多機能化，高速化が進むネットワーク・プロセッサ

C-Port Corporation（*A Motorola Company*）
Andrew Funk

本稿では，近年のネットワーク装置に導入が進むネットワーク・プロセッサ（*NPU: Network Processing Unit*）の役割，および，システム・デザインに*NPU*を適用する際に重要なキー・ポイントについて述べる．ここではモトローラの第二世代*NPU*である*C-5e™ NP*を具体例として取り上げ，その特徴と機能について詳しく概説する．

次世代ネットワークにおけるC-5e™ NPの役割

次世代ネットワークは，現状の階層的なインフラストラクチャの末端にデータ装置や音声装置（デスクトップコンピュータや電話機）が接続されたインターネットの構成とは大きく異なるものになるであろう．次世代ネットワークでは，無線，CATV，光ファイバー，xDSL，通信衛星など，多様な接続形態によってインターネット資源にアクセスするさまざまな『情報端末』が主流となる．こうした端末やそれによって可能となるサービスが普及すれば，ネットワーク・システムの構築方法は劇的な変化を遂げると予想される．この革命的な変化はすでに始まりつつある．

現在のネットワーク装置は，次世代ネットワークの要求を満たすために高性能化の一途をたどっている．これらの装置は，ワイヤ・スピードでのスイッチングやルーティングを処理するだけではなく，数多くの新たなサービスに対応できるものでなければならない．QoS（Quality of Service）や効率的に管理されたマルチベンダ・ネットワークを実現するためのDiffServ，MPLS（Multi-Protocol Label Switching），Voice over IPサービス，IPv6サポート，SLA（Service Level Agreement）による課金／制御など，多くの新しい機能が求められている．同時に，必要とされるインタフェースの種類や速度も，1.5Mbpsから10Gbpsに至るまで増加の傾向を示している．

帯域幅だけでなく，性能やサービスも含めた要求の高まりは，通信分野の変化の速度をさらに加速することになる．その結果，Time-to-Marketに対する短期化の要求と共に製品のライフサイクルの短縮が引き起こされている．しかし一方では，システム・デザインの複雑化により製品の開発サイクルは長期化の傾向がある．ネットワーク・システムの開発者は，製品ラインと製品世代の間でのソフトウェア資産の継承性を犠牲にして，ハードウェア／ソフトウェアの機能分担を根本から検討し直す経験を，新たに製品を開発するたびに何度となく繰り返してきた（ライン・カードも例外ではない）．その結果が，Time-to-Marketの長期化，開発コストの高騰，製品ライフサイクルの短縮である．

従来にないタイプの半導体デバイスであるNPUは，速度性能と柔軟性のトレードオフの考え方に変化をもたらす．それは，ほぼすべての通信機能をハードウェアの速度性能低下を招かずにソフトウェアによってプログラミング可能とするからである．NPUは，最初のリリースから長期間が経過した製品でも高度な新しい機能をソフトウェアで追加できるため，カスタム・ハードウェアの開発につきもののリスクの高い長期の開発サイクルという問題を避け

られる．ネットワーク装置ベンダは，単に速度性能のみを重視した製品開発ではなく，顧客に高度なサービスを提供する目的で集中的に貴重な開発リソースを利用できるようになる．優れたNPUは，ネットワーク装置の設計過程を根本的に変革する『通信プラットフォーム』となり得る．その1例がモトローラのC-5e NPである．

C-5e NPは，ネットワーク装置ベンダが同一のハードウェアおよびソフトウェア・アーキテクチャに基づいた多様な製品を短期間で市場に供給するためのクラス随一の能力を備えている．モトローラのスマート・ネットワークス・プラットフォーム☆1の一部であるC-5e NPは，優れたネットワーク・プロセッサ・テクノロジに加えて，『標準』プログラミング・インタフェース，通信ソフトウェア・コンポーネント（モトローラおよび提携企業より提供），および統合的な開発環境を提供する．そのため，新製品のTime-to-Marketが大幅に短縮されるだけではなく，ソフトウェア・サービスを通してTime-in-Market☆2は劇的に延長される．

ネットワーク・プロセッサについて

ネットワーク・プロセッサは，ネットワークのトラフィックを処理する半導体製品で広く用いられる表現である．本稿では，設計者がソフトウェアの再プログラミングによって通信データの転送機能を変更できるデバイスを，『ネットワーク・プロセッサ（NPU: Network Processing Unit）』と定義する．ただし，この定義を行ってもNPUと見なされる製品は多岐にわたり，すべてを比較するのは困難である．用途に対するNPUの適合性を判断する場合は，以下に示すいくつかのキー・ポイントを考慮する必要がある．

