3. 実例 4

動的再構成プロセッサ（DRP）

本村 真人[*1] 藤井 太郎[*2]
古田 浩一朗[*3] 安生 健一朗[*4]
矢部 義一[*5] 戸川 勝巳[*6]
山田 順也[*7] 伊澤 義貴[*8]
佐々木 僚子[*9]

NEC エレクトロニクス 通信システム事業部

[*1] masato.motomura@necel.com
[*2] taro.fujii@necel.com
[*3] k.furuta@necel.com
[*4] k.anjo@necel.com
[*5] yoshikazu.yabe@necel.com
[*6] k.togawa@necel.com
[*7] j.yamada@necel.com
[*8] yoshi.izawa@necel.com
[*9] r.sasaki@necel.com

コンフィギュラブル・プロセッサとリコンフィギュラブル・ロジック

近年，プロセッサアーキテクチャの世界では，コンフィギュラブル・プロセッサが1つのキーワードになっている感がある[1]．詳細に見ていくとさまざまなアプローチがあるが，広い意味で捉えると，コンフィギュラブル・プロセッサとは，アプリケーションに合わせてプロセッサのマイクロアーキテクチャに追加・変更を加えることが可能なプロセッサである．多くの場合，この追加・変更は，処理を加速することを目的としてプロセッサのデータパスに専用ハードウェアを追加し，この専用ハードウェアを制御するための拡張命令を定義することで行われる（この目的であらかじめ命令セット内に命令拡張可能な仕組みを組み込んでいる）．コンフィギュラブル・プロセッサの成功の鍵は，(1) 専用ハードウェアによりどの程度の性能加速が得られるか，(2) ソフトウェア開発の際に専用ハードウェアをどの程度意識する必要があるか，の2点である．当然ながら，専用ハードウェアを作りこみ，これに併せて何がしかのソフトウェアチューンを行うからには少なくとも数倍程度以上の性能向上を期待する場合が大半であり，プロセッサの一部を改変するのみでこれを実現できるかどうかが成否の分かれ道となる．

一方，これと似て非なるものとして，FPGA に代表されるリコンフィギュラブル・ロジックがある．リコンフィギュラブル・ロジックは，ゲートレベルのロジック構成を出荷後に変えることができるデバイスであり，従来は，主として本格的な LSI 開発の前段階のプロトタイプ用途に用いられてきた．しかしながら，ユーザの手元で任意にハードウェアをプログラムできるという特徴を活かして，リコンフィギュラブル・ロジック上にユーザ所望のアプリケーション処理系を構築する，いわゆる「リコンフィギュラブル・コンピューティング」の試みが1990年代より活発化してきている[2]．リコンフィギュラブル・ロジックに収容可能なハードウェアの規模が数百万ゲートに達し大規模な処理系を実装可能になったことや，CPU マクロや乗算器アレイを埋め込んだ FPGA が一般的になってきたこともあり，今後，信号処理などを中心に適用事例が増えていくものと考えられている[3]．「リコンフィギュラブル・コンピューティング」の狙いは，処理をハードウェア化することにより，アプリケーションに内在する並列性を最大限に引き出し，プロセッサ処理とは桁違いの性能を実現することである．一方，その本格利用のためには，ソフトウェアとして書かれた処理をハードウェアとして合成する設計ツール（従来のコンパイラに相当）が重要となる．プロセッサ向けコンパイラと比べると，LSI 設計 CAD 技術をベースとした複雑な設計ツールとなるため，そのハードルの高さが「リコンフィギュラブル・コンピューティング」実現の1つの障壁になってきているといえる．

コンフィギュラブル・プロセッサは，プロセッサベースのソフトウェアシステムの中にハードウェアを取り込んでいくアプローチであり，逆に，リコンフィギュラブル・ロジックは，ハードウェアの方からソフトウェアと

の敷居を下げていく手段を提供している，といえる．アプローチは正反対であるが，いずれも，ハードウェアとソフトウェアの境界が次第にぼやけ始めているという現在の技術状況を示しているといえよう．

