分散制御用リアルタイム通信 *Responsive Link* の設計および実装

山 崎 信 行[†,☆]

本論文では，各種ロボットやメカトロニクス，FA，OA 等の様々な分散リアルタイム制御用途に使用することのできるリアルタイム通信規格 *Responsive Link* の設計・実装について述べる．*Responsive Link* は，柔軟なリアルタイム通信を実現するために，ソフトリアルタイム通信（データリンク）とハードリアルタイム通信（イベントリンク）の分離，パケットに優先度を付加しノードごとに高優先度パケットが低優先度パケットの追い越し，パケットの優先度が異なると優先度ごとに別経路を設定して専用回線や迂回路を実現可能，ノードごとに優先度を付け替えることができ分散管理型でパケットの加減速を制御可能，ハードウェアによるエラー訂正，通信速度を動的に変更可能，トポロジーフリー，Hot-Plug&Play 等の様々な機能を実現する．現在，*Responsive Link* は，並列分散制御用のシステムオンチップである *Responsive Processor* および *Responsive Multithreaded Processor* 上に実現されている．また，国内では情報処理学会試行標準 IPSJ-TS 0006:2003 として標準化され，国際的には ISO/IEC SC25 WG4 において標準化作業が行われている．

Design and Implementation of Real-time Communication *Responsive Link* for Distributed Control

Nobuyuki Yamasaki[†,☆]

In this paper, we design and implement *Responsive Link* for distributed real-time control, which can be applied to various electronic control systems including robot systems, mechatronic systems, home automation, office automation, factory automation, etc. In order to realize flexible real-time communications, *Responsive Link* has many unique features including separation of data transmission for soft real-time (data link) and event transmission for hard real-time (event link), independent routing of the data link and the event link, priority based packet overtaking (the packet with higher priority overtakes other packets at each node.), packet acceleration/deceleration using priority replacement (packet priority can be replaced with a new priority level at each node to accelerate/decelerate packets under distributed control.), prioritized routing (when multiple packets with different priority levels are sent to the same destination, the different route can be set to realize exclusive communication lines or detours.), dynamically variable link speed (800, 400, 200, 100, 50, 25, 12.5 [Mbaud]), hot-plug&play, topology free, etc. *Responsive Link* is implemented on *Responsive Processor* and *RMT Processor* for distributed control. *Responsive Link* has been standardized as IPSJ-TS 0006:2003 in Japan. *Responsive Link* is also under standardisation at ISO/IEC JTC1 SC25 WG4.

1. はじめに

近年，各種ロボットやメカトロニクス，ファクトリオートメーション，ユービキタスコンピューティング等の様々な分野において，分散リアルタイム制御が行われつつある．これらのように，単一プロセッサでは制御しきれないような大規模システムや，アクチュエータやセンサが広範囲に広がっているような制御システムを設計しようとすると，効率的な分散リアルタイム制御が必要になってくる．そのためには，プロセッサやコントローラ間のリアルタイム通信が必要不可欠であるが，分散制御用途のリアルタイム通信規格はほとんど存在していなかった．

そこで，本論文では，分散制御を可能にするためのリアルタイム通信規格である *Responsive Link* の設計・実装について述べる．ここで，*Responsive Link* の仕様・概要は文献 1) に記載されているが，本論文

† 慶應義塾大学
Keio University
☆ 現在，産業技術総合研究所特別研究員を兼務
Presently with an associate researcher of National Institute of Advanced Industrial Science and Technology

では *Responsive Link* の具体的な設計・実装に関して述べる．

現在，*Responsive Link* は情報処理学会試行標準 WG6[2] において IPSJ-TS 0006:2003 として国内の標準化が行われ，ISO/IEC JTC1 SC25 WG4[3] において国際標準化作業を行っている．国際標準化が実現されると，異なるシステム間でのリアルタイム通信も可能になり，より大規模な分散リアルタイム制御システムが実現可能になると考えられる．

2. リアルタイム性

リアルタイム性とは，処理や通信等の正しさが時間にも依存するという性質である．狭義には，与えられた時間制約（デッドライン）を守るということを意味する．

2.1 ハードリアルタイムとソフトリアルタイム

リアルタイム性は，以下の 2 つに大別することができる．

2.1.1 ハードリアルタイム性

ハードリアルタイム性とは，必ず時間制約を守らなければならない性質であり，時間制約を少しでも破ると価値が 0 になる性質である．狭義には，時間制約を破った場合，システムに損害を与える可能性のあるリアルタイム性のことである．主に制御系の通信や演算を行うタスクが要求するリアルタイム性であり，以下のような特徴がある．

通信 データ量は小さいが，レイテンシ（遅延）に対する要求が厳しい．スループットよりレイテンシを重視する．主に制御コマンドや同期信号等であり，サイズは小さいが時間遅延に厳しく，時間制約を破ると価値がない（動作しないだけではなく危険をともなうこともある）場合が多い．

演算 演算量は小さい場合が多いが，時間制約を厳守する必要性がある．

時間粒度とデッドライン 時間粒度が小さく，デッドラインが短い場合が多い（100 [μsec]〜10 [msec] 程度）．

2.1.2 ソフトリアルタイム性

ソフトリアルタイム性とは，時間制約を多少破ることを許容する性質であり，時間制約を破っても価値はただちに 0 にはならない．多くの場合，時間制約を破ると，時間経過とともに価値が減少していく性質を指す．また，時間制約を破っても，システム自身に損害を与えることはない．主にマルチメディア系の通信や演算を行うタスクが要求するリアルタイム性であり，以下のような特徴がある．

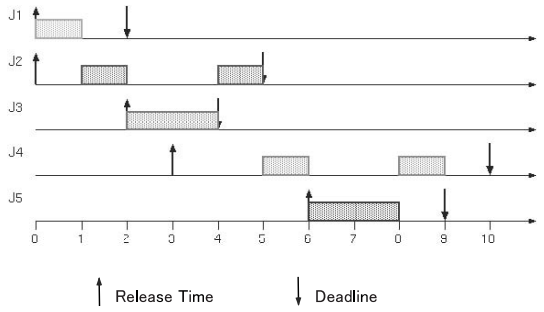

図 1 EDF スケジューリング
Fig. 1 EDF scheduling.

通信 データ量が非常に大きく（ストリーミング等），レイテンシよりもスループットを重視する．通常，バンド幅保証によって実現される場合が多く，保証されたバンド幅内でマルチメディアデータ（映像等）を流す場合が多い．

演算 演算量はそれなりに大きい（MPEG のデコード等）が，時間制約は制御系に比較すれば厳しくない．

時間粒度とデッドライン 時間粒度が比較的大きく，デッドラインが長い場合が多い（10 [msec]〜1 [sec] 程度）．

このように，同じリアルタイム性といっても，ソフトリアルタイムとハードリアルタイムでは，リアルタイム性を要求するアプリケーションも異なるし，特徴も異なることが分かる．これら性質の異なるリアルタイム性を同時に扱うことのできるリアルタイム通信規格として *Responsive Link* の設計・実装を行う．

2.2 リアルタイムスケジューリング

ネットワークで相互に接続されたリアルタイムシステムにおいては，すべての起こりうる場合を想定することは不可能であるので，リアルタイムスケジューリングが必要となる．

リアルタイムスケジューラには，動的スケジューリングとして EDF（Earliest Deadline First）等があり，静的スケジューリングとして RM（Rate Monotonic）等があるが，たいていの場合，優先度に従ってプリエンプションを行いながら実行を行う．**図 1** に EDF によるスケジューリング例を示す．