- プログラミング可能な機能と特性
- プログラミングが必要な要素
- NPUの集積レベル
- さまざまなプロトコルや機能の実現要求に対する柔軟性
- NPUのプログラミング手順

多くのNPUには，パケットのクラシフィケーションに使用するフィールドや，ある特定タイプのパケットのクラシフィケーションに使用する検索アルゴリズムをソフトウェアによって制御できることなど，いくつかの共通する特性がある．しかし，機能の統合レベルやNPU向けソフトウェアを開発するための言語／開発環境など，その他の多くの特徴は千差万別である．設計者は，これらの相違を検証することで目的に応じたNPUを決定することができる．

十分なプログラマビリティ

すべてのNPUはプログラミングに対応しているが，一般的にはプログラミング可能な要素や機能は大きく異なる．NPUが真のプラットフォームとして利用されるためには，多様なインタフェース，プロトコル，製品種別等，あらゆる環境に適用できるものでなければならない．そのために必要なのは，レイヤ2からレイヤ7に至るプロトコル・スタックのすべてのレベルでのプログラマビリティである．

C-5e NPは，データフロー型VLIW（Very Long Instruction Word）プロセッサとRISCコアを併用することにより，この種の完全なプログラマビリティを提供している．C-5e NPの内部でのパケット／セル・データの処理は，データフロー・アーキテクチャに基づいて動作するVLIWマイクロ・エンジンにより行われる．データの各ビットは，これらのエンジンを通して転送され，データの抽出や操作に対して特に制限が設けられることはない．RISCコアはC言語でプログラミングされ，フォワーディングやその他のポリシー処理等に必要な情報へのアクセスを，パケット解析やデータ操作の負荷に影響を与えることなく実行することができる．

C-5e NP内のプログラミング可能な要素は，特定のプロトコルを目的としたものではなく，さまざまなプロトコルの処理または拡張を考慮して機能が単純化されている．こうしたプログラマビリティの汎用性により，多様なインタフェース間でのパケット／セル／データ・ストリームを取り扱うプロトコルのサポートが可能となる．その結果，新たに登場したマルチサービス・キャリア・ネットワークの基盤である，キャリアのエッジ装置等への適用が可能となる（**図-1**）．

機能別コプロセッサによる処理

NPUは，プロトコル・スタックの各階層で制限なくプログラミング可能であるという点が長所となる反面，検索アルゴリズム，メモリ管理，メッセージのキュー処理

☆1 モトローラがネットワーク・マーケットに対して提供する，半導体デバイス，開発環境，ソフトウェア等の要素技術をすべて統括したプラットフォームの総称．
☆2 市場での製品のライフサイクル．

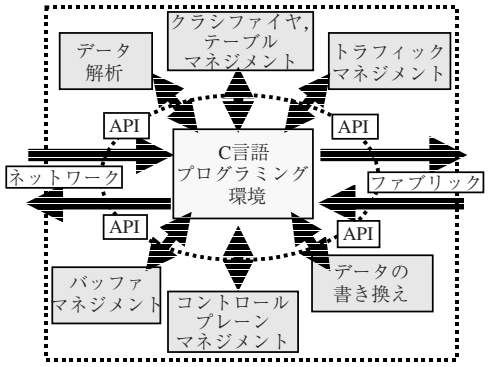

図-1　柔軟性を持ったプログラミング・モデル

などの基本的な操作までもプログラムしなければならない点が短所ともなり得る．NPUを評価する際には，ハードウェアにより実現することも可能なものの中で，本当にプログラムによる制御を可能にしなければならない基本操作はどれなのか，ということを念頭に入れることが重要である．ある帯域幅のデータを処理するために割り当てることが可能なプログラミング・サイクル数はNPUの重要な仕様の1つである．専用のハードウェアで処理を行う等，プログラミング以外の方法でより適切に実装可能な処理に対してもプログラミング・サイクルを割り当てなければならない種類のNPUは，本来プログラミング処理が求められる機能やアプリケーションに対して制限を加えることになってしまう．

C-5e NPは，すべての処理要素間で共用される，機能ごとに専用のコプロセッサをいくつか備えている．テーブル・ルックアップ，バッファ・マネジメント，キュー・マネジメントのすべての操作は，オンチップのハードウェア・ユニットである，テーブル・ルックアップ・ユニット（TLU），バッファ・マネジメント・ユニット（BMU），キュー・マネジメント・ユニット（QMU）により実行され，ハイレベルの操作インタフェースをプロセッサに提供している．RISCコアからテーブル・ルックアップなどの処理負荷を削減できれば，データ転送用のプログラミング・サイクルが固定的に決まることになり，より柔軟性が増してサービスの強化につながる．また，これらのコプロセッサを共用するという特性により，プログラミングの可能な1つの要素を常に決まったリソースに割り当てることなく，柔軟な利用が可能となる．