すでに活発に開発が行われているこれら両技術に対し，本稿で紹介するのはダイナミック・リコンフィギュラブル・プロセッサ（Dynamically Reconfigurable Processor: DRP），すなわち，動的に再構成可能なプロセッサである[4),5)]．その技術的な特徴は以下の3点である．

(1) 均一の小規模プロセッサをアレイ上に並べた並列プロセッサ構造
(2) 個々のプロセッサに単位演算をマッピングし，それらを空間的に連結することで処理を実現するハードウェア型のプログラミングモデル
(3) 個々のプロセッサの命令切り替え能力を，ハードウェア構造を動的に切り替えることに活用したダイナミック・リコンフィギュレーション型の実行モデル

コンフィギュラブル・プロセッサとの比較で言うならば，個々のプロセッサに固定的な専用ハードウェアを付加するのではなく，小規模プロセッサのアレイ上に空間的に処理をマッピングすることによりハードウェアをプログラムすることが特徴といえる．一方，リコンフィギュラブル・ロジックと対比すると，ゲートレベルのプログラマビリティを有するロジックエレメントをアレイ状に並べるのではなく，プロセッサを構成要素として用い，かつその命令切り替え能力をハードウェア構造の切り替えというかたちで活かすことで，「ダイナミック・リコンフィギュレーション」を実現している点が異なる．

DRPの位置づけと設計思想

図-1 に表されているように，DRPはシステムLSIへ搭載するIPコアである．CPUコアをリプレースするものではなく，ハードウェア的な処理方式の特徴を活かして，CPUが苦手な処理をプログラマブルに加速する補完的な位置づけを持つ．

前章で説明したように，DRPは一見並列プロセッサアレイのように見えるが，その実行方式は従来の並列プロセッサのそれとは大きく異なる．従来の並列プロセッサの場合，一般的には，アプリケーションを並列タスクあるいは並列スレッドに分割した上で，タスク／スレッドを個々のプロセッサの命令シーケンスとしてコンパイルする．これら命令シーケンスが各プロセッサ上で個々に実行され，必要に応じて相互に同期・通信を行いながら全体処理を遂行することが基本である．

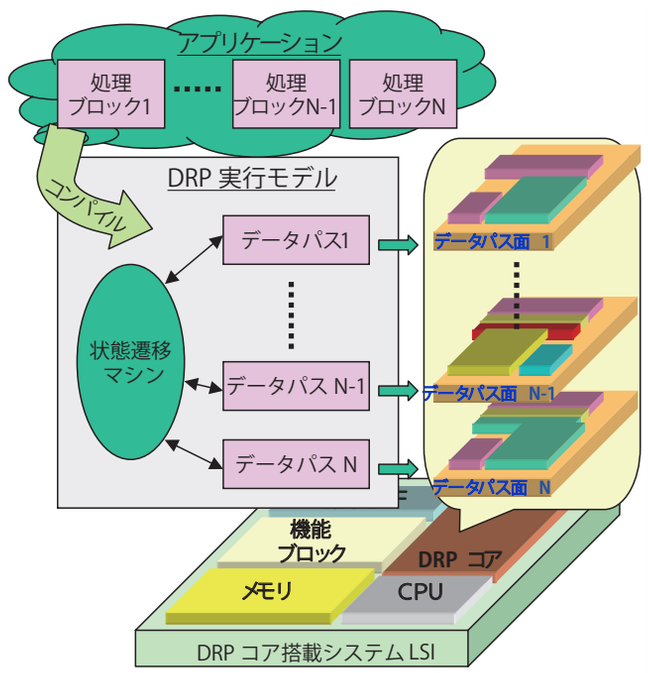

図-1 DRPの設計思想

これに対し，DRPの実行モデルでは，アプリケーションをまず時間的に大まかに複数の処理ブロックに分割し，(1) 分割された処理ブロック間の制御フローを規定する状態遷移マシンと，(2) 各処理ブロックに対応するデータパスというかたちでコンパイルすることを前提としている（図-1）．その上で，各データパスは，プロセッサのアレイに空間的にマッピングされ，各状態に対応する「データパス面」として実装される．DRP上で処理を実行する際には，状態遷移マシンに従って，状態が変わるたびに「データパス面」が切り替わることで，その時々に最適なデータパス構成で全体処理が実行されることになる．

