システムＬＳＩ設計技術　99－8
（2001．1．12）

動的機能変更可能な通信ネットワークノード

室岡 孝宏　宮崎 敏明
NTT 未来ねっと研究所
神奈川県横須賀市光の丘 1-1
電話：0468-59-3572
電子メール：murooka@exa.onlab.ntt.co.jp

あらまし　我々は、高速性が要求される通信処理をターゲットとした、動的機能変更が可能な通信ネットワークノード構成と、そのプロトタイプシステムを開発した。本ノード構成は、機能変更が可能な複数のプロトコルプロセッシングモジュールと、それらの搭載機能並び負荷状態に応じて通信パケットを各モジュールに振り分けるディスパッチャで構成される。プロトタイプシステムのプロトコルプロセッシングモジュールは可変要素として、FPGA、CAM(連想メモリ)、MPU を有し、それらを設定する事で機能を変更する。本発表では、提案するノード構成の特徴と機能について議論すると共に、プロトタイプシステムを紹介する。

キーワード　通信ノード、動的機能変更、FPGA、連想メモリ

Dynmaic Reconfigurable Network Node

Takahiro Murooka and Toshiaki Miyazaki
NTT Network Innovation Laboratories
1-1 Hikarinoka Yokosuka-shi, Kanagawa
239-0847, JAPAN.
Phone : +81-468-59-3572
E-mail : murooka@exa.onlab.ntt.co.jp

Abstract　We developed a dynamically reconfigurable network node that is tuned for high-speed and complex multilayer packet manipulation. The key idea is a dynamic function assignment mechanism; each packet processing task is assigned to several processing modules in an on-the-fly manner. With this mechanism, we can freely arrange the modules and add extra ones if more processing power is needed. In addition, the processing modules are realized using field programmable gate arrays (FPGAs), content addressable memory (CAM) and micro processing units (MPUs). Thus, the functionality of each module can be dynamically changed at anytime. In this paper, the system concept and its implementation are described with an example application.

key words　Telecommunication node, Dynamic reconfiguration, FPGA, CAM

1 はじめに

World wide web(WWW) に基づく多様なサービスの出現により、インターネットユーザ数は急激に増加している。金融や商取り引き等、社会生活上に必須なサービスも提供され、インターネットは社会資本の地位を確立している [1]。現在のインターネット網は、通信の信頼生を保証するメカニズムを有していない。そのため、信頼生が要求されるサービスは、サービスを提供するアプリケーションで信頼性を確保している。信頼性の確保は、再送処理やデータの暗号化が使用され、端末での処理を複雑にする。端末の携帯化の流れを考慮すると、端末での処理の単純化は、低消費電力を実現するための重要な要素である。

ネットワーク中を流れるパケットは、OSI の 7 層モデル [2] に準じたプロトコルでパケット転送されている。一般的なネットワークノードは、ハードウエア化が容易な第二層、三層の転送処理を行なっている。第四層以上は、転送処理に加えて複数のパケットの関連性や順序性を考慮した取り扱いが必要になる。関連する一連のパケットの集合をフローと呼ぶ。アプリケーションに応じて、信頼性の確保や、端末処理の一部の肩代りを行なうためには、フロー単位での柔軟な処理が必要になる。大規模なネットワークのノードは、膨大な数のフローを高速に処理する必要がある。しかし、今日のネットワークノードは、第四層以上を柔軟かつ高速に処理するために十分な処理能力と柔軟性を兼ね備えていない。

今後、ネットワーク端末の携帯化や、サービスの高度化を支えるためには、ネットワークノードで第四層以上を処理する必要がある。そのためには、多様なアプリケーションに対応可能で、柔軟性と高速性を両立するネットワークノードが要求される。OSI の 7 層モデルは、この要求応えるノード構成を見出すヒントを与えてくれる。OSI のモデルでは、物理層からアプリケーション層までを 7 層に分割し、各層の処理を単純化している。そのため、第三層以下では、ハードウエア実装が可能である。たとえ、ソフトウエアで実装しても、処理に要するメモリや MPU のステップ数は抑制することができる。各層は基本的に独立であるため、同一層の処理を並列に処理することも可能である。すなわち、各階層に応じた処理機能を複数準備し、それらを組み合わせることで、機能と性能の双方に柔軟なノードを構成することができる。これは、動的機能変更可能な通信ネットワークノード実現の可能性を示している。

