中密度実装クラスタにおけるバリア同期情報管理方式

中密度実装クラスタにおけるバリア同期情報管理方式

早川　潔*

Managing Method of Barrier Synchronization Value on a Middle Density Cluster

Kiyoshi Hayakawa

ABSTRACT

本稿では，SCMDクラスタにおけるバリア同期情報管理方式および性能評価について述べる．SCMDクラスタは，1筐体(1シャーシ）に3台のPCが実装され，そのシャーシが10シャーシ（ノード数は30）搭載されたクラスタであり，シャーシ内ネットワークおよびシャーシ間ネットワークを搭載している．シャーシ内ネットワーク（SCICネットワーク）は，シャーシ内ノード間をより密に結合し，シャーシ間ネットワークは，とりまわしがよくスケーラビリティの高い汎用ネットワークでシャーシ外ノード間を結合する．本研究では，SCICネットワークを使用したシャーシ内ノードの同期処理を行うハードウェア同期処理ユニットにおける新たな同期情報管理方式を提案する．その同期情報管理方式を採用した同期ユニットをSCMDクラスタの一部のノードに実装し，評価した．その結果，シャーシ内ノード間のバリア同期処理を3.86μsで処理できた．

Key Words: computer cluster, collective communication, barrier, FPGA

1　はじめに

近年，汎用CPUの低消費電力化の傾向にあり，そのCPUを使用したPCクラスタの低消費電力化における研究も行われている[1][2][3][16]．また，低消費電力CPUを使用することにより，CPUファンなどの冷却装置が小型化され（または省かれ），より高密度に実装可能になってきている[1][2][3]．

高密度実装クラスタとして，Green Destiny[1]というクラスタが知られている．Green Destinyは，19インチラックに42シャーシ(本稿では，複数のノードが収められている筐体．Green Destinyでは「ユニット」と呼ばれている）搭載されている．1つのシャーシに

24ノード搭載している．そのネットワーク構成は，100Base/TXおよびGigabit Ethernetが採用されている．各プロセッサは100Base/TXのSwitchで結合され，さらにそのSwitchがGigabit Ethernet Switchに結合される構成となっている．

国内においても，MegaProto[2]と呼ばれる低消費電力および高密度実装クラスタが開発されている．1シャーシあたり16個の低電力プロセッサ（Transmeta社のEfficeon）が搭載されている．ネットワーク構成は，ユニット内に独立した2系統のGigabit-Ethernetのスイッチを用意し，ノードにも2系統のGigabite-Ethernetポートを用意し，それぞれを各スイッチに接続する形をとる．

いずれのクラスタも1シャーシに20台前後のノードが搭載されている．本研究で開発しているSCMDクラスタの1シャーシあたりのノード数は3台程度なので，SCMDクラスタを「中密度実装クラスタ」と位置づける．SCMDクラスタにおいてもできるだけ低消費電力・高密度を目指しているが，それと同時に長期間運用可能なクラスタを目指している．

近年のCPUの開発サイクルは急激に速くなり，2，3年もすれば入手が困難になってきている．そのような状況の中で，故障箇所の補充などが難しくなっている．そこで，SCMDクラスタでは，FAで使用するPICMG仕様のマザーボードを採用し，そのマザーボードにインテルが供給しているEmbedded CPUを搭載することを考えている．インテルのEmbedded CPUを採用することにより，安定してCPUを調達できる．PICMGのマザーボードにもPentium Mが搭載できる製品がでてきており，より低消費電力なクラスタが実現可能になってきている．

1シャーシに複数台数のノードが搭載されているクラスタの場合，ネットワークをシャーシ内とシャーシ外に分けて，構成したほうが，より効率よいデータ転送ができる．つまり，つまり，シャーシ間ネットワーク(Outer-chassis network)は，Ethernetなどのシャーシ間

2006年4月12日　受理
* 総合工学システム学科　電子情報コース
(Dept of Industrial Systems Engineering : Electrical Engineering and Computer Science Course)