この特徴的な実行モデルは，高位プログラム記述（具体的にはC言語）からハードウェア構成への高位合成技術を裏づけとしたコンパイラにより支えられている．DRP向けコンパイラ技術に関しては文献6）にその説明を譲り，本稿ではアーキテクチャを中心にさらに詳細に解説する．

アーキテクチャ

図-2 はDRPの基本構成要素であるプロセッサエレメント（PE）の構成を示したものである．PE内には

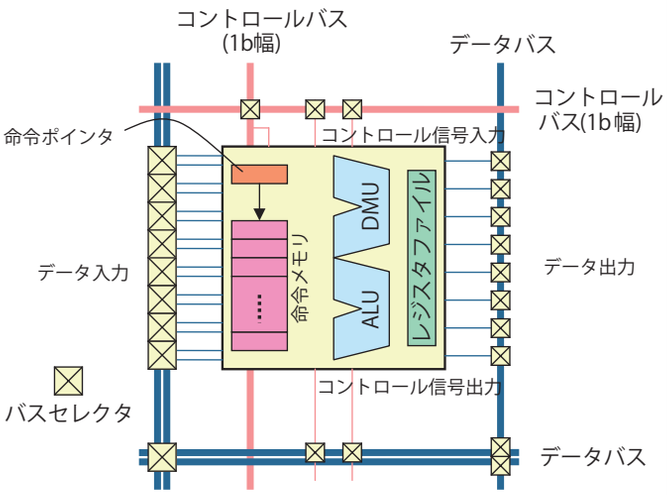

図-2　プロセッサエレメント（PE）

図-3　DRPタイル

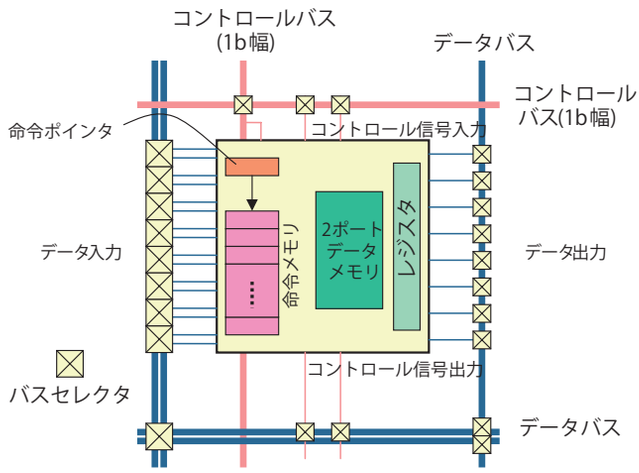

図-4　メモリエレメント（Mem）

演算ユニットである ALU と DMU（Data Management Unit），レジスタファイル，バスセレクタ，命令メモリが含まれている．ALU は，通常の加算・減算・シフト等の基本演算に対応した演算器であり，DMU はこれらの基本演算命令に加えて，ビットマスク・ビットセレクトなどのビット処理系の演算命令に対応している．バスセレクタは，PE 内の演算ユニットやレジスタファイルと PE の周りを縦横に走るデータバスとの入出力接続を指定するものである．

命令メモリには PE が実行する命令コードを複数個蓄えておく．それぞれの命令コードは，ALU/DMU が実行する演算を示すオペレーションコードのほかにバスセレクタの制御コードも含んでおり，レジスタファイルをバイパスして他の PE にフロースルーでデータを渡すなど，複数の PE を連結して柔軟なデータパスを構築できるようになっている．