C-5e NPは，さらに細かくQoS制御を行う用途向けにC-5e NPのファミリー製品Q-5^{TM} TMC（Traffic Management Coprocessor）への外部インタフェースもサポートする．Q-5 TMCは，最大256Kの個別のキューと最大3レベルのスケジューリングを実行できるように内部のQMUの能力を拡張できる．Q-5 TMCは，ハイレベルにコンフィグレーション可能でマルチ・プロトコルに対応し，C-5e NPのアーキテクチャとプログラミング・モデルに完全に統合されている．

ハイレベルの機能統合

NPUは，システムの統合度を高めて部品点数を大幅に削減および簡略化を図りながら，同時にASSP（Application Specific Standard Products）等複数のコンポーネントの機能を設計に組み込んだ上で，性能を向上させる必要がある．NPUの統合度が高まれば，コンポーネント志向の設計によく見られる各デバイス間のインタコネクションがボトルネックとなることを避けられるといった恩恵も得られる．C-5e NPでは，内部の処理ユニットにより統合されたコプロセッサ・エンジンをインタコネクションによる性能低下の影響を受けずに最大限に活用することができる．さらに，下位レイヤの機能（Ethernetフレーマ，SONET/SDHフレーマなど）がチップに組み込まれており，従来に比べてポート密度の高集積化とコストの低減が可能である．

プロトコルの独立性

NPUの設計ポリシーは，少数の限定的なプロトコルを処理する方式とプロトコルに依存しない柔軟な方法でデータ転送を処理する方式に大別される．前者の例として，IPv4に特化したテーブル機能などプロトコル独自の処理機能を持ったり，最小・最大のパケット・サイズなどのプロトコル独自のパラメータに基づいて動作を制限するタイプのNPUがある．このようなタイプのNPUは，ターゲット・アプリケーションにおいてコスト効果の高いソリューションを提供することができる．しかし，新たなプロトコルの追加や追加したプロトコルと既存のプロトコルとの同時処理が求められる場合，プロトコルに依存する機能は既存のデザインで新しい機能を実現する際の障害となる．たとえ現在のソリューションでコスト効果が得られたとしても，急速に変化する要求を満たそうとすればその効果は瞬時に相殺されてしまうだろう．

C-5e NPは，後者の柔軟な方式に属し，プロトコルに依存する機能はごく限られている．その代わり，デバイス中のあらゆる個所においてハードウェアにより高速実行される基本機能が提供されており，それらはソフトウェアから制御可能となっている．検索テーブルはソフトウェアでプロトコルに依存したプログラミングをしない限りプロトコル依存とはならない．どのデータがソフトウェア上で重要

か，またはパケット／セル・ストリームの中でどこにそのデータが現れるかについては，ハードウェアによって制限されることはない．このような手法により，次世代ネットワークのプラットフォームに求められるプロトコル要求の急速な変化に柔軟に対応することができる．

単純化されたプログラミング・モデル

NPUを評価する上で最も重要と考えられるのは，プログラミング手法を理解することである．NPUがFPGA，ASIC，ASSPに勝る点を1つだけ挙げるとすると，それは『生産性』に関してであろう．NPUはシステム設計者の生産性をより高めることができるのである．

生産性を確実に向上させるため，NPUのプログラミング・モデルは，開発者が必要なときにすぐに使用できるものでなければならない．リアルタイム通信システムでこれまで最も広く利用されてきたソフトウェア言語は，数百万人の熟練したプログラマが存在し，膨大な量のコードが存在するCおよびC++である．CやC++のプログラミングでは，コードベースでの移植性が拡張され，将来の世代のネットワーク・プロセッサや業界標準プログラミング・インタフェースにも利用することができる．これに対して，専用言語やステート・マシン・コードにはこうした柔軟性はみられない．

モトローラでは，C-Ware™・ソフトウェア・ツールセット（CST）によって，C-5e NP上でC言語のコーディングをサポートするフル機能のプログラミング環境を提供している．CSTには，コンパイラ，デバッガ，パフォーマンス・アナライザ，統合シミュレーション環境など，ハードウェアの完成に至るまでの段階で起こり得る，多くの実装上の問題を解決するためのツールが用意されている．