著者らは、機能変更可能な複数の処理モジュールと、それらを自由に接続可能な機構を設けた通信ネットワークノードを考案した。各モジュールの機能および接続は動的に変更可能であり、上述した OSI の 7 層モデルの各層の処理を、その負荷に応じて複数の処理モジュールで対応できる。本機構を、DFAM(Dynamic function arrangement mechanism) と呼ぶ。DFAM を使用したネットワークノードは、機能の柔軟性と処理性能の拡張性を兼ね備え、将来のネットワークノードに不可欠なアプリケーションに対応した複雑な処理を高速に実行する。

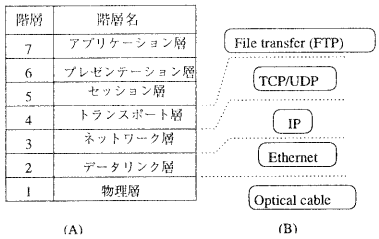

図 1: OSI の 7 層モデル (A) と、それに準じたアプリケーションの構成 (B)。

以下では、2 章で DFAM の概念について解説し、3 章で開発したプロトタイプ装置について述べる。4 章では、アプリケーションの搭載事例について報告し、5 章にまとめを示す。

2 コンセプト

2.1 概念モデル

ネットワークでパケットを送受信するために使用されるプロトコルは図 1(A) に示す OSI の 7 層モデル (OSI モデル) に準じて設計されている。また、ネットワークを使用するアプリケーションも、同様に OSI モデルに準じて実装される。図 1(B) は、アプリケーションとして一般的な FTP(File transfer program) を、光ファイバを媒体としたイーサネット [3] 上に実装した一例を示している。

OSI モデルでは、第一層で物理信号から抽出されたパケットは、上位層に向けて各層で順次処理される。第一層、二層の処理は、媒体の種類に依存し、厳しいタイミング制約が課せられているため、専用のハードウエアとして実装される。パーソナルコンピュータのネットワークカードがこれに相当する。第三層、四層の処理は、多様なプロトコルに対応するため、ソフトウエアとしてオペレーティングシステム (OS) 内に実

装されている。

図 1(B) では、第三層にインターネットプロトコル (IP)[4]、第四層にトランスミッションコントロールプロトコル (TCP)[5] もしくは、ユーザデータグラムプロトコル (UDP) [6] を使用した例を示している。第二層で、一つのイーサネットデータフレームとして処理されたパケットは、階層に応じて順に IP 層、TCP/UDP 層で処理される。IP 層では、パケットのヘッダ部分に記述された IP アドレスを評価し、自己宛のパケットであるか否かを判断する。自己宛の場合、上位層の UDP 層に転送され、そうでない場合は第二層の処理に送信データとして戻される。UDP 層では、IP 層から受け取る複数の IP パケットのペイロードデータを UDP ポート毎に連結し、送信側のアプリケーションが生成したものと同一のデータを作成する。TCP の場合は、UDP の処理に加えて、パケットの到着の信頼性を向上させる処理を行なう。第五層以上は、アプリケーションプログラムとして実装される。この部分では、第四層以下の実装形態に依存しない。

実装された FTP の機能全体は複雑であるが、各層に対応する各部分は単純である。そのため、通常ソフトウエアとして実装する部分をハードウエアとして実装することも可能である。大容量のネットワークに対応可能で、第三層、四層まで処理するノードを実現するには、ハードウエア化は必須となる。