通常のプロセッサでは，あるサイクルの実行結果をレジスタに蓄え，そのレジスタを参照しながら以降のサイクルの演算を行っていくことを基本としている．これに対し，DRP の PE では，複数の PE でコレクティブに演算を行うことでデータパスを構築することをアーキテクチャの基本としているため，このような PE 間で柔軟にデータを受け渡すための仕組みが非常に重要となる．

図 -3 は，PE を 8 × 8 のアレイ状に並べて構成した DRP タイルを示している．DRP タイルは，DRP を動作させるために必要な最小構成単位であり，これを繰り返し並べていくことで，より大きな DRP コアを構成することができる．PE アレイの左右にはメモリエレメント（Mem）が配置されている．また，DRP タイルの中心には状態遷移コントローラ（STC）というブロックが配置されている．

図 -4 は，Mem の構成を示したものである．PE と同様に命令メモリとレジスタを持っており，縦横のデータバスにスイッチを経由して接続されている．ただし，PE における ALU/DMU の 2 つの演算器の代わりに 2 ポートのデータメモリを有しており PE アレイ上に構築されるデータパスにバッファリング機能を提供している．PE の場合と同様，命令メモリ内に蓄えられた命令コードの指示により，2 ポートメモリのリードポートやライトポート，あるいはそれぞれのアドレス入力ポートとデータバス間の接続が規定される．ある Mem に対しては，データバス接続の設定により，タイル内の任意の位置の PE からアクセスが可能である．

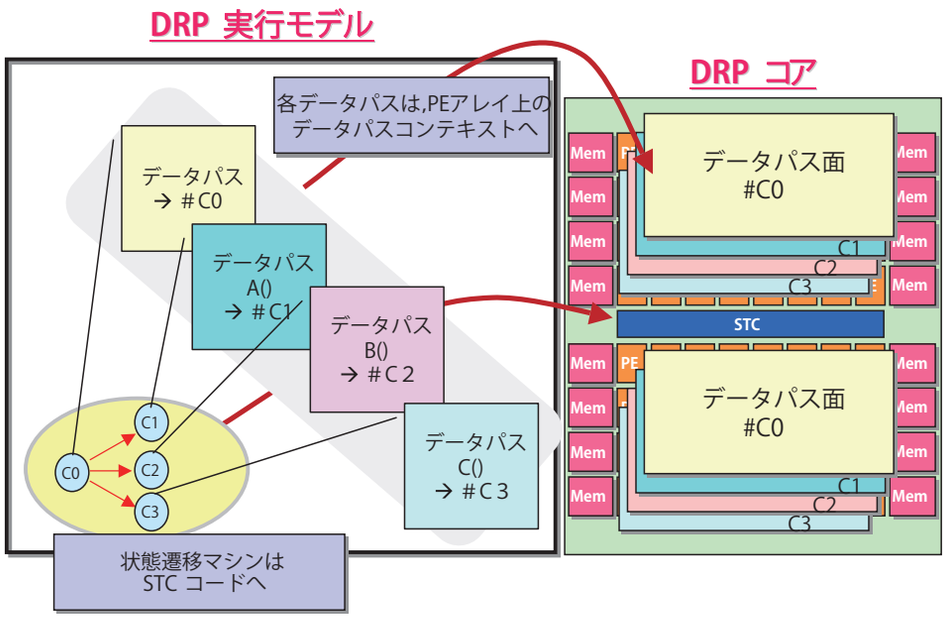

図-5　DRPアーキテクチャへのマッピング

図 -1 において，状態遷移マシンと各状態に付随するデータパス面という形が DRP の実行モデルの基本であることを説明した．**図 -5** では，この状態遷移マシンが STC に，データパス面が PE と Mem のアレイ上にそれぞれマッピングされることを説明している．STC の基本的な働きは DRP タイル内に命令ポインタを発行することである．タイル内の PE ならびに Mem は，発行された命令ポインタに従って，各自の命令メモリに蓄えられた複数の命令コードの中から所望の命令コードを選択する．STC 内には状態遷移マシンをトレースするシーケンサが存在し，状態が切り替わるごとに異なる命令ポインタをタイル内に発行する仕組みとなっている．