スマート・ネットワークス・プラットフォームの主眼は，安定したプログラミング・インタフェースの提供である．プログラミング・インタフェースがプロセッサに依存している場合，通信プロセッサはソフトウェアの柔軟性や移植性を提供することはできない．プロセッサのアーキテクチャは，プログラミング作業を容易にし世代を超えたソフトウェアの再利用を可能にするために，汎用の通信アプリケーション・プログラミング・インタフェース（API）をサポートする必要がある．モトローラのCSTでは，データ・フロー上で実行される，上位互換性を持つC言語のAPIとしてこうしたインタフェースが提供されている．ネットワーク・プロセッサは，製品の世代を超えたソフトウェアの継承性を確保することで，ソフトウェアの開発サイクルと信頼性を本質的に向上させる．ソフトウェアの信頼性は，システム全体の可用性を左右する最大の要因である．

C-5e NP－ソフトウェア・オプティマイズド・ネットワーク・プロセッサ

C-5e NPは，17個のRISCプロセッサ・コアと4個の機能コプロセッサ，それらを相互に接続する3種類の内部バスから構成される．16個のチャネル・プロセッサ（CP）はクラスタと呼ばれる4つのグループに分類されて，高い処理能力とプログラミング可能な回線インタフェースを提供する．エグゼクティブ・プロセッサ（XP）は，CPと同じRISCコアを使用し，回線インタフェースの代わりに，PCIインタフェースを持つ．特定機能向けコプロセッサ・ユニットは，全プロセッサ間で共有され，各ユニット間のアクセスを保証する．こうした構成により，アプリケーションのパフォーマンス要求に基づいて，コプロセッサのリソースを各CP間で自由に割り当てることが可能となる．

C-5e NPの内部相互接続は，3つの内部バスで構成されている．その中で最も主要なバスであるペイロード・バスは，広帯域かつレイテンシが保証されたバスであり，DMAを介してBMU，QMUとCPの間でペイロードとキューイング情報を転送し，すべての内部プロセッサに対して均等にメモリアクセスが可能となるようにアービトレーションを行う．また，トランザクション向きのリング・バスも用意されており，プロセッサ間のメッセージ送信を実行するとともにTLUへのインタフェースとしても機能する．このバスは，スロット・インサーション・アーキテクチャの採用により，レイテンシが制御され，テーブル機能への高速なアクセスを実現する．3つ目のバスは，コンフィグレーションと制御システムコールに使用される，すべてのマップされたメモリにアクセス可能なバスで，グローバル・バスと呼ばれる．このバスは，全プロセッサとその構成ロジックに対して共有・共通のアクセス手段を提供する．他の多くのオンチップ・バスと同様に，これらの相互接続は実際の負荷に対して十分に余裕を持った性能を備えており，全帯域幅は60Gbps以上になる．この帯域幅により，通常の動作時においてこれらのバスがボトルネックになることはない．