我々が提案する、新たなノード構成の基本モデルを図 2に示す。基本モデルは、処理機能プールと、データプールの二つの部分で構成される。処理機能プールには、OSI モデルの各層に対応する処理機能が複数準備され、それぞれ要求に応じて非同期に起動される。データプールには、処理機能プールで処理されるデータが一時的に蓄積される。蓄積されるデータには、どの処理機能で処理されるものかを表すタグが付加されている。図中のデータプール内に示す正方形はデータを表し、長方形の名前はタグの内容を示している。

機能プール内の、Function A と Function B は OSI モデルのある層に位置する。Function C は、データを処理し、二つのデータを生成している。その一つは、データプールを介して Function D に渡される。Function D は、Function A で処理されるデータを生成してる。Function F は、装置出力であり、データを装置外部に送出する。

我々のモデルは、複雑な通信処理を基本的な処理機能の集合として実現し、処理機能とデータ関係を処理結果に応じて動的に変更する。加えて、同一処理機能を複数準備することで、並列処理を行なう。こ

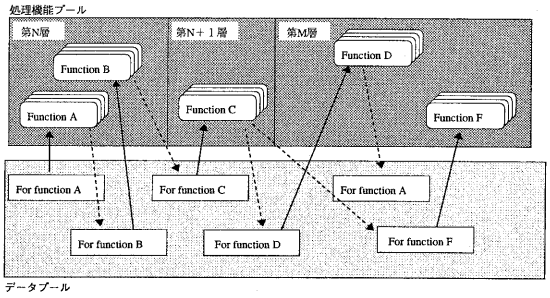

図 2: 提案する処理モデル。

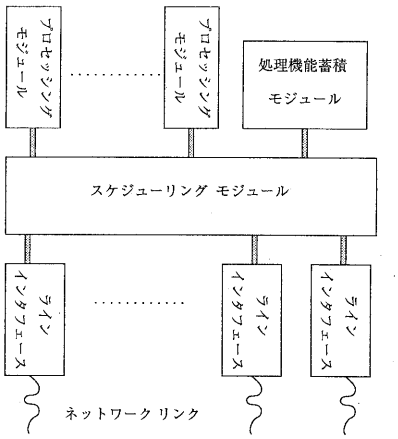

図 3: ネットワークノードの構造モデル。

のコンセプトに基づくメカニズムを DFAM(Dynamic Function Arrangement Mechanism) と呼ぶ。

2.2 DFAM

DFAM によるノードの構造モデルを図 3に示す。構造モデルは、各処理機能を搭載し処理を実行するプロセッシングモジュール、プロセッシングモジュールの負荷状態と処理機能に応じてデータの振り分けを行なうスケジュールリングモジュール、プロセッシングモジュールに搭載する処理機能を蓄積する処理機能蓄積モジュール、ならび、外界と接続するラインインタフェースで構成される。

プロセッシングモジュールは、FPGA や MPU で構成され、処理機能蓄積モジュールに蓄積された FPGA 設定情報やソフトウエアを搭載することで、その機能を実現する。プロセッシングモジュールへの機能搭載は、負荷状態と要求される処理機能に応じてスケ

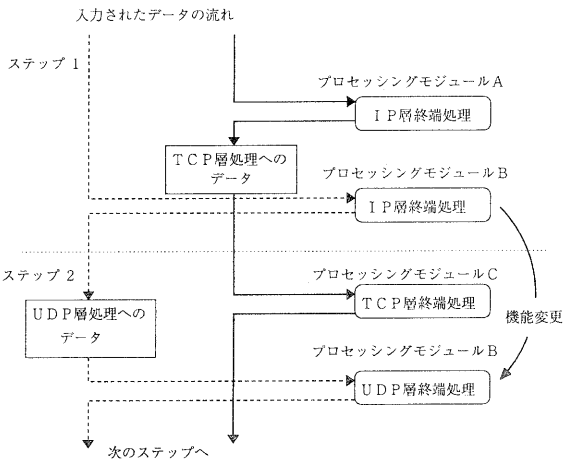

図 4: DAFM での処理の流れ。

ジューリングモジュールが行なう。スケジューリングモジュールと処理機能蓄積モジュールは、DFAM のデータプールに相当する。プロセッシングモジュールは、処理機能プールに相当する。

