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

2007－ARC－171 (18)
2007－EMB－ 3 (18)
2007／1／23

Redefis：動的再構成可能プロセッサを対象とした自動ASIP生成 ～動的再構成可能プロセッサVulcan2, および, その開発ツールISAcc

神戸隆行[†] Lovic Gauthier[†] Victor Goulart[†]
Antoine Trouve[††] 平木哲夫[†††]
山崎陽介[†††] 村上和彰[†††]

RedefisはSoCの計算エンジンとなる動的再構成可能なASIP(アプリケーション特化命令セット・プロセッサ)を動的再構成可能プロセッサを用いて実現するプラットフォームである. Redefisは Redefisプロセッサとソフトウェア開発ツールである Redefis ツールから成る. Redesis プロセッサは実行時再構成可能な LUT ベースの演算装置を備え, それをカスタム命令を介して利用できる. Redefis ツールは Redefis プロセッサでアプリケーションに特化したカスタム命令を実現するための演算装置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汎用の高級言語（ここでは 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. 以前開発した試作 Redefis プロセッサ Vulcan とその開発ツールとして Redefis ツール・チェインの経験を踏まえて新たな試作プロセッサとして Vulcan2 を開発しつつあり、また,Redefis ツールを C コンパイラの枠組みに統合して新たに ISAcc コンパイラ開発した.

Redefis: Automatic Generation of Dynamic Reconfigurable ASIPs - The Dynamic Reconfigurable Processor Vulcan2 and Its Development Tool ISAcc

Takayuki Kando,[†] Lovic Gauthier,[†] Victor Goulart,[†]
Antoine Trouve,[††] Tetsuo Hiraki,[†††] Yosuke Yamazaki[†††]
and Kazuaki Murakami[†††]

Redefis is a design platform for designing dynamically reconfigurable ASIPs (Application Specific Instruction Set Processors), which are going to be used as engines in future SoCs. The platform consists of the Redefis processor and its SW development toolset. The Redefis processor contains a LUT-based reconfigurable module capable to be reconfigured on-the-fly via custom instructions. The Redefis toolset analyzes the target application (written in high level C language) and generates specialized Custom Instructions which are referenced in the final compiled object code of the application. An early prototype of the Redefis processor, called "Vulcan" with its relative tool-chain have been developed. Based on the know-how obtained, a new prototype, "Vulcan2", and a restructured development toolset, called ISAcc, which integrates the previous Redefis design tool-chain into C compiler framework.

1. はじめに

近年, 組み込み機器においてもアプリケーションが複雑化, 大規模化し処理能力の向上が求められているが, 組み込み機器においては消費電力, コストなどの制約は厳しい. そこでシステムを1チップ上に集約するシステム・オン・チップ（SoC）などのアプローチが取られるようになった. これらSoCについて計算処理を行うエンジンとして汎用プロセッサやカスタム回路, または両者の組み合わせが用いられる. しかし汎用プロセッサでは高性能化に伴いクロック周波数が増加し, 消費電力が増加するとともに, 高品質な回路も必要となり製造コストの増加にも繋がる. 一方専用回路によるアプローチでは複雑化, 大規模化に伴い回路規

† 福岡知的クラスター研究所
Fukuoka Laboratory for Emerging and Enabling Technology of SoC
†† 九州システム情報技術研究所
Institute of Systems and Information Technologies / KYUSHU
††† 九州大学
Kyushu University

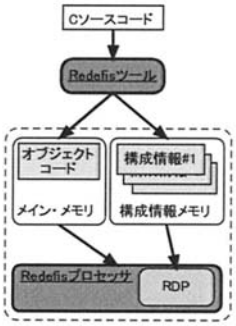

図 1 Redefis の構成

模が増大し, 設計・検証コストが増大するとともに, 市場へ供給するまでの時間も増大する. これらの欠点を補うため, アプリケーションに特化した命令セットを持つ特定用途プロセッサ (ASIP) が開発されるようになった. しかし ASIP 自身についても設計と検証問題はあり, これらの設計・検証コストを軽減するために動的（実行時）あるいは静的な再構成可能ハードウェア技術や回路の高位合成技術に基づく ASIP 開発の自動化技術がいくつか提案されてきている[13].