命令ポインタの値を切り替えると，タイル内のすべての演算器（ALU，DMU）が実行する命令，およびこれら演算器と 2 ポートデータメモリの相互接続関係が切り替わることになる．これは，あたかも，タイル内にあらかじめプログラム・カスタマイズされたデータパス面が複数個存在し，状態が遷移するたびに，データパス面が切り替わることを意味している．なお STC がトレースする状態遷移マシンにおいて状態分岐が存在する場合，その判断材料が必要となるが，これは PE アレイから STC へイベント信号を送ることによって実現されている．

IP コアとしてシステム LSI に集積される DRP コアは，このようなタイルを複数個並べて構成される．1 つのタイルごとに STC とアレイ部が組み合わされているため，各タイルが独立して別個の状態遷移マシンの制御下で動作することが可能となっている．またタイル同士を連携動作させることにより，全体が 1 つの状態遷移マシンの下で一括動作することも可能となっており，スケーラブルなアレイ構成を実現している．

アプリケーション

DRP アーキテクチャの実証とアプリケーション開発評価を目的に，2002 年に DRP-1 チップを設計試作した（**図 -6**）．DRP-1 は，8 個のタイルから構成される DRP コア（512PE），外部からの制御インタフェースである PCI，外部メモリを接続可能なメモリコントローラなどから構成される．本プロトタイプチップは，0.15 μmCMOS8 層メタルプロセスを用い，チップ上に 22M トランジスタを集積している．また，各 PE 内の命令メモリは 16 面分のデータパス面をオンチップに保持できる構成となっている．この DRP-1 チップを搭載した開発ボードと PC 上で走行する DRP 開発ツール（コンパイラ，デバッガ，統合開発環境）により，DRP アプリケーションを開発する環境を準備し，運用している（図 -6）．

■ DRP のアプリケーション特性

一般に，ストリームデータに対してその加工・編集・変換等の作業を行ういわゆるストリーム処理は，CPU が苦手な処理であることが知られている．これは，ロー

図-6　DRPアプリケーション開発環境

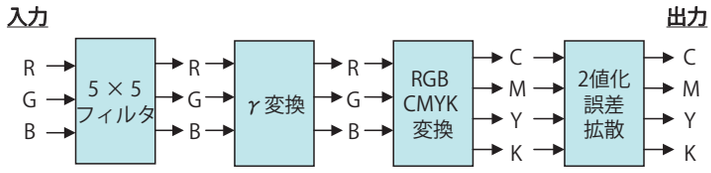

図-7　静止画・画像フィルタ・変換アルゴリズム

ド―ストア型のアーキテクチャでは頻繁にメモリアクセスとこれに伴うキャッシュミスが発生し，CPU の高速演算性能を活かすことが難しいからである．これに対し，DRP では，入力されたデータストリームに対し，そのデータストリームに応じたデータパスを構築してパイプライン型・データ並列型などの並列性を活かした最適な処理を行うことができる．CPU のようにメモリアクセスの間処理をストールするのではなく，データストリームの入力，出力，加工，一時記憶など，すべて並列に行うことにより，低いクロック周波数でも高性能な処理が可能となる．

このようなストリーム処理の例として，ここでは簡単な静止画の画像フィルタ・変換アルゴリズムの例を紹介する（**図 -7**）．入力された RGB 形式（24b）の画像ストリームに対して，5 × 5 フィルタ，γ 変換，CMYK 変換，2 値化・誤差拡散，などの処理を行い，CMYK 形式（32b）の画像ストリームを出力するものである．

表 -1 は，本画像処理アルゴリズムを CPU，DRP-1，

	処理サイクル数（Cycle/ 画素）	周波数（MHz）	A4 300dpi 処理時間（秒）	速度性能比
CPU	1185	500	19.5	0.0054
DRP-1	1	80	0.11	1
ハードワイヤド・ロジック	1	200	0.042	2.5