スケジューリングモジュールは、プロセッシングモジュールとラインインタフェースの双方に共通のインタフェースで接続される。ユーザは、モジュールを自由に組合せ、自分の仕様に最適な装置構成をえることができる。たとえば、複雑な処理を行なうノードを構成する場合、数多くのプロセッシングモジュールを搭載することで対応できる。また、低速のネットワークを多数収容するノードを構成する場合、プロセッシングモジュール数を少なくし、その分多くのラインインタフェースを搭載する。

DFAM の動作を図 4に示す。実線で示す処理の流れでは、データは最初にプロセッシングモジュール A に搭載された IP 終端機能で処理され、次にプロセッシングモジュール C に搭載された TCP 終端機能に転送されている。破線で示す処理の流れでは、プロセッシングモジュール A がデータを処理中で、プロセッシングモジュール B が A と同一機能を搭載していて、かつ処理をしていない場合を示している。この時、データはプロセッシングモジュール B で処理される。処理の結果、どのプロセッシングモジュールにも搭載されていない UDP 終端処理が必要になり、次のステップで無負荷となるプロセッシングモジュール B の機能を UDP 終端処理をに変更し、データを再びプロセッシングモジュール B に渡している。

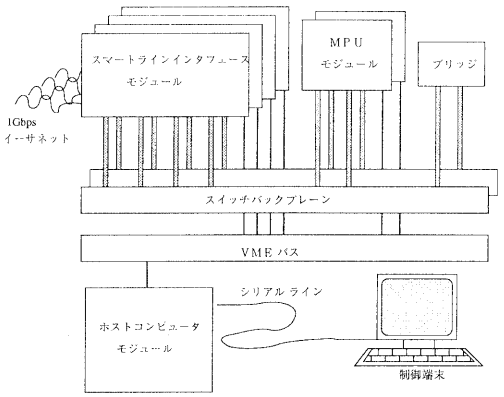

図 5: プロトタイプノードの構成。

プロセッシングモジュール 間の効率良いデータ受渡しが、本モデルを実装した装置の性能を左右する。パーソナルコンピュータでは、低コストと拡張性を両立するため、配線共有のバスをデータ受渡しに使用している。しかし、配線共有のバスは帯域を共有すると共に、繁雑な占有権の受渡オーバーヘッドのため、本来バスが持つデータ転送性能を確保するのは困難である。一方、クロスバスイッチ等を用いた場合、コスト的には配線共有のバスに比べて割高になるが、各データ転送が独立に行なわれるため、各モジュール間の転送帯域は確保できる。そのため、DFAM をスイッチ型のモジュール間データ転送を基本とする考え方で実装した。

3 プロトタイプノード

3.1 装置概要

図 3 の構造モデルに基づいて試作した、プロトタイプノードの構成を図 5 に示す。プロトタイプノードは、4 枚のスマートラインインタフェースモジュール、2 枚の MPU モジュール、 1 枚のホストコンピュータモジュールで構成される。ホストコンピュータモジュールを除く全てのモジュールは並列な 2 つのスイッチバックプレーンで接続される。スマートラインインタフェースモジュールは、図 3のプロセッシングモジュールとラインインタフェース双方の機能を有し、ハードウエアでデータを処理する。MPU モジュールは高性能マイクロプロセッサを搭載し、ハードウエアだけでは困難な OSI モデル 5 層以上のデータをソフトウエアで処理する。スイッチバックプレーンは、データを各モジュール間で互いに独立に 2Gbps の帯域で

転送可能である [8]。データ到着の競合を抑制するため、プロトタイプでは二つのスイッチバックプレーンを用い、一つのモジュールに二つのバックプレーンインタフェースを搭載している。ホストコンピュータモジュールは、装置全体の制御と設定を行なう。ホストコンピュータと各モジュールは、制御データの受渡しを、VME バス [7] を介して行なう。