Redefis プラットフォームは SoC の計算エンジンとなる動的再構成可能 ASIP の自動生成を動的再構成可能プロセッサである Redefis プロセッサと Redefis ツールによって実現する (図 1) . Redesis プロセッサはリコンフィギュラブル・データ・パス (RDP) という LUT ベースの動的再構成可能な演算装置を備え, それをカスタム命令を介して一個の演算命令として利用できる. Redefis ツールは, 汎用の高級言語（ここでは 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, Redefis プロセッサでカスタム命令のための 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自動的に生成する.

試作 Redefis プロセッサ Vulcan とそのための Redefis ツールとして Redefis ツール・チェインについては既に開発, 評価, 報告した[3),4),6)~8)]. それらの経験を踏まえて新たな試作プロセッサとして Vulcan2 を開発を進めており, また Redefis ツール・チェインのツール群については, より安定で堅牢な C コンパイラの枠組みに統合して新たに ISAcc コンパイラ開発した.

以下,2 節で関連研究について述べた後, 3 節では現在開発中の新たな Redefis プロセッサである Vulcan2 について, また 4 節で新しい Redefis ツールである ISAcc について説明する. 5 節では簡単な例について実験した結果を示し,6 節でまとめと今後の課題について述べる.

2. 関 連 研 究

Redefis と他システムの特徴は以下のような軸にまとめて考えられる：

動的再構成と動的再構成の頻度 動的再構成可能とは単に再構成可能なだけでなく実行中に構成を変更できることであり, 利用状況に応じて動作を変えて HW 資源を無駄なく利用できると期待される一方, 実行時の再構成に掛かる時間がオーバーヘッドになると考えられる.

粒度と一様性 PE の粒度と一様性がより細粒度で一様であるほど, 自動化がしやすく, HW 資源が無駄なく利用できると期待される一方, 結線が長くなり実行時間が長くなり, 構成情報が増え再構成時間が長くなると考えられる.

高級言語から生成する際の自動化の度合い 高級言語で記述されたアプリケーション・プログラムを分析し,ASIP のカスタム命令を設計, 実現する際, 自動化の程度が高いほど設計・検証コストを抑制できるが, アプリケーションに特化した最適化技法を取り込むことが難しい.

以上の軸に沿って様々な再構成可能プロセッサを分類すると表 1 のようになる.

3. Redefis プロセッサ - Vulcan2

Redesis プロセッサは LUT ベースの一様で細粒度の処理要素 (PE) をバスで接続したリコンフィギュラブル・データ・パス (RDP) という動的再構成可能な演算装置を備え, それをカスタム命令を介して一個の演算命令として利用できる.

Vulcan2 は新たな試作 Redefis プロセッサであり, Vulcan[7)] について得られた経験を[4)] を元に, 現時点では機能レベルの仕様だけが設計完了し, それに基づいて機能レベルのシミュレータが完成した段階にあり,FPGA 上の試作について現在設計を行っている☆.

Vulcan2 は以下のような特徴がある：

再構成可能な演算装置 RDP RDP (図 2) は動的再構成可能な演算装置であり, 細粒度で一様な PE をバスで接続してさまざまな演算を実現する.

- PE としては 6 入力 2 出力 LUT を 22 個 6 段, 計 132 個を備えている.
- これら PE を接続するバスは LHL と呼ばれる 64 ビット幅バスを 4 組,LVL と呼ばれる 128 ビット幅バスを 6 組備え, これらがスイッチ VLSW で接続される☆☆. $n(n = 0...5)$ 段

☆ SoC とのインターフェースの枠組みはまだ定まっていないため, 現在は通常の RISC プロセッサと同様にメモリ空間からのみデータを得, メモリ空間にのみデータを出力できる.