注釈：
- CPU：市販の 500MHz, 4 並列スーパースカラプロセッサ (in-order)．コンパイラによる最適化のみ実施（ソース記述はアルゴリズムに忠実）
- ハードワイヤド・ロジック：RTL 記述を論理合成で速度最適化．RTL には DRP コンパイラの中間出力を使用．DRP-1 と同一製造プロセス (0.15μm) を仮定

表-1　性能評価例

ハードワイヤド・ロジックでそれぞれ処理したときの性能を比較したものである☆1．CPU ではメモリに蓄えられた画像データを個々にアクセスしながら処理を行うため，1 画素あたりの処理に 1,000 サイクル以上を要している．一方，DRP-1 では，本画像処理アルゴリズムを 1 画素あたり 1 サイクルで処理できるパイプライン型のデータパスを構築して処理を行っている．このため，周波数では CPU の方が数倍早いものの，処理性能比では，DRP-1 の方が 200 倍高性能という結果が得られている．

一方，同一の画像処理アルゴリズムを，DRP にマッピングするのと同一の処理アーキテクチャに基づいて単純にハードワイヤド・ロジック化した場合とを比べると，処理性能比では，ハードワイヤド・ロジックが DRP よりも約 2.5 倍勝っている．これは，ハードワイヤド・ロジックにおいては論理ゲートのチェーンで構成される動作周波数上のクリティカルパスが，DRP 上ではデータパスを介した複数個の PE のチェーンで実現され，DRP の動作クロックがハードワイヤド・ロジックの 2.5 分の 1 となるためである．アルゴリズムをデータパス化する際のアーキテクチャが同一であるため，この動作クロックの違いがそのまま処理性能の違いとなっている．

ハードワイヤド・ロジックは，あくまでも LSI に焼き付けられたハードウェアであり，後から変更することはできない．この 2.5 倍という性能差は，DRP を用いてプログラマブルにハードウェアを構成することに払った代償である，といえる．なお，当然ではあるが，この動作クロックの差の絶対値はアプリケーションにより変動する．たとえば，より深いパイプライン化を推し進めることができる場合は，DRP においてハードワイヤド・ロジックと同等の性能を実現することも可能である．また，DRP アレイ構造のチューンアップや DRP コンパイラの性能向上とともに，この動作クロックの差自体も次第に減少してきている．

☆1　2002 年段階の評価である．評価条件に関しては表 -1 の注釈を参照．

ダイナミックリコンフィギュレーションのメリット

DRP は，プロセッサアレイを基本アーキテクチャとして採用することで，プロセッサの命令切り替え能力をデータパスの動的な切り替え能力として活用している．このようなダイナミックリコンフィギュレーションの魅力とメリットは，いくつかの側面で語ることができる．

(1) Data-Resident Computing

DRP では，レジスタファイルやメモリに処理途中の中間データを保持したまま，データパス面を切り替えることが可能である．すなわち，従来型の並列計算モデルのように物理的に異なるハードウェアモジュール間でデータを受け渡すのではなく，データを保持したまま逆にハードウェアモジュール（すなわちデータパス面）を入れ替えることで，異なる処理間のデータ受け渡しが可能である．従来の並列計算モデルでは，データ転送バンド幅が全体処理のボトルネックになる場合が多かったが，この Data-Resident 型の処理方式ではデータ転送自体を不要とすることで，データ転送ボトルネックの解消を図ることができる（逆にデータパス構成情報の大量転送が必要となるが，DRP ではこれを PE 内に命令メモリを置くことで実現している）．この際，たとえばデータの配置に応じてその並列アクセスに最適なようにデータパス面を構築する，などの工夫も必要となってくる．