3.2 スマートラインインタフェース

スマートラインインタフェースの構成を、図 6 に示す。このモジュールは、大規模 FPGA、CAM(連想メモリ)、高速 SRAM、1Gbps のネットワークインタフェース、2 つのスイッチバックプレーンインタフェース、VME バスインタフェース、オンボードコントローラで構成される。処理機能は、FPGA、SRAM、CAM を使用し、ハードウエアとして実装する。搭載する機能の制御や、メモリ内データのメンテナンスはオンボードコントローラが行なう。バックプレーンインタフェースと、ネットワークインタフェースは FPGA に直接接続さる。FPGA に、搭載された処理機能は、各インタフェースからのデータを直接取り扱うことができる。VME バスインタフェースと オンボードコントローラは、ボード上のローカルな PCI バスに接続される。オンボードコントローラは FPGA からの割り込み処理要求に迅速に対応する必要がある。そのため、リアルタイム OS である RT-Linux[9] を OS のカーネルに採用した。

FPGA に搭載する処理機能の開発を容易化するため、テンプレートを準備した。その構成を、図 7に示す。周辺部品へのインタフェース回路を部品毎に準備することで、部品固有の処理をアプリケーション回路から隠蔽し、開発期間の短縮を可能とした。ユーザの処理機能は、各プロトコルに固有な部分を処理するプロトコル処理エンジンと、ユーザ独自の処理であるユーザアプリケーションエンジンで実現する。

FPGA の設定／再設定はオンボードコントローラのコマンドで実行される。設定に必要なデータはファイルとしてオンボードコントローラに蓄積される。よって、複数種類の FPGA 設定を準備し、必要に応じて任意に変更することが可能である。理想的には、第二層の最小データフレーム到着間隔である 96nsec [3] 以内で設定変更が終了する必要がある。しかし、それを可能とする動的変更可能な FPGA は現在市販されていない。プロトタイプでは、論理容量とクロック周波数の要求条件から、大規模で、高速な FPGA(APEX 20K-400[11]) を採用している。この FPGA は、設定に約 1 秒を必要とする。そこで、FPGA の機能変更中の代用として必要数より多いプロセッシングモジュールを冗長に搭載することで、動的機能変更を装置レベルで実現している。

開発したスマートラインインタフェースの概観を図 8に示す。Compact-PCI システムに標準的なサイズのカードであり、オンボードプロセッサをメザニンカードとして搭載している。FPGA は、オンボードコントローラの下に搭載されている。

3.3 DFAM のインプリメント

DFAM が処理機能プールとデータプールで構成され、並列処理によるパケット処理を可能にすることは、第二章で述べた。プロトタイプでは、処理機能プールをスマートラインインタフェースの FPGA、MPU モジュール、ホストコンピュータモジュールを使用して実装する。ホストコンピュータモジュールは、FPGA の設定ファイル、MPU モジュールのプログラムファイルを蓄積し、必要に応じて各モジュールに渡し、その機能を変更する。一方、データプールは、スマートラインインタフェースモジュールのスイッチバックプレーンインタフェースと、スイッチバックプレーンを使用して実装する。データフレームの一時的な蓄積と転送先の制御は、スイッチバックプレーンインタフェース内の FIFO と、インタフェースの転送先制御レジスタの操作で行なう。

DFAM では、データに付加するラベルで、その取り扱いを決定する。ラベル管理は装置全体の性能に影響を与える。プロトタイプではラベル管理手法の違いによる影響を評価するために、分散的ラベル管理と、集中的ラベル管理の双方でのインプリメントを可能にした。