接続でノイズが入りにくいような処理が施されているネットワークを採用し，シャーシ内ネットワーク（Inner-chassis network）では信号線を直接結合させるような比較的に密なネットワーク(例えば，バス結合)を採用する．シャーシ間では，グランドループなどのノイズの原因となる現象が生じやすいので，パルストランスを用いたアイソレーションを行う必要がある．一方，シャーシ内ではそのような現象が起こりにくく，ノイズ対策を施す必要がほとんどない．よって，シャーシ内接続におけるノイズ対策としては，ダイオード（または抵抗）を使ったターミネーションを施す程度でよい．また，シャーシ間では，シャーシ内よりケーブル長が長くなるので，パラレル転送などは向いていないが，シャーシ内はケーブル長が短いのでパラレル転送がしやすい．このシャーシ内ネットワークを利用して，バリア同期などの集合通信処理を高速に実行できる．

そこで，本稿では，シャーシ内ネットワークを使用したハードウェア同期ユニットについて述べる．同期ユニットにおけるバリア同期変数管理方式を提案し，バリア同期の処理速度をシミュレーションおよび実機で測定し，バリア同期の処理速度削減効果について検証する．また，シャーシ間ネットワークを合わせたクラスタ全体のバリア同期の処理時間を見積ることにより，クラスタ全体のバリア同期処理時間の削減効果も検証する．

2　SCMD クラスタ

中密度実装クラスタとして，SCMD クラスタシステムを構築した（図 1 参照）．SCMD クラスタシステムは PC ボード（PICMG 規格のボード CPU：PentiumIII600MHz）を 100Base/TX の Ethernet および SCIC ネットワークで結合したクラスタシステムである．100Base/TX の Ethernet 接続をシャーシ間ネットワークとし，SCIC ネットワークをシャーシ内ネットワークとする（2 つのネットワークをまとめて「SCMD ネットワーク」とする）．本クラスタでは，1 シャーシあたり 3CPU ボードが実装されている．各 CPU ボードの PCI スロットに SCIC ネットワークボートと呼ばれる PCI ネットワークボードを実装する．

3　シャーシ内バリア同期処理

SCIC ネットワークを利用したシャーシ内バリア同期（以後「SCIC_Barrier」とする）を行う同期ユニットについて述べる．

3.1　SCIC ネットワークボード

SCIC ネットワークボードは，Control Chip およびシャーシ内ネットワークコネクタ（Link_A　および Link_B）のみで構成されている．Control Chip は，同期・通信処理をハードウェアで実装することにより，高速化を実現する．シャーシ内ネットワークコネクタは 20 ピンのパラレルケーブル用のコネクタであり，パラレルケーブルを用いて Link_A と Link_B を接続する(図 4 参照)．

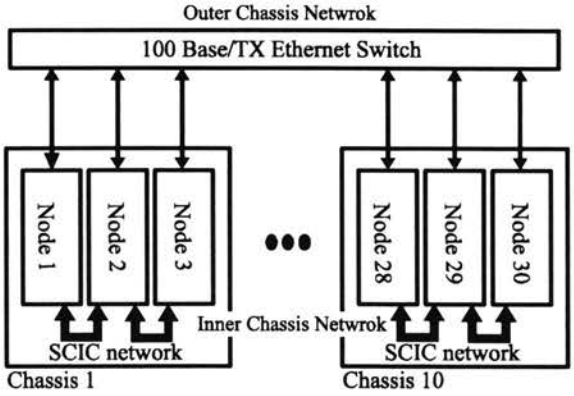

図 1. SCMD クラスタ

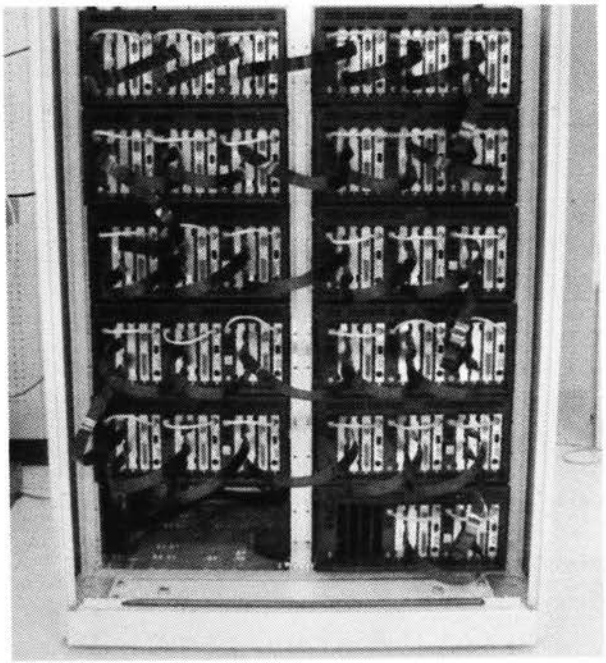

図 2. SCMD クラスタの写真

SCIC ネットワークボードは，本来，メッセージパッシング型の通信処理および拡張型バリア同期処理を低レイテンシで実現するために開発された PCI ボードである[5]．しかし，今回，この SCIC ネットワークボードをシャーシ内ネットワークに特化した PCI ボートとして開発する．

3.2　SCIC Control Chip

図 5 に Control Chip 内のブロック図を示す.