プリエンプションは，演算の場合はコンテキストスイッチに相当するが，通信の場合はパケットの追い越しに相当し，まずパケットの追い越しを実現する必要がある．

ここで，従来研究として，低優先度パケットによって待たされている高優先度パケットの優先度を先送りし，待たされている低優先度パケットの優先度を後続

の高優先度パケットの優先度に一時的に変更し，待たされていた低優先度パケットを強制的に先送りすることができる技術が開発されている[4),5)]．これは，オペレーティングシステム等で用いられている優先度継承を応用していると考えられるが，この技術では，同一通信経路上において，先行する低優先度パケットが衝突を起こしている際に，先行する低優先度パケットを後続の高優先度パケットが追い越すことができない．また，同一通信経路上において，先行する低優先度パケットが衝突を起こすたびに，少なくとも，低優先度パケットの優先度が後方の高優先度パケットの優先度に変更され，結果的に，低優先度パケットの優先度が高くなり，低優先度パケットにもかかわらず，場合によっては，送信先に早く着きすぎてしまい，低優先度パケットのジッタが大きくなってしまうという問題が発生する．

それに対して，*Responsive Link* では，今まで実現されていなかった優先度によるパケットの追い越しを実現する機構を研究開発し，その機構を有効に用いてリアルタイム通信を実現することを目的とする．基本的にはリアルタイムスケジューラでスケジューリングされた通信を最適に（つまりリアルタイムに）行うことを実現する．

2.3 既存の通信インタフェース

現在，汎用の高速通信インタフェースとして Ethernet，ATM，Fibre Channel，IEEE-1394，USB2.0 等の様々な規格が存在している．

Ethernet は，安価でポピュラーな通信インタフェースであるが，CSMA/CD 方式を採用しており，通信の際に衝突があった場合は再送するというプロトコルである．したがって，通信の際にまったく衝突がなければ通信レイテンシが小さく通信速度も速いが，ひとたび衝突が生じると再送を行い始めるので最悪通信時間を規定できずリアルタイム性がないという大きな問題点がある．関連して，ベストケースとワーストケースの差が非常に大きいというのが問題となる．

ATM はバーチャルチャネルを張ってのバンド幅保証が可能なので，ソフトリアルタイム通信が可能である．しかしながら，規格自体が複雑で大きいので，分散制御等の組み込み用途には事実上使用することは困難である．

IEEE1394 は，主に AV 機器やパーソナルコンピュータの I/O 機器を接続するための規格であり，アイソクロナス転送によってソフトリアルタイム通信を実現することが可能である．しかしながら，アイソクロナス通信モードではエラー訂正を行っていない，最大通信ノード数が 63 と少ない，Plug&Play 時にネットワーク全体にリセットがかかる，トポロジーがスター構造のみでループを許さない，耐故障性がない等，分散制御用リアルタイム通信としては多くの問題点をかかえている．

USB2.0 は，主にパーソナルコンピュータの I/O 機器を接続するための規格であり，豊富な PC 用 I/O 機器を手軽に接続することが可能である．しかしながら，最大通信ノード数が 127 と少ない，トポロジーがツリー構造のみでループを許さない，必ずルートコントローラ（PC）が必要である，耐故障性がない等，分散制御用リアルタイム通信としては IEEE1394 と同様に多くの問題点をかかえている．

3. 設　　計

Responsive Link は，各種ロボット，自動車，プラント，ホームオートメーション等の種々の分散制御を実現するために必要なハードリアルタイム通信，および，画像，音声等のマルチメディアデータを滑らかに伝送するために必要なソフトリアルタイム通信の両方を同時に可能にするように設計を行う．

3.1 ハードリアルタイム通信とソフトリアルタイム通信の分離

ソフトリアルタイム通信（以下，単にデータと呼ぶ）のデータサイズ（画像データ，音声データ等）は大きく，それに対してハードリアルタイム通信（以下，単にイベントと呼ぶ）のデータサイズ（制御コマンド，同期信号等）は非常に小さい．したがって，従来型の 1 系統の通信路ですべての通信を行う方法では，同時に通信すべき通信データとして，大量のデータパケットと，ごくわずかではあるが分散リアルタイム制御用途には非常に重要なイベントパケットが同一種類のパケットとして存在する．データとイベントを，共有された同一の通信線を通して時分割に通信を行う従来方式ではイベント伝達の時間が正確にバウンドできないので，ハードリアルタイムシステムは実現困難であると考えられる．

また，複数のモジュールで 1 つの通信チャネルを共有するシリアルバスでは，同時に何台のモジュールが通信するかによってバンド幅が動的に変化し時間をバウンドすることが困難であり，実効速度も出にくい．

さらに，リアルタイム通信におけるトレードオフとして，ソフトリアルタイム通信は主にバルク的なマルチメディアデータの通信等に用いられ，ハードリアルタイム通信は主に制御等に用いられるので，

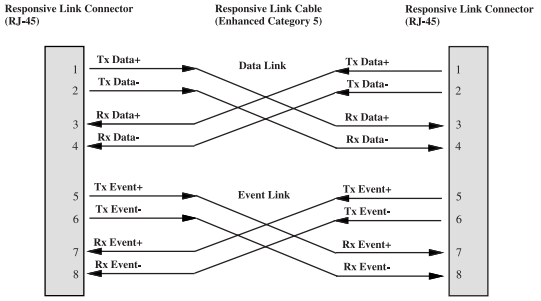

図 2 *Responsive Link* インタフェース
Fig. 2 *Responsive Link* interface.

- ソフトリアルタイム：バンド幅保証 ⇒ スループットをできるだけ上げたい，
- ハードリアルタイム：レイテンシ保証 ⇒ レイテンシをできるだけ小さくしたい，

という要求がある．しかしながらパケットサイズを大きくするとスループットは高くなるが，同時にレイテンシも長くなる．逆にパケットサイズを小さくするとレイテンシは短くなるが，オーバヘッドが大きくなりスループットが低くなる．

したがって，まずデータラインとイベントラインを分離し，かつ各ラインの結合形態を point-to-point の双方向シリアル通信として設計・実装する方針をとる（**図 2** 参照）．以下，それぞれをデータリンク，イベントリンクと呼ぶ．データリンクではパケットサイズを固定長かつ大きめにしてソフトリアルタイム通信に使用し，イベントリンクではパケットサイズを固定長かつ小さめにしてハードリアルタイム通信に使用する．

3.2 パケットフォーマット

図 3 に *Responsive Link* のパケットフォーマットを示す．通信パケットは，ヘッダ部，ペイロード部，トレイラ部から構成する．ヘッダ部は優先度付きのネットワークアドレスから構成し，トレイラ部は制御情報とステータスから構成する．

通信パケットは固定長で，ハードリアルタイム通信用のイベントリンクのパケットサイズは 16 バイト（ペイロード：8 バイト）と小さくし，ソフトリアルタイム通信用のデータリンクのパケットサイズは 64 バイト（ペイロード：56 バイト）と比較的大きくする．

図 1 のようなリアルタイム通信を行うためには，優先度を用いた追い越し機構を設計・実装する必要がある．まずはそのために，図 3 の通信パケットのヘッダ部に対して，**図 4** に示すようにネットワークアドレスに優先度を付加する．優先度のレベルは，小さすぎると十分に優先度が機能せず，大きすぎるとオーバヘッドとなる．ここで，Rate Monotonic Scheduling では

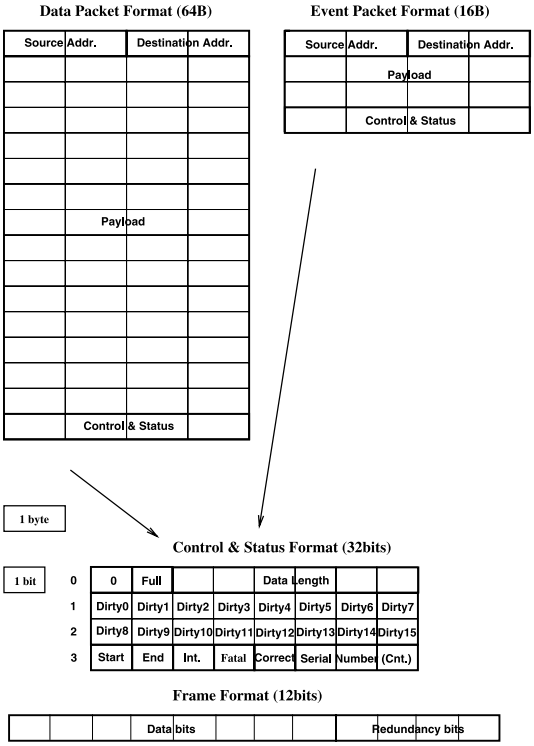

図 3 *Responsive Link* のパケットフォーマット
Fig. 3 Packet format of *Responsive Link*.