分散的ラベル管理

ラベルは、各スマートラインインタフェースが個別に管理する。他のモジュールとの整合性は、VME バスを介して情報交換することで確保する。

集中的ラベル管理

ラベル管理は、ホストプロセッサモジュール、もしくは MPU モジュールのどちらかで行う。この場合、プロセッサの処理能力が装置全体の性能を支配する。

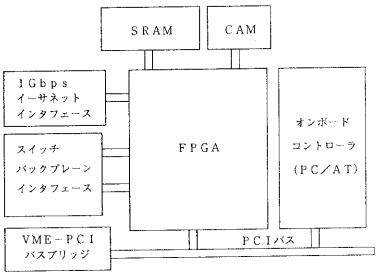

図 6: スマートラインインタフェースの構成

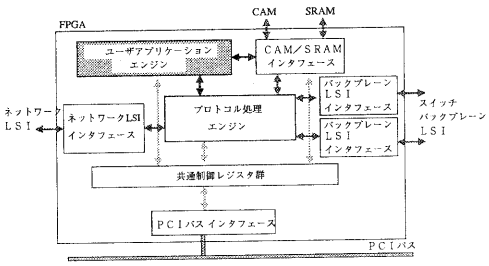

図 7: FPGA 内のモジュール構成

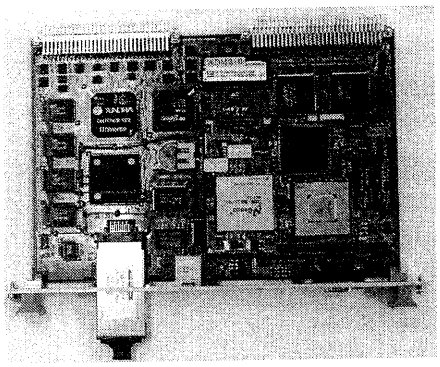

図 8: スマートラインインタフェース

最初のアプリケーションのインプリメントでは、早期に動作させるという意味から集中的ラベル管理を採用した。

4 アプリケーションの搭載

最初のアプリケーションとして、DFAM 内でのラベル転送処理との整合により、短時間で実装可能と考えられる MPLS(multi-protocol label switch)[10] に基づくルータを選択した。MPLS ルータは、パケット転送をそのヘッダ情報を基に作成したラベルを用いて転送処理を行なう。

MPLS ルータのパケット転送概要を、図 9 に示す。ネットワーク入口側のノードは、ネットワークへ流入する全てのパケットにラベル付加する。 ラベルには、そのパケットの転送先の他、ネットワーク内で転送優先順位等の付帯情報も含まれる。また、ラベルはネットワーク内で一意にする必要があるため、集中的に管理される。MPLS ルータは、パケットの配送処理をラベルの情報のみで行う。

プロトタイプに搭載した構造を、図 10 に示す。MPLS ルータの実装には、 6 枚のスマートラインインタフェースとホストコンピュータモジュールを使用した。MPLS ルータは、上位層の処理は行なわないので、MPU モジュールは使用していない。バックプレーンインタフェースの共通化により MPU モジュールとスマートラインインタフェースは相互に入れ替え可能である。本実装では、6 枚のスマートラインインタフェースカードを搭載した。

実線矢印は、内部でのデータと、制御の流れを示している。実線長方形は、装置モジュール／サブモジュールを示している。破線長方形は、搭載された MPLS ルータの機能ブロックを示している。

MPLS ルータ機能で、他のルータとラベル情報を交換し、自己ルータが使用するラベルの管理を行なう

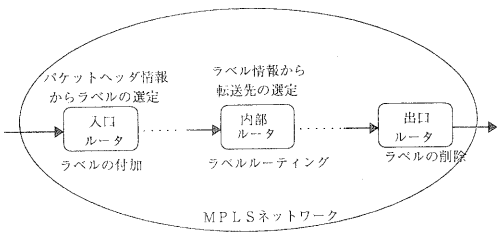

図 9: MPLS でのパケット転送

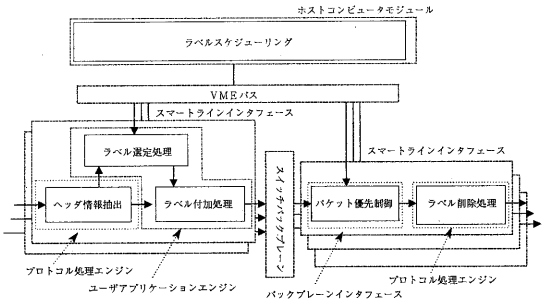

図 10: MPLS ルータの搭載例。

部分は、ホストコンピュータモジュールに搭載した。ラベル添付、削除、優先制御の部分は、全て スマートラインインタフェースの FPGA と、スイッチバックプレーンインタフェースを使用して実装した。