Control Chip は，PCI-Wishbone ブリッジ，同期制御部，通信制御部，送受信パケット処理部，および共有メモリで構成される.

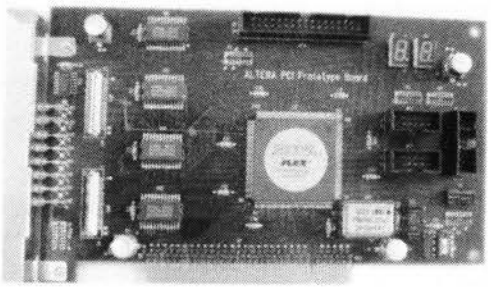

図 3. SCC ボード

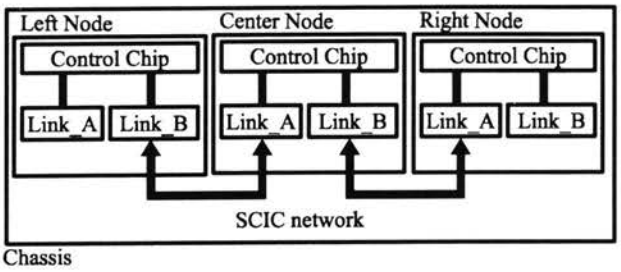

図 4. SCIC ネットワークを利用したノード間接続

PCI-Wishbone ブリッジは，PCI バスプロトコルと Wishbone バスプロトコルをインターフェースし，PCI のターゲット機能をサポートする．同期制御部は，SCIC ネットワークを使用した同期処理アクセスを行い，任意参加バリア，Fuzzy バリアの同期を処理する．通信制御部は，パケットデータ処理を行う．送受信パケット処理部は，通信モードを解析し，パケットを同期パケットと通信パケットに分けてパケット送受信処理を行う．共有メモリには，各ブロック間が連携して処理する場合に必要なデータを格納する.

本稿では，同期ユニットの設計・実装について述べるので，それに関連する「パケット送受信処理部」および「同期制御部」についてのみ言及する.

3.3　Control Chip 内でのブロック間接続

Control Chip の各ブロックは，WISHBONE[15]を用いて接続する．WISHBONE は，Open Cores で公開されているオープンソースの IP コア間インターフェースバスである．WISHBONE を用いることにより，比較的簡単に各ブロック間を接続できる.

WISHBONE の規格には，Point to Point 接続・クロスバスイッチ接続などの接続形態があるが，本 Chip では 4x4 Shared Bus Inter Connection 接続を用いる．このバスにおけるマスタ IP コアとして，PCI-WISHBONE ブリッジを設定し，スレーブ IP コアとして，同期制御部・通信制御部・疑似グローバルクロック部・共有メモリを設定する.

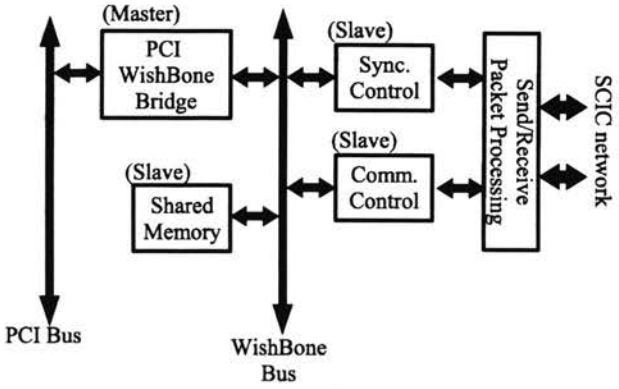

図 5. SCIC Control Chip の構成

3.4　パケット送受信処理部

図 6 にパケット送受信処理部の構成を示す.

パケット送受信処理部では，8 ビットの全 2 重の双方向通信を行う．その際，同期・通信の種類別にモード（同期モード・通信モード）を設け，モードごとに効率のよい通信方式を採用する.