以下の各項で，各機能ユニットについて詳しく説明する．**図-2**に，チップの概略のアーキテクチャを示す．

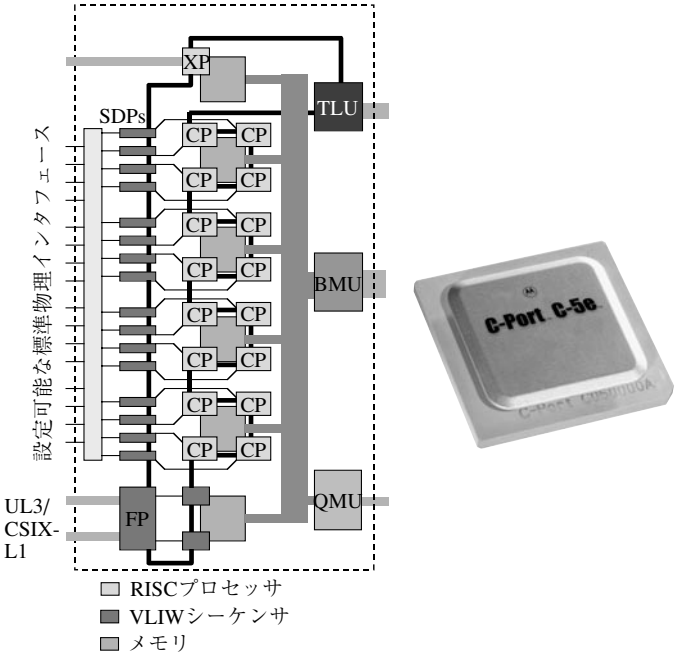

図-2　C-5e NPアーキテクチャ概略

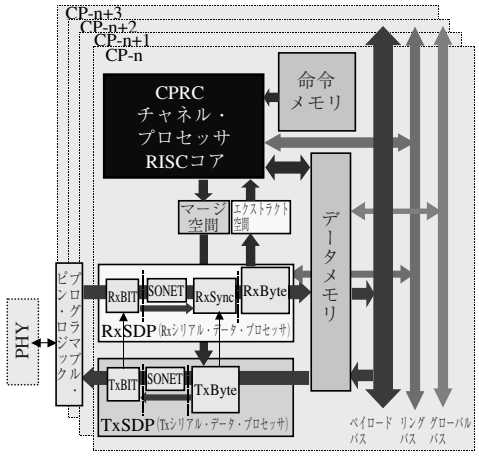

図-3　チャネル・プロセッサの構造

チャネル・プロセッサ（CP）

CPは，C-5e NPの中のプログラミング可能な要素である．一部のNPUの中には，専用の命令セット・アーキテクチャのみが用意されているが，C-5e NPの場合はプログラマブル・エンジンの処理を3つのユニットに振り分けることによって，パケット処理用に最適化された命令セットと理解しやすいプログラミングとの間のバランスを維持している．主要なプログラマブル・ユニットはチャネル・プロセッサ・RISC・コア（CPRC）である．このコアはC言語によってプログラミングが可能で，データを処理する際のCPの動作を制御する．その他，送受信用にそれぞれシリアル・データ・プロセッサ（SDP）が用意されている．これらのユニットはワイヤ・スピードでライン・データを処理するために組み込まれており，高級言語ライクなマイクロコードでプログラミングされる．**図-3**に，CPの構造を示す．

SDPは，マイクロ・シーケンサとして，データ転送パイプライン処理の中に組み込まれる．受信用SDP（RxSDP）にはプログラミング可能な3つのシーケンサが組み込まれており，それらは入力されるデータ・ストリームをパイプラインで処理する．送信用SDP（TxSDP）にはプログラミング可能な2つのシーケンサがあり，出力されるデータ・ストリームを処理する．データ・ストリームはすべてビット／バイト単位でアクセス可能で，上記の各シーケンサで順に処理される．データの抽出，廃棄，書き換え，およびデータ・ストリームへの挿入は，どのシーケンサでも可能である．また，シンボル・マッチング，チェックサムとCRCの生成，およびプロトコル・フィールドの識別等は，すべてこれらのプログラマブル・シーケンサが実行する．各SDPのシーケンサは，機能を限定して構成したロジック・ブロックによってスクランブルやエンコードなどの処理がサポートされる．

SDPは，Ethernet MACやSONET/SDHフレーマなどのリンク・レイヤのコントロール機能を実装する目的で用いられる．コントロール信号も含めた回線インタフェースはプログラミング可能であり，各種のインタフェースを構成でき，プロトコル解析はここで実現される．多くのテスト条件と条件分岐を使用することでステート・マシンを簡単に実装することが可能なので，対象フィールドの抽出およびテーブル・ルックアップは完全にプログラマブルに行われ，開発者が使用・修正不可能なデータフィールド等の制約はない．

CPRCはMIPS☆3ライクなアーキテクチャで構成されており，シンプルで標準的な命令セットを持ち，したがって一般のGNUソフトウェア開発ツール群を使用して簡単にプログラムすることができる．CPRCは4つの独立したレジスタ・セットを持っており，ステートの保存や復元を行うことなく，プロセッサ上で複数のスレッドを切り替えることができる．コンテキストの切替えに要するのはシステム・クロックのわずか数クロック分に過ぎない．CPRCは，ペイロードのDMA転送，割込み，例外処理，およびローカルのSDP処理を管理する制御用ロジックを持ち，また

☆3 米MIPS Technologies社のマイクロプロセッサ・アーキテクチャ．

グローバル・バスを介して他のリソースにアクセスする．CPRCの中核となるプログラミング・モデルはC-Wareアプリケーション・プログラミング・インタフェース（C-Ware API）で，このC-Ware APIにより，チップ内の他のコンポーネントに対するすべてのインタフェース機能が抽象化できる．これによりレジスタ・レベルの定義を考慮することなくコードが記述できるとともに，モトローラが今後も提供するC-Port™ネットワーク・プロセッサ・ファミリとの互換が可能になる．