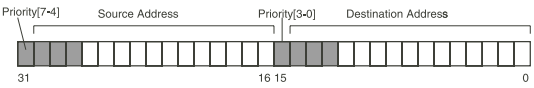

図 4 *Responsive Link* のヘッダフォーマット
Fig. 4 Header format of *Responsive Link*.

256 レベルで十分であるということが分かっているので[6]，*Responsive Link* では 256 レベル（8bit）の優先度を採用する．優先度は 0 が一番低く，数字が大きくなるに従って高くなる．

Responsive Link の最大通信ノード数は，ネットワークアドレス長に制限され，優先度を使用しない場合，理論的には 2^{32} ノードとなる（図 4 参照）．*Responsive Link* の規格で推奨している使用法（ノードごとにノードアドレスを割り当て，12 bit の送信元アドレス，12 bit の送信先アドレス，8 bit の優先度を用いてルーティングを行う）の場合には，$2^{12} = 4096$ ノードとなる．4096 よりノード数が大きなシステムを構築する際には，経路にアドレスを割り当てる（24 bit のネットワークアドレスと 8 bit の優先度を用いてルーティングを行う）ことにより $2^{24} = 16\,\mathrm{M}$ ノードまでのノード数をサポートする．

3.3 優先度による追い越し機構

優先度を用いたパケットの追い越し機構を実現する

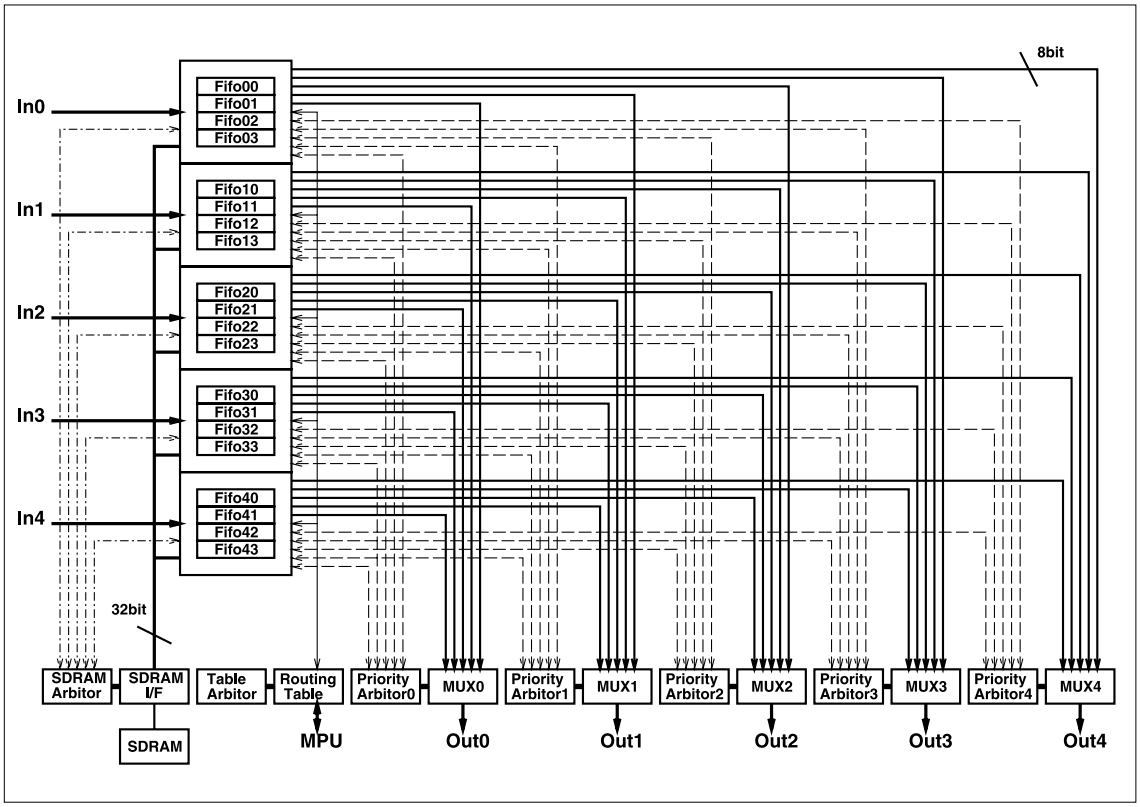

図 5 *Responsive Link* のネットワークスイッチ
Fig. 5 Network switch of *Responsive Link*.

ために，追い越し用バッファと退避用外部記憶を有したネットワークスイッチを設計する[7]．**図 5** は 5 入力 5 出力で 1 つの入力部あたり追い越し用バッファが 4 パケット分あるネットワークスイッチの構成を示している．図 5 において，最後の数字はポート番号を示している．入力ポート（In0〜4）から入力された通信パケットは，通信ノードで衝突しない場合，そのまま出力ポート（Out0〜4）へ出力を行う．異なる入力ポートから入力された通信パケットが同じ出力ポートに出力を行う場合，通信パケットに付加された優先度に従い，低い優先度の通信パケットを追い越し用バッファ（意味的には追い越され用バッファ）に貯めて出力を待たせ，高い優先度の通信パケットを先に出力させる．高い優先度の通信パケットの出力の後に低い優先度の通信パケットを追い越し用バッファから出力ポートに出力し，優先度に従った通信パケットの追い越しを行う．

この際，内部のスイッチングは，ヘッダ部受信のオーバヘッドおよびルーティングテーブルの参照時間を隠蔽するために（3.4 節，4.1 節参照），図 5 のように 8 bit パラレル（byte 単位）で行うように設計する．

上記の通信パケットの追い越しを実現するために通信パケットの大きさと等しい追い越し用バッファを複数本入力ポート側に設ける．さらに，出力が待たされ続けているときに入力が入り続けバッファが溢れそうになった場合に，追い越し用バッファの内容を一時的に退避するための退避用外部記憶を設けることができるようにする．

図 6 は図 5 のネットワークスイッチの 1 つの入力部の詳細を示している．図 6 において，最後の数字はポート番号を示している．通信パケットの追い越しを行うために，まず，入力ポート（In）から入力された通信パケットを，入力ポインタ（In-Pointer）で指し示されている追い越し用バッファ 0 から追い越し用バッファ 3 のうち使用されていない空バッファに書き込む．入力パケットのヘッダ部分は必ずすべて受信して追い越し用バッファに書き込み，その受信されたヘッダを元に**図 7** のようなルーティングテーブルを参照し出力ポート番号と優先度を得る．得られた出力ポート番号は図 6 のリンクストローブ（L0〜L4）に書き込む．たとえば L2 ビットが有効であればその入力パケットの出力先は出力ポート 2 であることを示す．

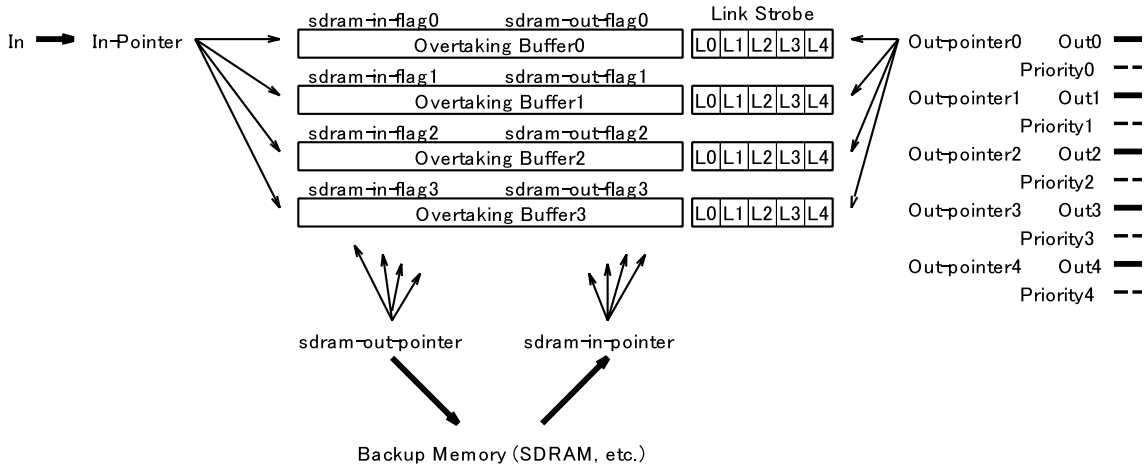

図 6 *Responsive Link* の追い越し用バッファ
Fig. 6 Overtaking buffer of *Responsive Link*.