SCIC ネットワークのパラレルケーブルは，20 本のパラレルケーブルなので，8 本をデータ線に使い，残りの 2 本をモード線に使用する．通信手順としては，モード線にモードをある一定期間送出し，その後データを送出するという手順で行われる.

もし，バッファに貯められなくなったら，上位階層でフロー制御 Inner Unit Connect の信号は，全てパケット送受信処理部(Send/Receive Packet Processing Block)へ送られる．パケット送受信処理部は，さらに送信部(Send Block)および受信部(Receive Block)に分かれる.

送信部では，パケットバッファからデータを取出し，そのデータにモード信号を付加して Inner Unit Connect を介して送信する.

受信部では，モード信号を解析して，モードに対応した制御部の受信パケットバッファへ受信パケットを格納する．受信パケットが Buffer に 1 つ以上格納されている場合，Receive Ack 信号を送出し，同期・通信処理部へパケットが到着していることを通知する.

Mode Select(in)と Mode Select(out)の間で簡単なフロー制御を行えるようにする．Buffer が FULL（またはそれに近い状態）のときに，Mode Select(in)にデータが届いてしまった場合，Mode Select(in) は Mode

Select(out)に対して，send stop 信号を送出する．send stop 信号を受け取った Mode Select(out)は，パケット送信を一時停止するコントロールパケットを生成し，相手ノードに送信する．そのコントロールパケットは相手ノードの Mode Select(in)に受け取られる．Mode Select(in)は stop 信号を Mode Select(out)へ送出する．stop 信号を受け取った Mode Select(out)は Start 信号が送出されるまで，パケットの送出を中断する．

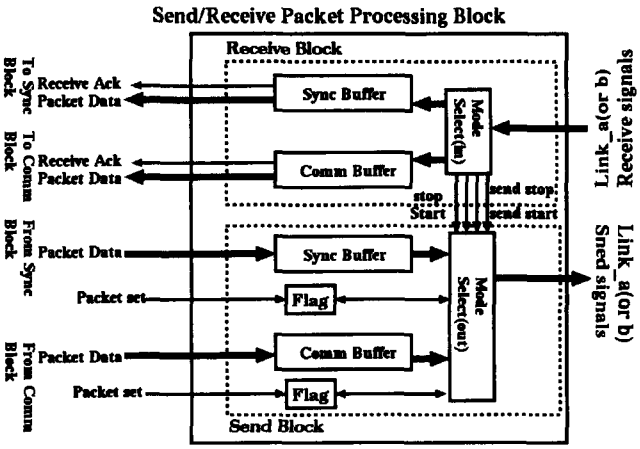

図 6. パケット送受信処理部の構成

3.5　同期制御部の構成

図 7 に SCIC_Barrier を処理する同期制御部の構成を示す．

同期制御部は，Wishbone インターフェース，パケット解析&データ送出部，同期メモリおよび同期検出部で構成される．

同期メモリの構成は，Center ノードと Right または Left ノードの間で異った形をとる．Center ノードの同期メモリは，バリアフラグ(図では B-Flag)，バリアマスク(図では B-Mask) およびバリア到達情報(図では B-Reach)で構成される．一方，Right および Left の同期メモリは，バリアフラグのみで構成される．また，同期検出部は，Center ノードのみに実装される．バリアフラグおよびバリアマスクは 3 ビットの変数であり，各ビットがシャーシ内ノードに対応している．

同期パケットには，同期到達パケットおよび同期成立パケットと呼ぶ 2 種類のパケットを用意する．

同期到達パケットは，Ceneter ノードへ同期成立を知らせるパケットであり，同期成立パケットは，Center ノードが Right および Left ノードへ同期の成立を知らせるパケットである．同期パケットはパケット解析&データ送出部によって，自動的に生成される．

バリア同期を行う際に，Wishbone 経由で同期番号および同期参加情報が書き込まれ，その書き込まれた同期番号のバリアフラグをアクセスし，必要に応じて，同期パケットを自動的に生成する．

つまり，ノードプロセッサが PCI および Wishbone を介して，バリアフラグをセットしたときに，パケット解析&データ送出部が同期到達パケットを自動生成する．

また，バリアが成立した場合，同期検出部がパケット解析&データ送出部に同期成立パケット生成を依頼し，その依頼を受けたパケット解析&データ送出部は，バリア成立パケットを自動生成する．