CPのアーキテクチャに関して最も重要な点は，SDP-CPRC間で協調処理を行うメカニズムである．この処理は，『スコープ』と呼ばれる共有レジスタで行われる．各スコープは，セルやパケットを処理するために必要となるデータを保持する．CPRCはこれらのスコープをSDPと共有し，そのコヒーレンシはハードウェアで管理される．スコープは送受信用に各2個ずつ，計4つ用意されている．RxSDPは，要求されたデータの抽出，適切な検索を行い，スコープの使用権をCPRCに返すことで，CPRCが処理するためのスコープとなる．CPRCが受信スレッドの実行を開始するときには，処理に必要なすべてのデータは単純なデータ構造になっているので，Cプログラムでアクセスしやすい．送信側では，CPRCは同様のデータ構造に基づいてスコープを準備し，送信のためにスコープ全体をTxSDPに渡す．その後，CPRCは次の処理に移行する．

CPRCのプログラミング・モデルは，データ・スコープの抽象化等によって大幅に単純化される．つまり，パケットの内容に関する操作のほとんどがデータ転送作業で不要となり，関数の呼出しだけで送受信のデータ・ストリーム処理が実行される．

チャネルの帯域幅と機能性の拡張

すでに述べたように，4つのCPは『アグリゲーション』と呼ばれる機能を使ってグルーピングできる．こうした4つのCPはクラスタと呼ばれ，インストラクションおよびデータメモリを共有し，処理を緊密に同期させることが可能となる．1つのクラスタは最大で全二重1Gbpsの帯域幅を管理できる．CPが1つか4つかにかかわらずプロトコルや関数を実行するコードが同一に見えるこの同期機構により，複雑なプログラミングを要さずに帯域幅の拡大が容易となる．

SDPは内部ループ・バックにも対応しており，『リサーキュレーション』と呼ばれる機能を実装することができる．これは回線インタフェースを使用せずにCPもしくはCPクラスタを通してパケット・データを転送する処理であり，回線インタフェースだけでは実装できないプロトコルをパイプライン的に接続した複数のCPにパケット／セルを渡すことで実現することができる．さらに，必要なデータだけをリサーキュレーションするCP間で転送するので，ペイロード・バスの帯域幅が確保される．リサーキュレーションは，カプセル化されたプロトコルやその他のパイプライン化された処理に関して，深いレベルで効率的な実装をすることができる．これは機能性と帯域幅をトレードする，確実で簡単な方法である．

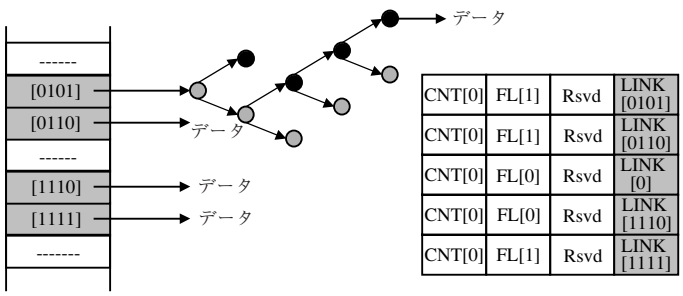

図-4　ツリーを指定したハッシュ・テーブル・データ構造

テーブル・ルックアップ・ユニット（TLU）

TLUは，リング・バスを介してコマンドを受け取ると，外部に接続された133MHzZBT SRAMをパイプライン的に使用しながら，実行ユニットでコマンドを処理し，その結果をリクエスト元のプロセッサに返す．TLUの処理は，IPv4のルーティング・テーブルなどのプロトコル特有のクラシフィケーションに関する機能をそのまま実装するのではなく，データ構造に対する基本的な処理で構成される．読出し，書込み，検索，読出し＋検索などの処理は，ハッシュ，ダイレクト・インデックス，バイナリ・ツリー，最長一致などのルックアップ・アルゴリズムと組み合わされて実行される．これらのテーブルは，すべて外部のZBT SRAMに格納される．

テーブルの大きさは，使用するルックアップ・アルゴリズム，キーのサイズ，各エントリに格納される関連データのサイズ，およびテーブルのエントリの総数で規定される．検索を継続する別のテーブルを示すエントリを使用し，テーブルを複数リンクすることで，さらに複雑なテーブルを作成することができる．この方法で，バイナリ・ツリーを指定して目的の検索を一意に解決することで，ハッシュ・テーブルにおける競合は解消される．また，任意の数のテーブルを連結できるために，設計者は複数の方法で問題の解決が可能となる．図-4に，このデータ構造の例を示す．