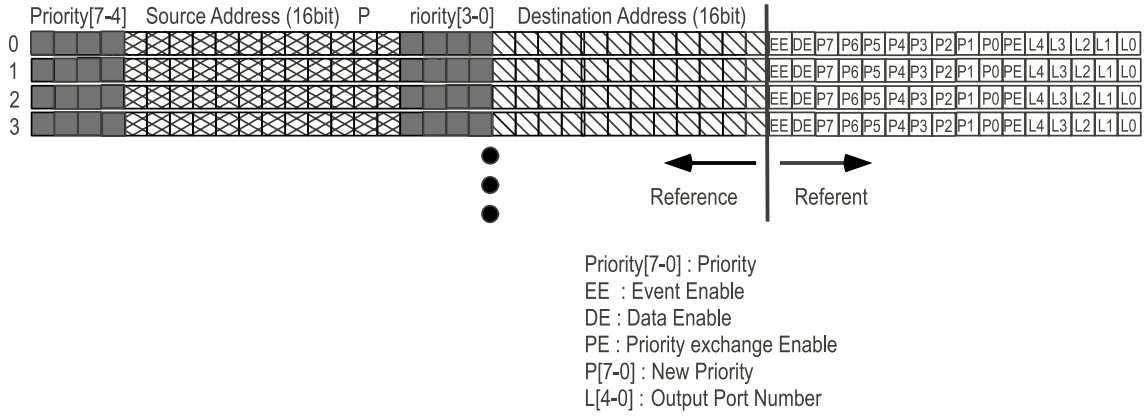

図 7 *Responsive Link* のルーティングテーブル
Fig. 7 Routing table of *Responsive Link*.

図 6 において L0 から L4 までの複数ビットが有効であればマルチキャストを意味し，すべて有効であればブロードキャストを意味する．入力部の出力側は出力ポートごと（Out0〜Out4）にそれぞれ独立に各追い越し用バッファのリンクストローブを参照し，自出力ポートのリンクストローブが有効な場合，出力側ポート側に配置された当該優先度調停器（図 5 の Priority ArbitorN）に対して優先度とともに出力要求を行う．図 6 の PriorityN は図 5 の Priority ArbitorN に接続されている．優先度調停器は，出力要求が 1 つの入力ポートからだけある場合はただちに出力許可を与え，出力要求が複数ある場合は優先度の一番高いものに出力許可を与えるようにする．一番優先度の高い要求が複数ある場合は，ラウンドロビン方式で出力許可を与える．

通信パケットの衝突がない場合や，衝突があってもその時点での最高優先度の通信パケットの場合は，ヘッダの受信とルーティングテーブル参照の遅延時間後にただちに出力を開始する．入力部の各出力ポート側ではパケットの送信終了直後に対応するリンクストローブを無効にし，すべてのリンクストローブが無効になったらそのバッファが空であることを意味する．

たとえば，In-pointer が追い越し用バッファ 1 を指している場合，入力ポート In から入力されたパケットは，まずヘッダ部が追い越し用バッファ 1 に入る．次にそのヘッダを元にルーティングテーブルを引き，リンクストローブと優先度を得る．たとえば，L1 と L3 が有効だった場合，Out-pointer1 と Out-pointer3 はともにその追い越し用バッファ 1 を指し，Out1 と Out3 側が出力要求とともにその優先度をそれぞれ Priority1 と Priority3 に出力する．たとえば，Out3 にすぐに出力可能であれば，出力許可が Priority Arbitor3 から

与えられるので，ただちに追い越し用バッファ1からOut3に出力を開始する．出力が終われば，Out3側が追い越し用バッファ1のL3をクリアする．また，Out1にはただちに出力許可がおりなかったとすると，出力許可が得られるまで出力要求と優先度をPriorty1に出力し続ける．ここで，Out1への出力待ちの状態で，同じくOut1へ出力したい高優先度パケットが新たに追い越し用バッファ2に入ってきた場合，Out-pointer1はより優先度の高いパケットの入っている追い越し用バッファ2を指すようになり，その高優先度パケットの出力要求と優先度をPriority1に出力するようになる．後から到着した高優先度パケットの出力が終わると，他にOut1に出力したい高優先度パケットがない場合，Out-pointer1は再び追い越し用バッファ1を指して，同様に出力を継続しようとする．このように，同一系路上の先行する低優先度パケットが待たされている際にも，後続の高優先度パケットが追い越していくことを可能にする．

追い越し用バッファに通信パケットが待たされているにもかかわらず，入力パケットが連続して入力された場合，バッファを溢れさせないように少なくとも1つ以上の空バッファをつねに用意するようにする．

図6において，空バッファが少なくなっていき残り1本になってしまった場合，次の入力パケットは退避用外部記憶（SDRAM等で実装）に退避を行うようにする．そのバッファを他でその期間に使用されないように，sdram-in-flagNというフラグを立てておくようにする．出力が進んで空バッファの残りが多くなり2本以上になると，退避用外部記憶に退避されていた入力パケットを優先度を考慮して追い越し用バッファに書き戻すことにより，出力を継続する．退避用外部記憶から書き戻されるバッファは，その期間に他で利用されないようにsdram-out-flagNでフラグを立てておく．

この機構を少ない回路量で実現するために，退避用外部記憶のメモリ領域を優先度ごとに区切り，同じ優先度の領域には到着順に退避する．復帰するときには，高い優先度の領域から順に退避するようにし，同じ優先度内では退避した順に復帰するようにする．この機能を実現するために，同じ優先度のメモリ領域は到着順のリングバッファとして設計する．退避用外部記憶が使用されている際には，退避用外部記憶に退避されている最高優先度のパケットのメモリ領域を指し示すポインタや，リングバッファのリード/ライト・ポインタ等は，つねに最新となるように更新を行うようにする．こうすることで，退避用外部記憶のメモリ効率は悪くなってしまうが，リアルタイム性と回路数削減の両方を実現する．

図6において退避用外部記憶出力ポインタ（sdram-out-pointer）は，退避されるべき入力パケットの退避用外部記憶（SDRAM）へのセーブアドレスを指し示すようにし，退避用外部記憶入力ポインタ（sdram-in-pointer）はデータがリストアされる空バッファのアドレスを指し示すようにする．

また，退避用外部記憶が溢れそうになると，そのノードのプロセッシングコアに対して割込みをかけられるようにする．退避用外部記憶が溢れる場合は，アドミッションコントロールを行ってパケットの破棄を行ったり，送信元に送信データの一時停止を行ったりするように制御する等の方法が考えられるが，そのプロトコル自身は*Responsive Link*の規格では定めていない．それらは上位のプロトコルで行うことになるので，上記割込みをかける閾値を設定可能にするように設計する（実際の応用現場においては，SDRAMに退避されること自体，ほとんどなかった）．