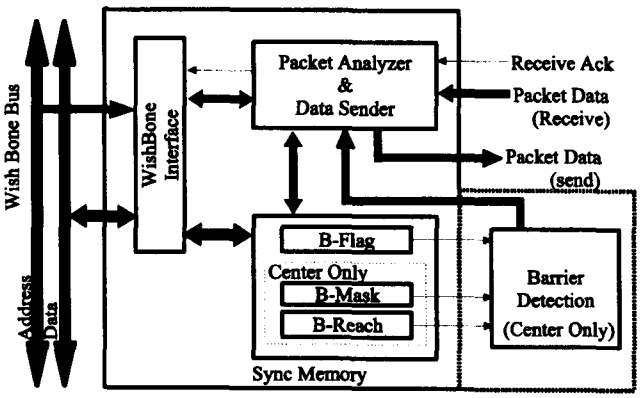

図 7. 同期制御部の構成

3.6　同期情報の管理方法

同期メモリの中にある同期変数は変数ごとに管理方法が異なる．バリアフラグは各ノードで分散管理し，バリアマスクおよびバリア到達情報は Center ノードが集中管理する．バリアフラグは同じ同期番号のバリアフラグが各ノードに配置される．各ノードの同期番号が同じバリアフラグは同期成立パケットによって，coherence が保たれる．バリアマスクおよびバリア到達情報は，Center ノードが集中管理する．Center ノード以外のノードが，バリアマスクおよびバリア到達情報を変更する場合，同期到達パケットを Center ノードに送ることにより，Center ノードがそれらの変数を変更する．

Center ノードにおけるバリア同期手順に伴う同期管理手順を以下に示す．

① Center ノードのプロセッサが同期ポイントに到達したら，特定の番地に同期番号と同期情報（バリアマスクおよびバリアフラグの情報）を書き込む．

② Wishbone インターフェースがそれら書き込みを検知して，その同期番号に対応したバリアフラグをセットし，バリア到達情報を更新し，同期メモリにバリアマスクを保存する．

③ 他のノードから同期到達パケットが届いたら，パケ

ット解析&データ送出部が同期メモリの同期情報を書きこむ.

④ 同期検出部が，書き込まれたバリア情報(Wishbone インターフェースまたはパケット解析&データ送出部が書いた情報) をもとに，同期を検出する.

⑤ 同期検出部がバリア同期完了を検出した場合，パケット解析&データ送出部に同期成立パケット生成を依頼し，バリアフラグをリセットする.

⑥ 同期検出部の依頼にしたがって，パケット解析&データ送出部が同期成立パケットを自動生成し，Left および Right ノードにそのパケットを送る.

LeftおよびRightノードにおけるバリア同期手順に伴う同期管理情報手順を以下に示す.

① Left または Right ノードのプロセッサが同期ポイントに到達したら，特定の番地に同期番号と同期情報を書きこむ.

② Wishbone インターフェースがそれら書き込みを検知して，その同期番号に対応したバリアフラグをセットし，パケット解析&データ送出部へ同期到達パケットを生成するように要求する.

③ パケット解析&データ送出部は Wishbone インターフェースの要求にしたがって，同期到達パケットを生成して Center ノードへ送る.

④ Center ノードから同期成立パケットが届いたら，パケット解析&データ送出部がバリアフラグをリセットする.

4　クラスタ全体バリア同期

クラスタ全体のバリア同期は，シャーシ内外のネットワークを利用して行う．基本的には，シャーシ外ネットワークトポロジーを図 8 のような Tree トポロジと考えて，シャーシ内のみのバリア同期を行い，それが終了後にシャーシ間で同期を行う形でバリア同期を行う.

クラスタ全体のバリア同期は，各シャーシを「Root Chassis」，「Blade Chassis (Inter-Level)」，「Blade Chassis(Low-End)」の 3 つに分け，それぞれ異なった手順で同期を行う（図 9 参照）.

Root Chassis の同期手順を述べる.

① SCMD ネットワークを利用した Chassis 内のバリア同期を行う.

② 各ノードは下位階層 Blade の Center ノードに RootChassis のみのバリア同期完了を知らせ る.

③ 各ノードは下位階層 BladeChassis 全てが同期完了したことを示すパケットを受け取る.

④ 再度，SCMD ネットワークを利用した Chassis 内のバリア同期を行う.

⑤ バリア同期成立を知らせるパケットを下位階層 (Level)BladeChassis の Center ノードへ送る.