(2) ハードウェア実装効率の向上

データパス面を切り替えるということは，すなわち，処理ブロック単位で時間的に逐次化しているということである（図 -1）．逐次化と言えば性能面からはマイナスであるように思えるが，LSI への実装時のシリコン面積を一定とした条件の下ではそうとも言えない．決められた面積内にデータパスを押し込む必要があり，利用できる並列性には自ずと限界があるからである．DRP では，アーキテクチャにあらかじめ逐次性を組み込んでおくことにより，むしろ，同一シリコン面積により多くのハードウェアを集積可能である，というメリットを実現している．

(3) ハードウェアのソフト化・仮想化

PE 内の命令メモリと外部メモリ間でデータパス面の構成情報を入れ替えることにより，オンチップでは蓄えきれないさらに大きなハードウェアを DRP コア上に仮想的に実装することが可能となる．これは，回路量の制約から設計者を開放したスケーラブルなハードウェアの実現を意味する．ソフトウェアとハードウェアの決定的な違いは，実は，ソフトウェアはどれだけ巨大でも，時間さえかければ実行できることが保証されている点にある．この点こそ，逐次的に命令シーケンスをたどっていくプロセッサ型の実行モデルの強みにほかならない．ダイナミックリコンフィギュレーションとは，このようなソフトウェアの優れた能力をハードウェアの世界に持ち込むものであるといえる．

今後の展望

システム LSI の大規模化による開発期間の長期化，開発費の高騰とともに，従来ハードワイヤド・ロジックに頼らざるを得なかった処理をプログラマブルに実現する IP コアへのニーズは急速に高まりつつある．

(1) 1 つのシステム LSI をたくさんのシステムに使いまわしたい．そのためには，処理内容をハードワイヤド・ロジックに固めずに，なるべくプログラマビリティを保ちたい

(2) 開発期間の長期化により，システム LSI が完成したときには，すでに処理内容が陳腐化している危険性が増している．LSI の完成時点で（あるいは完成後も）処理内容を定義できる仕組みが欲しい

(3) 製造工程と回路設計の双方が複雑化することにより，LSI 開発時に起こる製造起因の問題と設計起因の問題の切り分けが加速度的に難しくなりつつある．規則的なアレイ構造を持つコアで製造起因の問題解析を簡素化するとともに，製造後にプログラムするコアにより，製造起因と設計起因の問題とを完全に分離できるようにしたい．

DRP は，まさしくこのようなニーズに応える IP コアであるといえる．

具体的な応用システムという視点では，特に，動画・静止画の変換・加工処理を行う画像処理機器や各種パケットのフィルタリング・加工・転送処理を行うネットワーク機器（サービスルータ，ネットワークサーバ，etc.）などで，このような IP コアへのニーズが高い．上記（1），(2) の理由に加え，これらの機器ではストリーム型処理が中心のため，CPU ベースの処理系では要求スループットを満たせない場合が多いからである．DRP-1 プロトタイプチップを用いたベンチマーキング・利用形態の探索を通じて見えてきたさらに具体的な応用システム形態を踏まえ，現在，DRP 搭載 LSI の製品開発を推進中である．

謝辞　DRP の開発に多大なるご協力をいただいている NEC エレクトロニクスの諸氏，ならびに DRP コンパイラの開発を推進いただいている NEC システムデバイス研究所の諸氏に感謝いたします．

参考文献

1）マイクロプロセッサが SoC を強くする，日経マイクロデバイス，No.230, pp.31-45，日経 BP (Aug. 2004).
2）末吉，天野：リコンフィギュラブルシステム，オーム社 (2005).
3）Virtex-4 特集 , Xcell Journal, No. 52, Xilinx (Apr. 2005).
4）Motomura, M. : A Dynamically Reconfigurable Processor Architecture, in Proceedings of Microprocessor Forum (Oct. 2002).
5）本村，若林，粟島，戸井：DRP のデバイス・アーキテクチャ，Design Wave Magazine，2004 年 8 月号，pp.62-68，CQ 出版（2004）.
6）若林，粟島，戸井，本村：DRP の開発環境，Design Wave Magazine，2004 年 8 月号，pp.69-78，CQ 出版 (2004).

（平成 17 年 9 月 8 日受付）