リアルタイム通信を実現するために，優先度によるパケットの追い越しをこのように再送を行わななくてよいように設計する．

3.4 ルーティング

*Responsive Link*の経路制御は，図7に示すようなルーティングテーブル（経路制御表）を設定することによって行う．ルーティングテーブルは，*Responsive Link*コントローラ内に置き，そのノードのローカルなプロセッサから読み書きできるようにする．図7において，Reference部はパケットのヘッダと同一であり，Referent部に当該パケットに関する設定を行う．EEビットおよびDEビットは，それぞれそのラインがイベントリンク用の設定かデータリンク用の設定かを示す．両方とも設定されていれば，両リンクとも同様の設定になる．L[4-0]は，前述のリンクストローブビットであり，出力ポート（複数可）を指し示す．

ルーティングテーブルの大きさ（エントリ数）は実装依存で有限となるため，非常に大きな分散システムを構築する際には溢れてしまう可能性がある．ルーティングテーブルに入りきらない大規模なシステムを構築する際には，ローカルノードプロセッサの主記憶上に完全なルーティングテーブルを用意し，*Responsive Link*コントローラ上のルーティングテーブルはキャッシュとして用いるようにする．つまり，TLB付きのMMUとページテーブルを用いたメモリ管理と同様な管理手法を行うようにする．

そのために，ルーティングテーブルにヒットしない

エントリがあった際には，ローカルノードのプロセッサに対して割込みをかけると同時に，該当パケットを一時的に前述の退避用外部記憶に退避する．割込みをかけられたプロセッサは，主記憶上の完全なルーティングテーブルをソフトウォークしてエントリを検索し，そのエントリを *Responsive Link* コントローラ上のルーティングテーブルの適切なエントリとスワップするようにする（多くの場合，最近使用されていないエントリとスワップすると考えられるが，それは RT-OS のポリシ依存である）．*Responsive Link* コントローラ側は，イベントリンクとデータリンクそれぞれについて，LRU エントリアドレスが分かるように設計し，RT-OS に対してヒントを与えるようにする．その後，退避していたパケットを追い越しバッファに書き戻すことによって継続的にルーティングを実現する．

上記のような機構により，大規模な分散リアルタイムシステムが構築可能である．ただし，コントローラ内のルーティングテーブルに収まる範囲の規模でないと，厳密にハードリアルタイム性を維持するのは困難となる．

また，分散リアルタイムシステムの規模が大きくなればなるほど（つまりルーティングテーブルのサイズが大きくなればなるほど）通信のジッタは大きくなり，リアルタイム性の時間粒度も大きくなるが，近傍で激しく通信している経路をキャッシュに置き，そうでないものは主記憶上のルーティングテーブルに置く等の方法をとることにより，運用が可能であると考えられる．

3.5 パケットの加減速制御

リアルタイム通信パケットの制御を外部から行うことができるようにするために，通信ノードごとにパケットの優先度の付け替えができるようにして，分散管理型でのリアルタイム通信の制御を実現する[7]．

優先度の付け替えは，図 7 のルーティングテーブルを用いることによって行う．図 7 において，ネットワークアドレスと優先度を元にルーティングテーブルを参照し出力ポート番号を決定する際に，優先度を付け替えないモード（図 7 の優先度付替ビット PE が無効）の場合は優先度はそのままであるが，優先度を付け替えるモード（優先度付替ビット PE が有効）の場合，出力ポートから出力する際に優先度（Priority[7-0]）を新優先度（P7〜P0）に置き換える．つまり，現ノードでの通信パケットの優先度は入力パケットのヘッダに付加されている優先度で決定され，その優先度に従って追い抜きやルーティングが決定されるが，次ノード以降での通信パケットの優先度を制御することができる．ルーティングテーブルの設定はソフトウェア（分散リアルタイムオペレーティングシステム等）で行い，ルーティング（経路制御）自身はハードウェアで行うようにする．

このパケットの加減速制御機構により，たとえば，リアルタイム通信の流量やレイテンシを監視するミドルウェアを用いて，リアルタイム通信の制御を可能とする．リアルタイム性の低い通信パケットがバルク的に流れていて，そのパケットが他のリアルタイム性の高いパケットの通信のリアルタイム性を阻害していたとしたら，通信監視ミドルウェアが当該パケットの優先度を下げることによって，リアルタイム性の制御を行うことができる．あるいは，あるノードでデッドラインミスが発生してしまった場合，その通信パケットの優先度を途中の経路で上げることにより（特にホットスポットで優先度を上げると効果的），次回からのデッドラインミスを防ぐことが可能となる．

3.6 優先度に従った経路制御

優先度に従って専用回線や迂回路を設けたり，データの流量の制御を行ったりすることができるように，まったく同じネットワークアドレスを持つ通信パケットの経路を優先度によって別の経路に設定することができるようにする[8]．そのために，基本的にはネットワークアドレスと優先度の組でルーティングテーブルを参照するようにする．

優先度ごとに必ずルーティングテーブルを設定しなければならないと煩雑であるので，デフォルトルートを設ける．ネットワークアドレスは同じであるが優先度が一致する組合せ（経路）がルーティングテーブル上にない場合には，最も優先度の低い優先度 0 の経路がデフォルト経路となるようにする．つまり，

(1) ネットワークアドレスと優先度の両方が一致すればその経路が第 1 優先，
(2) ネットワークアドレスは一致するが優先度が一致しない場合，優先度 0 の経路，

となる．ここで，優先度 0 の経路はデフォルト経路となるので，途中で経路が消滅してしまわないようにルーティングテーブルに必ず登録する必要がある．

図 8 は，2 次元格子の交点に通信ノードがあるとし，まったく同じ送信元から送信先に対して異なる優先度の通信パケットを同時に通信している状態を示す．たとえば，優先度 0 のイベントリンクの経路上は別の通信ノードからの通信パケットも同じ経路を通って送信先に行くように設定しておき，優先度 3 の経路は送信元と送信先の優先度 3 の通信パケットしか通らないように設定しておくことにより，他の通信パケットと衝突が起きない専用回線を実現することができる．*Re-*

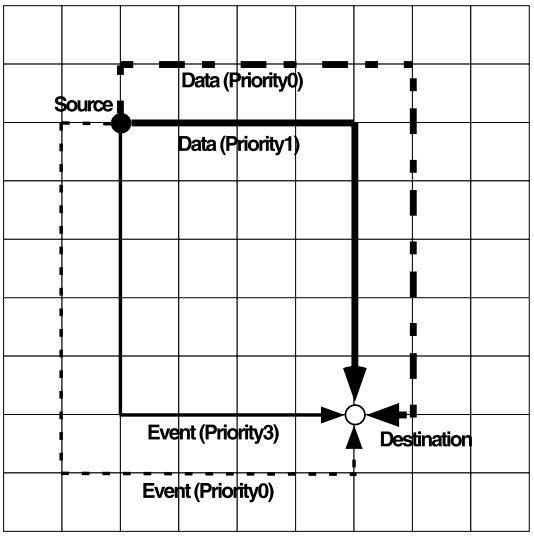

図 8 *Responsive Link* の優先度付経路
Fig. 8 Prioritized routing of *Responsive Link*.