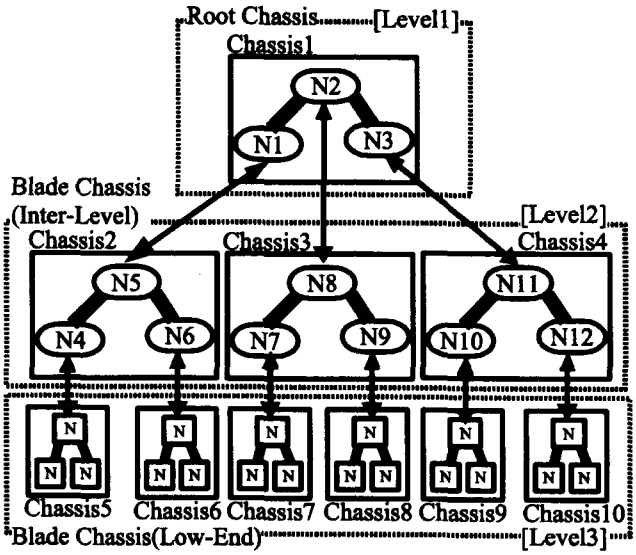

図 8. クラスタ全体同期のネットワーク構成

Blade Chassis (Inter-Level)の同期手順を以下に述べる.

① 上の階層の Chassis 送られてくるパケット Center ノードが受け取る.

② SCMD ネットワークを利用した Chassis 内のバリア同期を行う.

③ ライトおよびレフトノードが上位階層の Chassis でのバリア同期完了を示すパケットを下の階層の Center ノードに送る.

④ ライトおよびレフトノードが下位階層 Chassis の Center ノードから送られてくるパケット受け取る.

⑤ 再度，SCMD ネットワークを利用した Chassis 内のバリア同期を行う.

⑥ Center ノードが上位層のノードにパケットを送る.

⑦ Center ノードが上位層から送られてくるパケット受け取る.

⑧ 再度，SCMD ネットワークを利用した Chassis 内のバリア同期を行う.

⑨ ライトおよびレフトノードが階層にパケットを送る.

Blade Chassis (Low-End)の同期手順を以下に述べる.

① 上の階層のChassis送られてくるパケットCenterノードが受け取る.
② SCMDネットワークを利用したChassis内のバリア同期を行う.
③ Center ノードが上位層のノードにパケットを送る.
④ Center ノードが上位層から送られてくるパケット受け取る.
⑤ 再度, SCMDネットワークを利用したChassis内のバリア同期を行う.

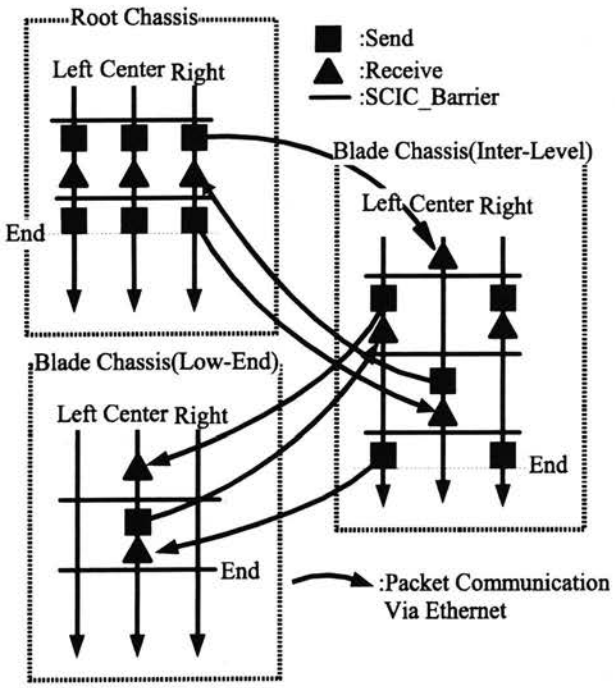

図 9. クラスタ全体バリア同期の処理手順

5　性能評価

同期ユニットの評価として, SCIC_Barrier における同期ユニット内の処理時間を見積もり, 実機でSCIC_Barrier の実行時間を測定した. また, クラスタ全体バリア同期の実行時間も見積もった.