☆☆ 各 LHL バスと各 LVL バスの交点にある $6 \times 4 = 24$ 個の VLSW は LHL の $i(i = 0...63)$ 番目のラインと LVL の i 番目及び $i + 64$ 番目とを接続する LSW スイッチの集まりであ

表 1　再構成可能プロセッサの比較

名前	PE 数	粒度	一様性	再構成頻度	高級言語記述から自動生成
Nios-II(Altera[1)])	8 個	粗粒度	8 種	静的	手動
Extensa(Tensilica[14)])	少数	粗粒度	多種	静的	自動
DRP-1 (NEC[10)])	512 個	粗粒度 (8bit ALU+DMU)	1 種	動的/フェーズ毎/部分再構成可能	自動
DAPDNA (IPFLEX[5)])	384 個	粗粒度 (16bit/32bit ALU)	20 種	動的/フェーズ毎	分割自動 構成情報は別に設計
XPP-III(PACT[11)])	80 個	粗粒度 (数個の ALU)	2 種	動的/フェーズ毎	自動
Redefis	**132 個**	**細粒度 ($6bit \times 2$LUT)**	**1 種**	**動的/命令毎**	**自動**

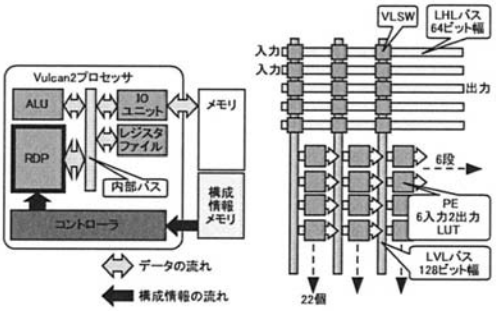

図 2　Vulcan2 とそのＲＤＰ部

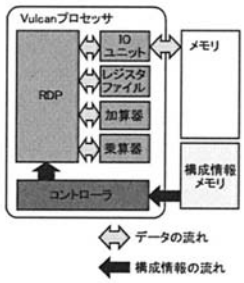

図 3　Vulcan

目の PE は n 番目の LVL バスから任意の 6 本を選んで入力を得,$n+1 \pmod 6$ 番目の LVL バスから任意の 2 本を選んで出力できる.

32 ビット Mips 準拠の基本命令セット　Vulcan2 は 32 ビットの Mips 準拠の命令セットを持ち, それらの命令は 5 段パイプラインで処理される☆. この基本命令セットに対して RDP 部に構成を設定する prcf 命令とその時点での構成に従った演算を実行する ci 命令が追加されている.

プリコンフィギュア (prcf) 命令　prcf 命令はどの構成を構成情報メモリから RDP 部に設定するかを指定する. prcf 命令は prcf 命令と ci 命令を除く基本命令セットの命令と並行して実行できる.

カスタム (ci) 命令　ci 命令には入力オペランドとして 32 ビットレジスタ 4 個, 出力オペランドとして 32 ビットレジスタ 2 個を指定できる☆☆.

る. 各 LSW は無接続あるいは,LVL から LHL への接続, またはその逆に LHL から LVL への接続の 3 種類の状態を取る.

☆ レジスタは Mips と同じく 32 ビットレジスタを 32 本備え, Mips と同じく $0 レジスタを読み出すと常時 0 であり, 書き込みは無視される.

☆☆ ci 命令が実行されると,RDP 部には各 ci 命令のオペランドとして指定されるレジスタ 4 個から合計 128 ビットの入力を 0 番目と 1 番目の LHL バスへ供給される. そして出力は 2 番目の LHL バスへ 64 ビットの出力を出し, この 64 ビットは各 ci 命令のオペランドとして指定される 2 個のレジスタへ書き込まれる.

3.1　Vulcan と Vulcan2

Vulcan[7] と実験[4] に基づくその改良である Vulcan2 では以下のような相違がある：

RDP の位置づけ　図 3 のように Vulcan では RDP 部は内部バスを兼ねており, 性能改善の見込みがなくてもカスタム命令を利用せねばならなかった. Vulcan2 では図 2 のようにデータの移動は専用のバスを介して行われ,RDP 部は ALU と並行して存在し, 純粋に演算を行う装置になっている.

基本命令セット　Vulcan では分岐やアドレス演算以外は全てカスタム命令であったためカスタム命令で全ての処理をカバーせねばならず, 大きなプログラムでは必要なカスタム命令が膨大な数になった. Vulcan2 では Mips 準拠の完備した基本命令セットを用意し, カスタム命令は処理が早くなる場合にのみ利用するようになった. また基本命令はパイプライン実行され RDP を必要としない演算を高速に実行できるようになった.