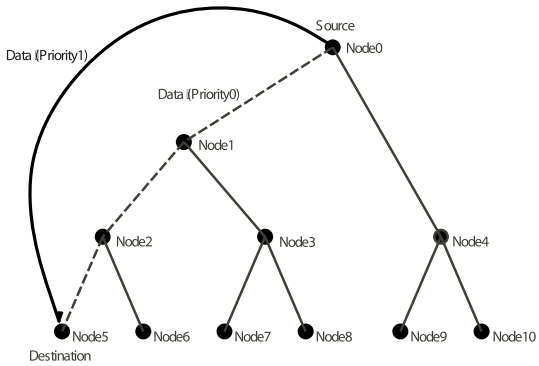

図 9 *Responsive Link* の優先度付木構造経路
Fig. 9 Prioritized tree routing of *Responsive Link*.

sponsive Link には優先度による追い越し機構があるが，衝突があると追い越しのために多少のオーバヘッドが生じてしまうので，このように優先度を用いてパケットの衝突がまったくない専用回線を設定することにより，レイテンシおよびジッタが非常に小さいリアルタイム経路の実現を可能とする．また，優先度が異なる経路を複数設定することによってマルチリンクを実現し，バンド幅を広げることも同時に可能とする．

制御用の分散システムでは**図 9** のような木構造をとる場合が多い．図 9 において通信ノード 0 から通信ノード 5 に通信する場合，優先度 0 の通信パケットは途中に通信ノード 1 と通信ノード 2 という中間ノードを経由して通信を行うが，優先度 1 の通信パケットは通信ノード 0 から直接通信ノード 5 へ通信を行うことができる．これは，たとえばヒューマノイドロボットを開発した際に，当初は頭モジュール，肩モジュール，肘モジュール，指モジュールと接続しそれらの経路をホップして通信を行っていたが，設計後にどうしても頭モジュールと指モジュール間の通信レイテンシが間に合わないと判明した場合，後付けで頭モジュールと指モジュールを直接接続し優先度を変えて通信することにより，容易に通信経路（この場合は専用回線）の増設を可能とする．この機能は，実システムを構築する際に大きな手助けとなる．

3.7 低レベル通信方式

Responsive Link は分散制御用途であるので，必ずエラー訂正を行わなければならない．その際，できるだけエラー訂正によってリアルタイム性が損なわれないようにする必要がある．

ここで，パケット単位で CRC を付加しエラー訂正を行う方法では，パケット全体を受信しないとエラー訂正できない．その場合，ホップごとにレイテンシが積算されていくので，リアルタイム通信用のエラー訂正としては好ましくない．そこで，レスポンシブリンクでは 1 ホップごとにフレーム（図 3 参照）単位でエラー訂正を行い，1 フレーム（8 bit データ＋ 4 bit 冗長符号）につき 1 bit のエラーであれば，再送することなしにハードウェアで誤り訂正を行うようにする．

3.7.1 CODEC

Responsive Link の CODEC は，8 bit の情報ビット列に，誤り訂正用の 4 bit の冗長ビット列を加えた 12 bit を 1 フレームとして通信を行う．本 CODEC で行われる符合化は，以下のような流れとなる．

(1) 巡回組織ハミング符合化（冗長ビット列を加える誤り訂正符合化）
(2) Bit Stuffing（連続した 1 の符合に 0 を挿入）
(3) NRZI 符合化

以下，各符合化[9] について説明を行う．

3.7.2 巡回組織ハミング符号化

誤り訂正符合として，生成多項式が $x^4 + x + 1$ の巡回組織ハミング符合を採用する．この符合化では，8 bit データの下位（LSB）側に 4 bit の冗長ビット列を付加することで，12 bit 中の任意の 1 bit の誤りを受信側で訂正することを可能にし，**表 1** より誤りの位置を特定できる．送信時には，これら 12 bit のビット列は，MSB 側から 1 bit ずつ送信を行う．

3.7.3 Bit Stuffing

1 が長時間連続することによって引き起こされるリンクへの直流成分の発生や，受信側のビット同期への支障を回避するために，通信データ中に 5 つの連続した 1 が現れた場合には，その後ろに 0 を挿入する．

表 1 シンドロームとエラーの位置
Table 1 Syndrome and error position.

Syndrome	Error Position (4 redundancy bits)	Meaning
0000	00000000 0000	No error
0001	00000000 0001	Redundancy-bit error
0010	00000000 0010	Redundancy-bit error
0100	00000000 0100	Redundancy-bit error
1000	00000000 1000	Redundancy-bit error
0011	00000001 0000	0bit error
0110	00000010 0000	1bit error
1100	00000100 0000	2bit error
1011	00001000 0000	3bit error
0101	00010000 0000	4bit error
1010	00100000 0000	5bit error
0111	01000000 0000	6bit error
1110	10000000 0000	7bit error

3.7.4 NRZI 符合化

最終的に送信される際に NRZI（Non Return to Zero Inverted）符合化を行う．NRZI 符合化は，0 を送る場合にはリンクのデータビットを反転し，1 を送る場合にはデータビットの状態を前のまま保持する．

3.7.5 セットアップパターン

電源投入直後や，予期できないバースト的なリンクエラー等の後は，送受信インタフェース間でフレーム同期がとれない場合がある．そのような場合，明示的にリンクの初期化を行うようにする．具体的には，以下に示すセットアップパターンを受信側に送信する．

セットアップパターン：000001111110

このパターンは，連続した 1 が 6 個以上は連続しないという bit stuffing の規則に反しているため，いかなる通常のパケットとも区別される．受信側では，このパターンを受信するとその後，最初に認識したフレームを，新しいパケットの第 1 フレームとして解釈する．

3.7.6 DPLL を用いたビット同期

受信側に DPLL（Digital Phase Lock Loop）機構を設計し，受信用クロックの立ち上がりエッジに同期して受信信号をサンプリングする．1 bit 転送あたりのサンプリング数はソフトウェアの設定によって可変（4，8，16，32，64，128，256）にする．DPLL では設定された周期ごとに受信用クロックを生成し，受信信号のエッジを検出することにより，信号のエッジ間の中央で受信用クロックが立ち上がるように，受信用クロックの周期を微調整を行う．

3.7.7 エラーの取扱い

Responsive Link では，誤り訂正符合化によって 1 [bit/frame] の誤りまでは自動的にエラー訂正を行うことができる．エラーの箇所を受信側で特定するために，図 3 のトレイラ部の Dirty ビットを立てる．具体的には，データリンクの場合ワード（4 byte）単位で，イベントリンクの場合バイト単位で，エラーのあった場所の Dirty ビットを立てる．エラーがハードウェアによって訂正されても，訂正しきれなくても Dirty ビットは立てるようにする．また，そのパケット中に 1 カ所でもエラー訂正が行われハードウェアで訂正しきれた場合，トレイラ部の Correct ビットを立てる．エラー訂正不可能だった場合，Fatal ビットを立てる．受信側のアプリケーションでは，これらを参考にし，たとえば，受信データを本当に制御に使用してよいかどうか等を判断することを可能にする．

表 2 通信速度とケーブル
Table 2 Communication speed vs. cable.

Speed (Mbaud)	100	200	400	800
Maximum Length (m)	100	80	60	40
Recommendable Cable	Cat5e	Cat5e	Cat6	Cat6

3.7.8 通 信 速 度

Responsive Link の通信（変調）速度は，様々な環境（コンフィギュレーション，アプリケーション）を想定し，800，400，200，100，50，12.5，6.25 [Mbaud] の範囲で段階的に可変とする．

表 2 に，通信速度と最大通信距離，推奨ケーブルの関係を示す．たとえば，最大変調速度 800 [Mbaud] で通信する場合，ケーブルには Category6 を使用し，最大通信距離は 40 [m] 以内である．この場合，DPLL の基準周波数には 3.2 [GHz] を使用するか，デューティ比が 1 対 1 の 1.6 [GHz] のアップダウンエッジ（もしくは 2 相クロック）を使用し，サンプリング数 4 で DPLL を行うことによって実現する．

レスポンシブプロセッサは組み込み用途を想定しているので，消費電力が大きな問題となる．一般に通信速度（動作周波数）を速くすれば消費電力が大きくなり，遅くすれば小さくなる．通信速度の変更は，受信クロックを変更するのではなく DPLL のサンプリング数を変更することによって行う．したがって，通信速度が遅い場合の通信は，通信速度が遅くなることによる安定性の増加と DPLL のサンプリング数が増加することによる安定性の増加という 2 重の恩恵を受ける．

4. 実装および評価

Responsive Link は，現在までに *Responsive Processor*[1] および *Responsive MultiThreaded* (*RMT*) *Processor*[10),11)] という 2 種類のチップ上に実装を行っている．