プロトコル処理エンジンは、IP ヘッダ情報を抽出し、条件に一致したパケットを選別する第三層の処理を搭載した。MPLS に独自な、ラベルの検索と添付はユーザアプリケーションエンジンに実装した。優先制御部分は、バックプレーンインタフェースの FIFO を使用した。優先処理の細かな制御を可能にするため、ホストプロセッサからアクセス可能な制御レジスタを設けた。

装置内では、ネットワークインタフェースで受信したデータに FADM のデータプールで使用するラベルを付加する。本来、FADM では機能情報をラベルとして付加する。実験では、同一機能を全てのスマートラインインタフェースに搭載している。そのため、ルータとして動作するためには、物理的な転送先を明確にする必要がある。そこで、モジュールそれぞれに異なる機能識別 ID を準備し、それをヘッダ情報とした。また、今回は動的機能変更は使用していない。

5 まとめ

本稿では、高速性と柔軟なパケット処理の双方を同時に実現する動的再構成可能なネットワークノード構成について紹介した。本構成は、OSI の 7 層モデルに準じて、複数の処理モジュールに分割搭載される単純な機能を、バックプレーンを用いて組み合わせることで、複雑な処理機能を実現することと、処理モジュールの機能を要求に応じて動的に変更することを特徴としている。同一処理を、複数の処理モジュールで並列に実行でき、高速なノード構築が可能である。ユーザは、この機能を使用して、スケーラブルで柔軟なノードを実現できる。上記のコンセプトに基づき、プロトタイプノードを構築し、MPLS に基づくルータの搭載実験を行った。

現在、搭載機能と性能の評価を進めている。現時点では、FPGA に搭載する論理回路の設計は、テンプレート用いて手作業で行なっている。今後、設計作業を容易化する CAD システムの構築を行なう予定である。

参考文献

[1] A. Bhargava and B. Bhargava, "Measurements and quality of service issues in electric commerce software," *in Proc. Application-Specific System and Software Engineering and Technology*, pp. 26–33, 1999

[2] Organization International Normalization (ISO), "Information technology – Open System Interconnection – Basic Reference Model: The Basic Model," ISO/IEC 7498-1, 1994.

[3] ANSI/IEEE P802.3z/D5.0, "Media Access Control (MAC) Parameters, Physical Layer, Repeater and Management Parameters for 1000Mb/s Operation," 1998.

[4] IETF (Internet Engineering Task Force), "RFC 791: INTERNET PROTOCOL," 1981.

[5] IETF (Internet Engineering Task Force), "RFC 793: TRANSMISSION CONTROL PROTOCOL," 1981.

[6] IETF (Internet Engineering Task Force), "RFC 768: User Datagram Protocol," 1980.

[7] VME Bus International Trade Association (VITA), *http://www.vita.com/* , Web page.

[8] H. Gang, J.H. Aylor and R.H Klenke, "An ADEPT performance model of the Mercury RACEway crossbar interconnection network," *in Proc. the 6th International Conference on Parallel Interconnects*, pp. 83–90, 1999.

[9] M. Barbanov and V. Yodaiken, "Introducing real-time linux," *Linux Journal*, Vol. 34, pp. 19–23, Feb., 1997.

[10] D. O. Awduche, "MPLS and Traffic Engineering in IP Networks," *IEEE Communications magazine*, vol. 37, no.12, pp. 42–47, 1999.

[11] ———, *http://www.altera.com/* , Web page.