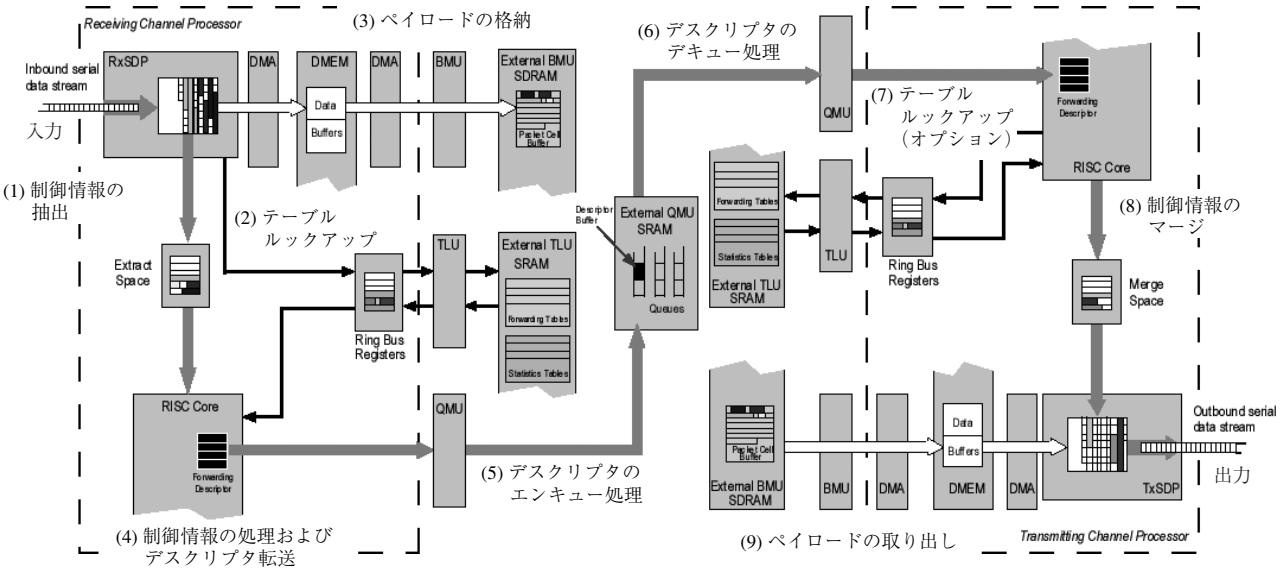

図-5　基本的なパケット・フロー

キュー・マネジメント・ユニット（QMU）

QMUは，CPRC間のメッセージ転送を処理するメッセージ・キューを管理する．このユニットの機能もC-Ware APIを通して抽象化されており，CPRCは別のポートへ転送すべき入力パケットについて，メッセージを別のCPRCにエンキューするための関数呼出しを行うだけですむ．プロセッサ間で転送されるメッセージはデスクリプタと呼ばれ，ソフトウェア上で定義されたデータ構造を持つ．QMUで扱うデスクリプタのフォーマットは，ユーザが自由に定義可能である．

QMUは，キューの制限とリソースの均衡化だけではなく，内部のキューのマルチキャスト機構もサポートする．QMUには，ポリシング，スケジューリング，シェーピングを行うためのハードウェアは存在しない．この機能は，CPのソフトウェアを通して，もしくは外部に接続されるコプロセッサによって実現される．直接接続可能なトラフィック・マネジメント・コプロセッサとして，モトローラはQ-5 TMCを用意しており，これによりシステム・デザインの包括的なQoS機能を提供することが可能である．

バッファ・マネジメント・ユニット（BMU）

BMUはペイロードの格納・取出しに用いられるすべてのメモリを管理するコプロセッサである．バッファ・メモリへのアクセスは，各CP（またはXP）のDMAエンジンを通して行われる．これらのDMAエンジンはプロセッサを介在させずに，バッファ・メモリとパケット／セルの格納・取出しを行うことが可能である．

BMUは，設定されたバッファ・プール群を通してバッファ・メモリを管理する．各バッファ・プールはチップ上のどのプロセッサにも，同サイズのバッファを複数割当てる．BMUはバッファの割当て／解除，および読出し／書込みを行うことを基本機能としている．CPはC-Ware APIを通じてBMUにアクセスする．

エグゼクティブ・プロセッサ（XP）

XPはCPと同様にRISCコアを持ったプロセッサであり，PCIを使ってホスト・プロセッサとC-5e NP間のすべてのやりとりを仲介する．ブートストラップ・ローダの実行はXPが行い，実行動作中のエージェントとしての動作や，CPの設定を管理／変更するためのコードの実行ができる．また，PCIインタフェースへのアクセスを監視し，ストリーミング転送のサポート，バス・マスタ動作のPCIインタフェースの提供を行う．