4.1 *Responsive Processor*

Responsive Processor 上に実装した *Responsive Link* はサブセットであり，フル規格の *Responsive Link* の下位互換性を有している．*Responsive Processor* 上の *Responsive Link* の片方向の最大変調速度は 100 [Mbaud] までであり，優先度も 4 レベル（2 bit）となっている．

Responsive Processor は，主にヒューマノイドロボットの分散制御を目的として研究開発されており，ロボット体内に 30 ノード程度が主に木構造で接続されることを想定して設計を行っている．そのために，*Responsive Processor* には，5 組（5×5 のネットワークスイッチ）の *Responsive Link* を設計・実装している．そのうち 1 組はプロセッサに接続し，4 組がチップ外部に出ている．追い越し用バッファをデータリンク，イベントリンクともに 8 パケット分ずつ実装し，退避用外部記憶は外付けの SDRAM を用いて最大 16 [MB] 実装可能である．

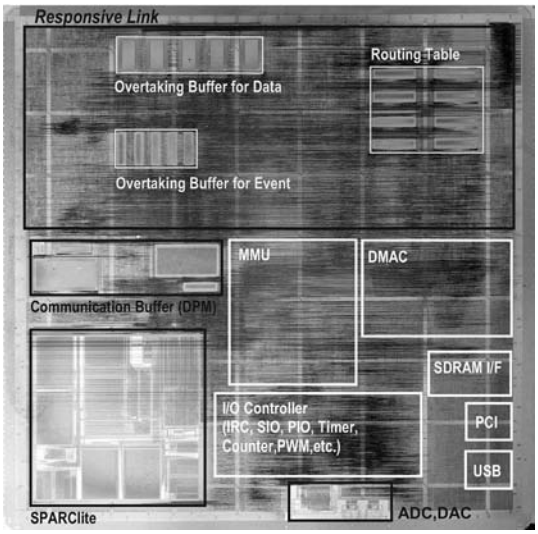

図 10 *Responsive Processor* のレイアウト
Fig. 10 Layout of *Responsive Processor*.

上記の目的から同様に，ルーティングテーブルには 256 エントリを実装している．ルーティングテーブルの実装には連想メモリ（CAM）を使用せず，デュアルポートメモリ（DPM）を並列に配置しコンパレータと組み合わせて実装している．CAM は一般に大きな面積と消費電力を消費してしまい，レイテンシも大きい．また，プロセスによっては CAM をサポートしていない場合もある．したがって，今後の実装のリファレンスとなる IP として，CAM を使用しないで実装を試みている．具体的には，32 エントリ分を格納できる DPM を 8 個配置し，それらを並列にアクセスするように実装している．ネットワークアドレスが一致するエントリを発見した場合，同時に優先度を比較し，完全一致した場合には，その時点でテーブル引きを終了する．優先度 0（デフォルトルート）と一致した場合には，そのエントリを候補として検索を続ける．その後，優先度まで完全一致するエントリがあれば，そのエントリを返し，最後まで優先度が完全一致しなければ，優先度 0 のエントリ（デフォルトルート）を返すように実装している．DPM なので計 16 ポートを有しており，256 エントリのルーティングテーブルに対して，平均 8 クロック，最大 16 クロックで 1 エントリのテーブル引きをすることができる．ここで，内部の追い越し機能付きネットワークスイッチは 8 bit 単位でスイッチングを行っており（3.3 節参照），一方，シリアル通信部では CODEC により 12〜14 クロックで 1 byte（8 bit）転送できることから，テーブル引きに 12 クロックかかったとしても，その遅れはスイッチ部では 1 クロック分にしか相当しない．また，ルーティングテーブルのデータはイベントリンク，データリンクの全入力ポートで共有されている．イベントリンクとデータリンクでそれぞれ 1 系統（計 2 系統）のコンパレータを有しており，イベントリンクとデータリンクで同時にテーブル引きをすることができる．同種のリンクで複数同時にテーブル引き要求がある際には，アービトレーションを行って排他的に使用するように実装している．

図 10 に *Responsive Processor* のレイアウトを示す．左下角の大きな正方形部分が SPARC プロセッサ（約 300 [k gates]）を示す．右上角の 8 個の長方形の集合が *Responsive Link* のルーティングテーブルであり，左上の 2 ブロック（各 5 個）の長方形の集合が *Responsive Link* のパケット追い越し用バッファである．大きい方がデータリンク用で小さい方がイベントリンク用である．ルーティングテーブルと追い越し用バッファに挟まれた領域がレスポンシブリンク部分であり，約 400 [k gates] ある．デザインルールは，以下のとおりである．

- プロセスルール：0.35μm CMOS 4 層メタル
- 総ゲート数：2,378 [k gates]
- ダイサイズ：14.5 [mm]×14.5 [mm]=210 [mm^2]
- パッケージ：416 ピン BGA（40 [mm]×40 [mm]）
- 電圧：3.3 [V]
- 最大消費電力：2 [W]

表 3 に，通信速度と消費電力の関係を示す．通信速度は動作中に動的に変更可能である．変調速度が

表 3 *Responsive Link* の速度と消費電力の関係
Table 3 Speed vs. power on *Responsive Link*.

Modulation (Mbaud)	100	50	25	12.5	6.25
Data Speed (Mbps)	67	33	17	8	4
Event Latency(μsec)	3.1	6.2	12.5	25.0	50.5
Power (W)	0.2	0.1	0.05	0.02	0.01

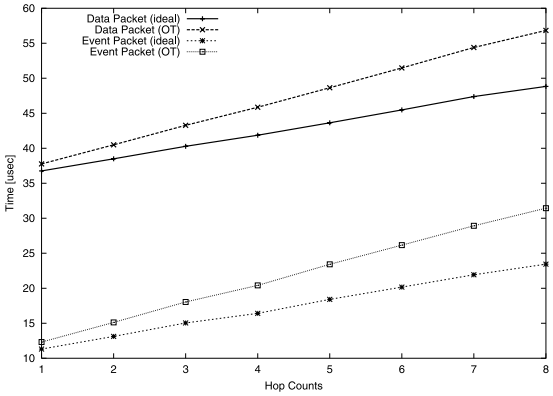

図 11 *Responsive Link* のホップ数とレイテンシの関係
Fig. 11 Hop counts vs. latency of *Responsive Link*.

100 [Mbaud] の際のイベント 1 パケット（ペイロード：8 byte）のレイテンシは，低優先度パケットがつねに経路上を流れている際に，高優先度パケットが追い越しを行う場合，

$$\textit{Latency of Event} = 2.1\,[\mu\text{sec}] + 1.0\,[\mu\text{sec}] \times n\,[\text{hops}]$$

で表すことができる．送受信のオーバヘッドが 2.1 [μsec] かかり，1 [hop] あたりルーティングテーブル参照や追い越し等のすべてのオーバヘッドを含めて 1.0 [μsec] かかる[12]．現在の代表的な高速ネットワークである Gigabit Ethernet や Myrinet[13] の 4 [byte] レイテンシが 10 [μsec] 弱であることから，イベントリンクは非常に低いレイテンシを実現していることが分かる．

図 11 に，*Responsive Processor* のアプリケーションで最も使用頻度の高い 25 [Mbaud] に変調速度を設定した際の *Responsive Link* のホップ数とレイテンシの評価を示す．イベントリンク，データリンクともに無負荷の場合（ideal）と，ノードごとの同一経路上に必ず低優先度パケットの通信が連続してあるという最悪ケース（OT）について測定を行った．評価には *Responsive Processor* 上の RT-OS である RT-Frontier[14] を用いており，OS のオーバヘッドを含んでいる．高優先度パケットのレイテンシは，この図において無負荷のライン（ideal）と追い越しのライン（OT）に挟まれた領域にバウンドされることになる．

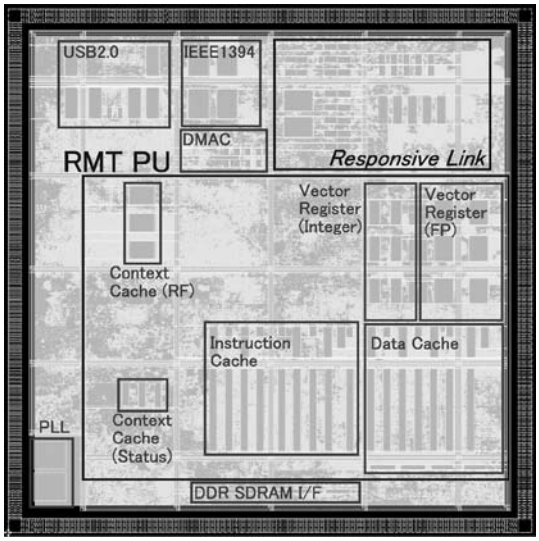

図 12 *RMT Processor* のレイアウト
Fig. 12 Layout of *RMT Processor*.