再構成のタイミング　Vulcan では再構成はカスタム命令の実行フェーズに含まれオーバーヘッドになっていた. Vulcan2 では prcf 命令を導入し, カスタム命令の実行（ci 命令）と再構成を分離し, 再構成と基本命令を並行に実行できるようになり, 再構成時間を隠蔽できるようになった. またループの外に再構成を追い出すことが出来ればループを効率よく実行できるようになる.

る.

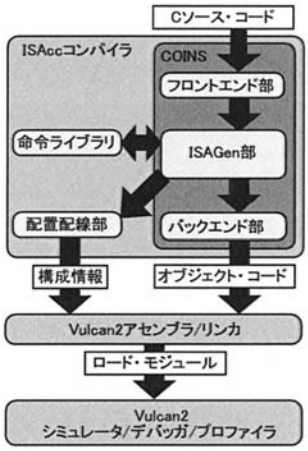

図 4 ISAcc の構成

PE 内メモリ Vulcan では各 PE が 2 ビット幅 512 ワードのメモリを備えていた. これは手で構成情報を作る場合には有効だったが, ツールから利用する際は配置配線の障害となった. Vulcan2 では PE 内のこうしたメモリは廃止された.

関数呼び出し Vulcan には関数呼出し命令がなかった. Vulcan2 では通常の関数呼び出しが可能になっている.

4. Redefis ツール - ISAcc

Redefis ツールは Redefis プロセッサでカスタム命令を実現するための演算装置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汎用の高級言語（ここでは 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.

ISAcc は Redefis ツールを ANSI C 言語規格に準拠した C コンパイラ☆1として統合したもの (図 4) である.☆2 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[2] を利用している.

COINS は研究目的でコンパイラを作成することを助けるフレームワークである. COINS ではコンパイラはパーサがソース・コードから内部表現を生成し, それを最適化パスが順次書き換え, 最後にバックエンドがマシン命令に書き換えながらレジスタを割付をしてアセンブラ・ソースを出力するというモデルを提供している☆3 ユーザは 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. 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.

☆1 ただし浮動小数点演算をサポートしない.

☆2 ISAcc 以外に周辺ツールとして機能レベル・シミュレータとそれに統合されたソース・レベル・デバッガやプロファイラ, 構成情報を可視化する Visualizer ツールも提供する.

☆3 COINS は標準の C フロントエンド, 中間表現, 標準的な最適化パス一式, メジャーなプロセッサ向けのバックエンドとバックエンド生成ツール, コンパイラ・ドライバなどを提供している.

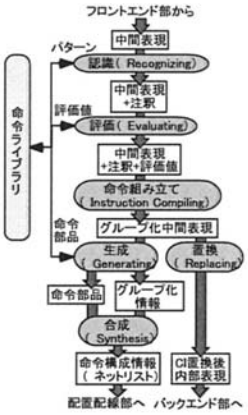

図 5 ISAGen 部の処理の流れ

4.1 フロントエンド部

C パーサ, ハードウェア独立の最適化☆4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（4.2）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☆5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.

ビット幅解析は区間演算の考え方を用い, 静的な解析で整数演算に必要なビット幅を見積もる☆6. この解析の結果ビット幅が小さくなれば一個のカスタム命令により多くの演算を詰め込めるようになる.

4.2 ISAGen 部

フロントエンド部 (4.1) 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☆7内で局所的に行われる. ISAcc の心臓部であり, 以下のように動作する (図 5).

(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

☆4 定数伝播, 共通部分式削除, 未使用コード除去, ループ展開, 関数のインライン展開など.

☆5 典型的には 3 項演算子で書かれる様なものであるが, 内部表現では 3 項演算子と if 文は区別されていないので if 文の中の計算の量がデータフロー化した際に RDP 部で計算できそうかどうかが基準となる.

☆6 この際, ループ展開, 関数のインライン展開, 定数伝播の結果を利用することで可能な限りビット幅を小さく抑える.

☆7 末尾以外に分岐を含まない最大のコード片. 関数は基本ブロックの集まりとして表される.