ファブリック・プロセッサ（FP）

FPは5Gbps以上の帯域幅を必要とするアプリケーションおよびシステムにC-5e NPを対応させるための高速インタフェースを備えている．このインタフェースは最大3.5Gbpsの帯域幅で動作し，セル・ヘッダのプログラミング処理を行うことで，バックプレーンやスイッチ・エレメ

ントを介した通信を行うことが可能となる．

基本的なパケット・フロー

C-5e NPを使ってパケット転送を行う場合の特別な制限事項は存在しない．データは，プログラム制御下にあるいずれかのバスを通じて転送することができる．ただし，多くのアプリケーションは**図-5**に示す基本的なパケット・データ・フローに準じている．

ネットワーク・プロセッサの進むべき方向

現在NPUは単純なライン・カードのフォワーディング・エンジンから複雑なゲートウェイ機能に至るまで，さまざまな用途に利用されている．NPUで提供されるインタフェース速度は，1.5Mbpsから2.4Gbpsに及び，これらすべてのアプリケーションに共通の方向性は，コスト効果の高い手法による市場への機能提供である．NPUの存在は，あらゆる場面で固定機能のASSPとASICソリューションとの組合せに対して競合を続けていくと予想される．設計者は単にNPU同士の比較だけではなく，既存デバイスとカスタム・ロジックの組合せで実現されるソリューションとNPUとの比較を評価していくことになるであろう．

NPUは，これからもさまざまな次元で改良が重ねられていくことだろう．現在多くのベンダが本格的に取り組み始めているのは，10Gbpsおよびそのすぐ後に登場すると予測される40Gbpsといった，次世代の帯域幅に対応するネットワーク・プロセッサの実現である．一般的に第一世代のNPUは，このような超高速ポートに対する性能要求に対応するためパイプライン構成を特化することを想定している．パイプライン構成は，クラシフィケーション，パケットの書き換え，およびトラフィック・マネジメントといった機能を分割して実現することになる．このようなNPUは，スイッチングや転送のプロトコルを装備することで，特定の用途に対しては十分な実力を発揮するであろう．

しかしながら，このようなテクノロジが幅広く採用されるに従い，さらに多くの機能がネットワーク・プロセッサに求められ始めている．ゲートウェイやコンテンツを認識した上でのスイッチング，アプリケーションの終端などに対応するため，NPUには単純なパイプライン構成ではなく，データ・フローでの真の意味でのアプリケーション処理の実現が求められる．この第二世代のネットワーク・プロセッサは，任意のプロトコル・レイヤでパケット・フローを監視・処理する能力を持ち，これまでに処理したパケットのコンテンツに基づいてフォワーディング・パスの動作を修正することが考えられる．1つの命令でより効率的に処理が実行されるようになると，これらのNPUでは，どれだけの帯域幅をNPUが処理できるかではなく，どれだけの機能を実装することができるかが重視されることになるだろう．

ま と め

C-5e NPは，ソフトウェアで最適化されるネットワーク・プロセッサの1つの例である．ソフトウェアによるデータ・フローの開発を可能にするために設計されたのがネットワーク・プロセッサである．しかしネットワーク・エレメントのフォワーディング・パスにおいて，設計者がデータ操作のコードを記述できるだけでは十分とはいえない．NPUは移植性，柔軟性，コード・サポートの容易性，およびデザイン全般の再利用という条件を満たしながら，より生産性が高められる方法で設計者がコード開発を行えるものでなければならない．それには，プログラミング可能なエレメントによってワイヤ・スピードのデータパスを実現するにとどまらない，より多くのものが必要である．完成度と使用効果に優れたソフトウェア開発環境，多くの各機能別アプリケーション・コードのライブラリ，他のコンポーネントとの協調動作，および適切に定義された不変のプログラミング・インタフェースも求められる．

プロセッサの仕様に従ってソフトウェアを構築するのではなく，むしろネットワーク装置の開発に関する課題を解決するためにソフトウェア指向で設計されたプロセッサ，それがC-5e NPである．これは，ハイレベルで単純なプログラミング・モデルに適した，プログラミング完全対応のハードウェア・アーキテクチャを採用することで実現されている．今後はC-5e NPのようなデバイスを用いることによって，より複雑な機能の開発期間がさらに短縮され，次世代ネットワークの製品展開が加速されると予想される．

（平成13年11月12日受付）