4.2 *RMT Processor*

RMT Processor 上に実装した *Responsive Link* は規格をさらに拡張しており，フル規格の *Responsive Link* の上位互換性を有している．*RMT Processor* 上の *Responsive Link* の片方向の最大変調速度は 800 [Mbaud] である．追い越し用バッファをデータリンク，イベントリンクともに 8 パケット分ずつ実装し，退避用外部記憶は外付けの DDR SDRAM を用いて最大 256 [MB] 実装可能である．ルーティングテーブルは 256 エントリ実装している．

RMT Processor は，分散リアルタイム処理を実現するために以下の機能をすべて 1 チップに集積（System-on-a-chip）している．

- リアルタイム演算機能（RMT Processing Unit：優先度付き 8way SMT，コンテキストキャッシュ，整数・浮動小数点ベクトルユニット等）
- リアルタイム通信機能（*Responsive Link*）
- コンピュータ用周辺機能（PCI64，USB2.0，IEEE1394，DDR SDRAM I/F，DMAC 等）
- 各種周辺制御機能（PWM 発生器，パルスカウンタ等）

図 12 に *RMT Processor* のレイアウトを示す．中

央から右下にかけての大きな正方形の部分が RMT プロセッシングユニット（約 6 [M gates]）を示し，右上の長方形の部分が *Responsive Link* である．シリアル通信を 4 本束ねたパラレル通信によるマルチリンクが可能なように規格を拡張しており，サイズは約 600 [k gates] ある．デザインルールは，以下のとおりである．

- プロセスルール：0.13μm CMOS 8 層 Cu 配線
- 総ゲート数：14 [M gates]
- ダイサイズ：10.0 [mm]×10.0 [mm]=100 [mm^2]
- パッケージ：BGA（40 [mm]×40 [mm]）
- 電圧：Core 1.0 [V]，I/O 2.5 [V]
- 最大消費電力：8 [W]

5. 標 準 化

Responsive Link は，国内においては情報処理学会試行標準 WG6（メンバ：産総研，慶大，九大，三菱電機，松下，東芝，富士通，日立）において標準化作業を行っており，2003 年 9 月に IPSJ-TS 0006:2003 として公開されている[2]．また，国際的には，ISO/IEC JTC1 SC25 WG4 において国際標準化を行っている[3]．標準化することにより，機械システムや組み込みシステムの電子部を構築する際の共通プラットホームの実現を目指しており，性質の異なるシステムの相互接続を潜在的に可能にする．

6. ま と め

本論文では，分散制御用リアルタイム通信規格である *Responsive Link* の通信アーキテクチャおよび設計の詳細について述べた．*Responsive Link* は，ソフトリアルタイム通信とハードリアルタイム通信の分離，パケットに優先度を付加しノードごとに高優先度パケットが低優先度パケットの追い越し，パケットの優先度が異なると優先度ごとに別経路を設定して専用回線や迂回路を実現可能，ノードごとに優先度を付け替えることができ分散管理型でパケットの加減速を制御可能，ハードウェアによるエラー訂正，通信速度を動的に変更可能，トポロジーフリー，Hot-Plug&Play 等の様々な特徴を有し，柔軟なリアルタイム通信を実現する．日本発のリアルタイム通信規格として，広く使用されることを期待している．

謝辞 *Responsive Link* の一部は科学技術振興事業団 PRESTO および SORST における研究成果です．

Responsive Processor の研究開発にあたり，国際標準創成型研究開発制度および電子技術総合研究所 COE（当時）の予算を使用させていただきました．その際，現産総研大蒔和仁情報処理研究部門長に大変お世話になりました．*RMT Processor* の研究開発にあたり，文部科学省科学技術振興調整費の予算を使用させていただきました．その際，安西祐一郎慶應義塾長に大変お世話になりました．

また，*Responsive Link* の標準化にあたり，情報処理学会試行標準 WG6，情報処理学会試行標準委員会，および ISO/IEC JTC1 SC25 WG4 *Responsive Link* SG の各委員に感謝いたします．特に産総研松井俊浩 DHRC センター長代理には大変お世話になりました．ここに，深く謝意を表します．

参 考 文 献

1) 山崎信行，松井俊浩：並列分散リアルタイム制御用レスポンシブプロセッサ，日本ロボット学会誌，Vol.19, No.3, pp.68–77 (2001).
2) http://www.itscj.ipsj.or.jp/ipsj-ts/index.html
3) http://www.itscj.ipsj.or.jp/committees/sc25/sc25.html
4) 戸田賢二，西田健二，高橋栄一，山口喜教：優先度先送り方式による実時間相互結合網ルータチップの実現と性能，情報処理学会論文誌，Vol.36, No.7, pp.1619–1629 (1995).
5) 西田健二，戸田賢二，高橋栄一，山口喜教：実時間用並列計算機 CODA のプロセッサアーキテクチャ，電子情報通信学会論文誌，Vol.J78–D–I, No.8, pp.777–787 (1995).
6) Liu, J.W.S.: *Real-Time Systems*, pp. 159–179, Prentice Hall (2000).
7) 山崎信行：分散管理型通信方法及び装置，特許第 3460080，特願平 11-343139 (2003).
8) 山崎信行：分散管理型通信方法及び装置，特開 2002-330162 (2002).
9) Halsall, F.: *Data Communications, Computer Networks, and Open Systems*, Forth Edition, Addison-Wesley Publishing Company (1995).
10) 山崎信行，堀 俊夫：分散リアルタイムネットワーク用プロセッサとその応用，情報処理，Vol.44, No.1, pp.6–13 (2003).
11) 伊藤 務，内山真郷，佐藤純一，薄井弘之，松浦克彦，山崎信行：Responsive Multithreaded Processor の設計・実装，情報処理学会研究報告 2003-ARC-145，pp.31–36 (2003).
12) 奥埜貢士，河野通宗，山崎信行，安西祐一郎：実時間 OS μ–PULSER への実時間イベント伝達機構の設計と実装，電子情報通信学会技術研究報告，Vol.98, No.687, pp.47–54 (1999).
13) http://www.myrinet.com/
14) Kobayashi, H. and Yamasaki, N.: Scheduling Imprecise Computations with Wind-up Parts, *19th International Conference on Computers*

and Their Applications, pp.232–235 (2003).

(平成 15 年 7 月 31 日受付)
(平成 15 年 11 月 1 日採録)

山﨑 信行（正会員）

1966 年 5 月 1 日生．1991 年慶應義塾大学理工学部物理学科卒業．1996 年同大学院理工学研究科計算機科学専攻博士課程修了．博士（工学）．同年電子技術総合研究所入所．1998 年 10 月慶應義塾大学理工学部情報工学科助手．2000 年 4 月同専任講師．1997 年〜2000 年科学技術振興事業団さきがけ研究 21 研究員．現在，産業技術総合研究所特別研究員を兼務．並列分散処理，リアルタイムシステム，ロボティクス，システム LSI 等の研究に従事．日本ロボット学会，IEEE 各会員．