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 (4.3) を呼び出して渡す☆1. 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 4.4 へ送られる.

4.3 配置配線部

ISAgen 部 (4.2) 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[12),9)] 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, いわゆるフォー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☆2ことで実現している.

4.4 バックエンド部

ISAcc 独自の処理として ISAGen 部 (4.2) 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☆3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.

4.5 Redefis ツール・チェインと ISAcc

Redefis ツールチェイン[6)] と ISAcc はどちらも ISAGen の基本アルゴリズムを採用しているが,[4)] Vulcan から Vulcan2 への設計変更, 及び COINS コンパイラ・インフラストラクチャの採用により幾つかの点改善された：

(1) 最適化パスの追加が容易化

(2) レジスタ割り当て機能☆4

(3) 再構成のスケジューリング機能の実現

☆1 配置配線が失敗した場合 (3) 組み立てからやり直す.

☆2 デフォルトでは PE 内を通過できるようにエッジを張り,PE が配置されるとそのエッジを一時的に取り外す.

☆3 即ち他のカスタム命令のための prcf 命令と干渉しない限り.

☆4 Redefis ツール・チェインでは ISAGen ツールとリターゲッタブル・コンパイラは別のツールであり協調してレジスタ割り当てられず, レジスタは使い捨てられていた.

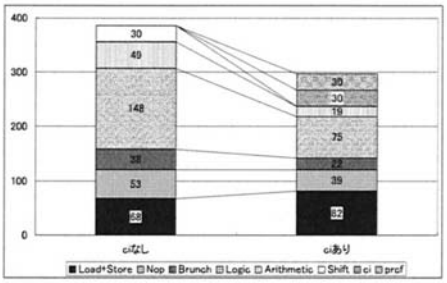

図 6 ステップ数の比較（例：CRC32）

(4) 配置配線のリトライ機能の実現

(5) ターゲット・アーキテクチャ変更コストの低減☆5

(6) 保守性・堅牢性・信頼性の向上

(7) デバッグ情報の出力機能の実現☆6

5. 実　　験

現在,FPGA 版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）とカスタム命令の再構成時間, 演算時間の比がまだ明らかになっていない. よって残念ながらスケジューリングによる再構成時間隠蔽の効果やカスタム命令化の効果を見積もることは出来ていないため, 図 6 では CRC32 の例についてどのくらいの命令がカスタム命令化されるかをカスタム命令化した場合としていない場合の実行ステップ数の違いとして比較したものである. カスタム命令化によって全体としてステップ数は 80 命令（22%）減少している. 命令種別毎に見るとシフト演算の全てと論理演算, 算術演算の約半分がカスタム命令化されていることがわかる. 残念ながらこの例ではループの中に 2 つのカスタム命令が含まれ, 交互に再構成しているため prcf 命令をループの外に出すことはできていない. しかし, アセンブリ・コードを確認した結果スケジューリングによって見かけ上 3 命令分だけ再構成時間が短縮されている.

表 2 は CRC32 の例で配線に成功した 2 つの命令の構成情報と別のサンプルの命令で配線に失敗した構成情報における PE 利用率である. PE による転送は一定の効果を上げて配線リソースの不足をある程度は補っているが, 現在の RDP の構成では転送で LVL バスも消費してしまうため遠くの PE に信号を送るには

☆5 COINS の低レベル内部表現, バックエンド生成ツール採用, また配置配線アルゴリズムをアーキテクチャ独立なアルゴリズムに変更したことでアーキテクチャの変更が容易になった.

☆6 コンパイラとして統合されたためデバッグのための情報を統一的に生成しやすくなった.

表 2 PE 利用率

命令	結果	遊休数 (%)	利用数 (%)	純利用数 (%)	転送数 (%)	
ISA13	成功	80 (60%)	52(39%)	40(30%)	12(9%)	
ISA14	成功	84 (64%)	48(36%)	47(36%)	1(1%)	
ISA4	失敗	38 (29%)	94(71%)	77(58%)	17(13%)	LHL15 本不足

向かず効果は限定的である. 成功した例では利用率が30〜40%台でまだあまり高いとはいえない. 一方失敗した例では 70%の利用率であるが, LHL バス方向に十数本ラインが足りず配置配線に失敗している.

