0 よりも命令数が増える場合もある．これは，カスタム命令化しても全体の命令数が減らない命令をカスタム命令化していると推定されるため，ISAcc のカスタム命令生成の制約の設定を変更する必要がある．

4.4 プログラムの実行結果

図 6 に DES 暗号化アプリケーションにおける RDP の PE 数と Vulcan2 シミュレータでの実行命令数の内訳の関係を示す．図 6 から，PE 数が増加するとカスタム命令の実行の割合も増加していることがわかる．このことから，PE 数が増加すると 1 つのカスタム命令で実現可能な機能が高機能化し，カスタム命令化することによって性能向上が得られるであろう処理が増加したために，カスタム命令が増えたと推定する．

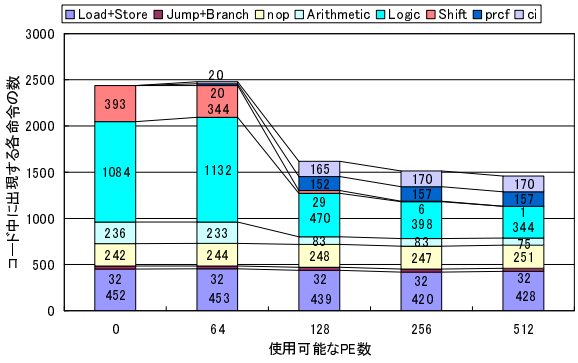

図 5　RDP の PE 数と Vulcan2 アセンブリコード中に出現する各命令数の内訳

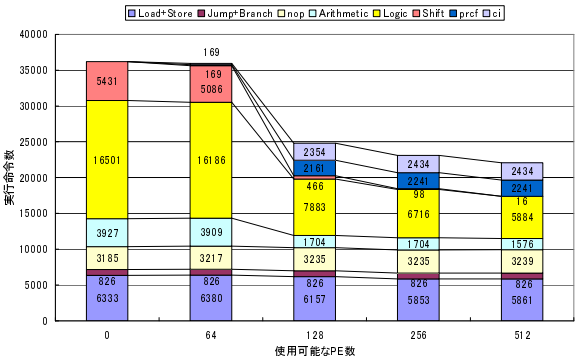

図 6　RDP の PE 数と実行命令数の内訳

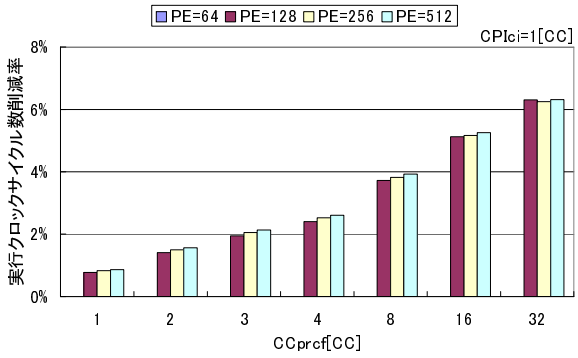

図 7　CC_{prcf} 及び RDP の PE 数と実行クロックサイクル数の削減率

4.5 再構成オーバーヘッド削減手法の効果

次に，同じカスタム命令が連続して実行される場合のカスタム命令の構成情報の再利用による再構成時間の削減効果，および基本命令と prcf 命令の並列実行による再構成時間の隠蔽効果について考察を行う．

4.5.1 構成情報の再利用による再構成時間の削減効果

図 7 に Vulcan2 シミュレータ上に DES 暗号化を実装した場

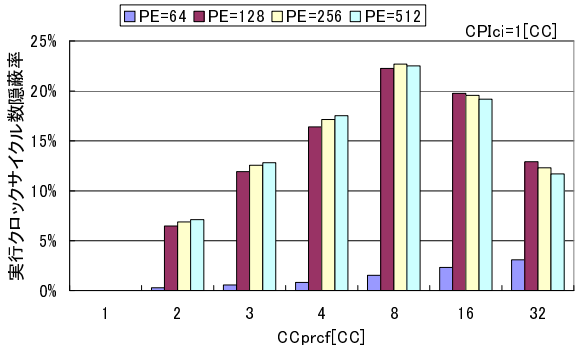

図 8　CC_{prcf} 及び RDP の PE 数と実行クロックサイクル数の隠蔽率

合の RDP の PE 数とカスタム命令の構成情報の再利用による実行クロックサイクル数の削減率との関係を示す．削減率とは，RDP の再利用をする手法を用いない場合の全実行クロックサイクル数を分母とし，削減手法を用いた場合に削減されるであろうクロックサイクル数を分子とすることで算出される．図 7 では ISAcc にて RDP の PE 数の設定を 64，128，256，512 個と変更した場合に対応するアセンブリコードを用いて，$CPI_{ci} = 1$ とし，CC_{prcf} を 1，2，3，4，8，16，32 と設定した際の実行クロックサイクル数の削減率を示している．