6. まとめと今後の課題

本発表では SoC の計算エンジンとなる動的再構成可能な ASIP を動的再構成可能プロセッサを用いて実現するプラットフォームとして以下を紹介した：

Vulcan2 LUT ベースの PE をバス接続したリコンフィギュラブル・データ・パス (RDP) という命令単位で動的再構成可能な演算装置を備えたプロセッサ.

ISAcc Vulcan2 プロセッサのためにカスタム命令の構成情報とカスタム命令を利用したオブジェクトコードの両方を C で記述されたソース・コードから生成するコンパイラ.

この両者について実験を行ったが, まだ簡単なサンプルしか試していないので, 今後は複数のより実用的なアプリケーション, 例えば各種暗号アルゴリズムや画像処理アルゴリズム, 特に[4] であまりよい結果を残せなかった事例について改善できているのかどうかを確認する必要がある. これには現在設計中の Vulcan2 の FPGA 版の設計を急ぐ必要がある.

実験では ISAcc についてカスタム命令の生成には命令数の削減と, スケジューリングについていくらか効果が見られた. しかし, ループの最適化としてループ外へ prcf 命令を置くには RDP 部で複数の構成情報をキャッシュし, 短時間でコンテキスト切り替えできる仕組みを考慮する必要がある. 一方 RDP 部については一様で細粒度の RDP にも関わらず, まだ PE の利用率は十分でない. ただ失敗している例でのリソースの不足はわずかであり,PE とバスの数, 段数と一段あたりの PE 数など, 実験によってよりよい RDP を求める余地がある.

謝辞 本研究の一部は, 平成 14–18 年度文部科学省知的クラスター創成事業「次世代システム LSI アーキテクチャの開発」に拠る.

参 考 文 献

1) ALTERA 社. http://www.altera.com/.
2) COINS 協会. http://www.coins-project.org/.
3) Victor GOULART, Lovic GAUTHIER, Takayuki KANDO, Takuma MATSUO, Toshihiko HASHINAGA, and Kazuaki MURAKAMI. "redefis - a system with a redefinable instruction set processor". In *SBCCI'06*. ACM, 2006.
4) 橋永寿彦, Lovic Gauthier, 神戸隆行, Victor Goulart, 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 CPSY2005-53〜63, pp. 7–11, 2005.
5) IPFLEX 社. http://www.ipflex.com/.
6) 松尾拓真, 首藤真, 橋永寿彦, 森江善之, Lovic Gauthier, 村上和彰. "redefis 開発環境の概要〜開発環境の概要とアプリケーション開発例〜". 電子情報通信学会技術研究報 CPSY2004-25〜31, pp. 19–24, 2004.
7) 松尾拓真, 首藤真, 橋永寿彦, 森江善之, Lovic Gauthier, 村上和彰. "vulcan〜redefis の一実施例とそれへのユーザ機能実施例の紹介". 電子情報通信学会技術研究報 CPSY2004-25〜31, pp. 13–18, 2004.
8) 松尾拓真, 首藤真, 橋永寿彦, 森江善之, Lovic Gauthier, 村上和彰. " 命令セット再定義可能プロセッサ「redefis」〜ユーザ機能の実装に適した soc プラットフォームの提案〜". 電子情報通信学会技術研究報 CPSY2004-25〜31, pp. 7–12, 2004.
9) L. McMurchie and C. Ebeling. "pathfinder: A negotiation-based performance-driven router for fpga". In *International Symposium on Field Programmable Gate Arrays*, pp. 111–117. ACM/SIGDA, 1995.
10) NECElectronics 社. Drp. http://www.necel.com/drp/.
11) PACT 社. Xpp. http://www.pactcorp.com/.
12) A. Sharma, S. Hauck, and C. Ebeling. "architecture-adaptive routability-driven placement for fpgas". In *International Conference on Field Programmable Logic and Applications*, pp. 427–432. IEEE, 2005.
13) 末吉敏則, 天野秀晴（編）. リコンフィギュラブルシステム. オーム社, 東京, 2005.
14) Tensilica 社. Xtensa configurable processors. http://www.tensilica.com/.