図 7 より，DES 暗号化の例では CC_{prcf} が増加すると削減率も増加することがわかる．これは，CC_{prcf} が増加すると，prcf 命令の削減により削減されるクロックサイクル数が増加するからである．また，CC_{prcf} が増加し 32 クロックサイクルになると，PE 数 256 より 128 の方が削減率が高くなってきている．これは，図 7 から CC_{prcf} の値が小さいときは使用可能な PE 数が増加することによってカスタム命令の機能が大きくなり，カスタム命令が増えることで prcf 命令が削減されたことによるメリットの方が大きかったが，CC_{prcf} の値が大きくなるにつれてカスタム命令の再構成時間の方が大きくなったためと推定される．

4.5.2　基本命令と prcf 命令の並列実行による再構成時間の隠蔽効果

図 8 に Vulcan2 シミュレータ上に DES 暗号化を実装した際の RDP の PE 数および CC_{prcf} と実行クロックサイクル数の隠蔽率との関係を示す．隠蔽率とは，再構成と基本命令を並列実行する手法を用いない場合の全実行クロックサイクル数を分母とし，並列実行を用いた場合に隠蔽されるであろうクロックサイクル数を分子とすることで算出される値である．$CPI_{ci} = 1$ とし，CC_{prcf} のを 1，2，3，4，8，16，32 と設定した時の実行クロックサイクル数をシミュレータを用いて計測した．

図 8 より，DES 暗号化においては CC_{prcf} が 8 以下までは，全ての PE 数で CC_{prcf} に伴い隠蔽率が増加をしているが，PE 数の上限が 128，256，512 個では $CC_{prcf} = 16$ で減少し始めている．このことから，RDP の PE 数それぞれにおいて，ある CC_{prcf} の数を超えた時点で prcf 命令と並列実行を行う基本命令の数が不足し始め，結果として CC_{prcf} を隠蔽出来なくなったと推定される．また，PE 数の上限が 128 個では $CC_{prcf} =$ 16，32，256 個では $CC_{prcf} = 8$，512 個では $CC_{prcf} = 4$ 以下でほかの PE 数よりも隠蔽率が高い．このことから，PE 数の増加に伴い，高い隠蔽率を示す CC_{prcf} の値は減少することがわかる．この理由は，PE 数が増加することによって 1 つのカスタム命令が実現できる機能の高機能化が進み，prcf 命令と並列実行される基本命令がカスタム命令内に取り込まれていくことで，prcf 命令と並列実行する基本命令の数が減少するためであると推定する．これにより，PE 数が増加することで prcf 命令と並列実行可能な基本命令の数が減少し，隠蔽が可能な CC_{prcf} の値が減少するために隠蔽率のピークとなる CC_{prcf} の値が小さな値になることがと推定される．

5. おわりに

本稿では，動的再構成可能データパスを搭載した特定用途向けプロセッサ Vulcan2 とそのソフトウェア開発環境である ISAcc を提案し，実際に ISAcc を用いて Vulcan2 シミュレータ上にアプリケーションを実装，および評価を行った．ISAcc を用いたアプリケーションの実装においては，C ソースコードから Vulcan2 アセンブリコード，およびカスタム命令の構成情報を自動的に取得することが可能であることが確認できた．また Vulcan2 で用いられている再構成オーバーヘッド削減手法についても，再構成時間の削減に効果があることが確認された．

今後は，Vulcan2 の FPGA 上へのマッピングを行い，カスタム命令の再構成時間や実行時間などのハードウェア依存の値を測定すると共に，ISAcc を用いて様々なアプリケーションの実装を行い，適切な RDP の構造の提案や ISAcc の改良を行っていく予定である．

文　献

[1] Victor M. GOULART FERREIRA, Lovic GAUTHIER, Takayuki KANDO, Takuma MATSUO, Toshihiko HASHINAGA, Kazuaki MURAKAMI, “REDEFIS: a system with a redefinable instruction set processor, ”Proceedings of the 19th annual symposium on Integrated circuits and systems design , SBCCI ’06, August 2006

[2] 橋永寿彦, Lovic Gauthier, 神戸隆行, Victor Mauro Goulart Ferreira, 薄田竜太郎, 平木哲夫, 山崎陽介, 長野孝昭, 村上和彰, “動的再構成可能プロセッサ Vulcan の評価”, 電子情報通信学会技術研究報告 VLD2005-98, pp.7-11, Jan. 2006.

[3] 神戸隆行, Lovic Gauthier, Victor Goulart, Antoine Trouve, 平木哲夫, 山崎陽介, 村上和彰, “Redefis：動的再構成可能プロセッサを対象とした自動 ASIP 生成技術 ～動的再構成可能プロセッサ Vulcan2，および，その開発ツール ISAcc～”, 情報処理学会アーキテクチャ研究会, Jan. 2007.

[4] 末吉敏則, 天野英晴, “リコンフィギャラブルシステム, ” オーム出版, Aug, 2005

[5] Gerry Kane, 前川守 “mips RISC アーキテクチャ - R2000/R3000-” 共立出版, Oct, 1992

[6] RFC2405 - The ESP DES-CBC Cipher Algorithm With Explicit IV
http://rfc.net/rfc2405.html