5.1　SCIC_Barrier のレイテンシ

SCIC_Barrier の同期ユニット内の処理時間をシミュレーションによって見積もった. シミュレーションはアルテラ社の QuartusII (Ver5.1) を使用した. 図 10 にSCIC_Barreir処理のタイミングチャートを示す. Centerノードにおける SCIC_Barrier の処理手順をおおまかに3つの工程にわけ, Right・Left ノードの SCIC_Barrier の処理手順をおおまかに2つの工程にわけ, タイミングチャートに表した. 表 1 に SCIC_Barrier の各工程の処理時間を示した.

図 10 のタイミングチャートおよび表 1 に各処理の処理時間から Center ノードの SCIC_Barrier は 2.7μ s で終了し, Left または Right ノードの SCIC_Barrier は 3.46μ s で終了するという結果を得た. SCCB クラスタ[11]での同期ユニットの処理時間 (約 0.2μ s) に比べると約 17 倍処理時間がかかっている.

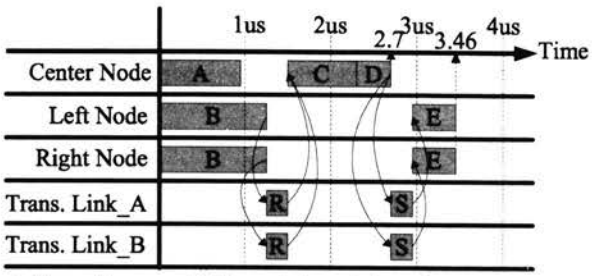

図 10. SCIC_Barrier の処理工程

表 1. 各プロセスにおける見積もり処理時間

プロセス	時間	プロセス	時間
A	0.96μs	B	1.25μs
C	0.79μs	E	0.495μs
D	0.39μs		
R	0.27μs	S	0.27μs

実際に, Right および Center ノード 2 ノードでの SCIC ネットワークを使用したバリア同期の処理時間を測定した.

その結果, Center ノードは 2.37μ s で完了し, Right ノードは 3.86μ s で完了した. 見積もり実行時間と実際に計測した結果がほぼ一致しているが, これは, デバイスドライバの処理時間を含んだ時間なので, 実際のシミュレーションのより早く処理されたと思われる.

Enternet のみを使った MPI_Barrier の場合, 2 台で約 120μs と約 1/30 削減されているが, SCCB クラスタ[11]でのバリア同期処理時間 (3.2μs) よりは処理時間が延びている. これは, 同期ユニットの差というより, デバイスドライバの処理時間の差である.

5.2　クラスタ全体バリア同期の評価

クラスタ全体バリア同期の実行時間を見積もった. 図

9 の手順をおおまかに見積もると以下のようになる.

N=2, 3 (N はプロセッサ台数) のとき,

$$T_{EB} = T_{IB} \cdots (1)$$

$N \geq 4$, $(L_I \geq 0)$のとき

$$T_{EB} = (4 + 3L_I)T_{IB} + 3(L_I + 1)T_{SR} \cdots (2)$$

ここで, T_{EB}はクラスタ全体のバリア処理時間, T_{IB}は SCIC_Barrier レイテンシ, L_I は Tree トポロジの Inter-Level 番号, T_{SR}は send/receive の処理時間である. 式(1)および式(2)をグラフで表す (T_{SR} をパラメータとして) と図 11 となる. 図 11 において, イーサーネットのみを利用した MPI_Barrier の実行時間と比較した. T_{SR} が 60us 程度だと, SCMD ネットワークを使用したバリア同期が有利であるが, T_{SR} が 120 を超えると, Ethernet のみを使用した MPI_Barrier が有利になるということが推測される.

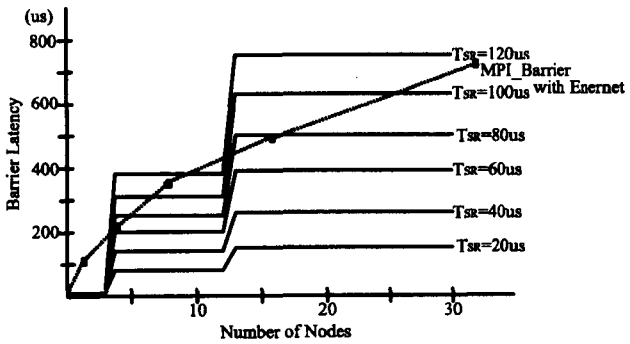

図 11. クラスタ全体バリア同期の処理時間見積もり

6　おわりに

シャーシ内ネットワークを使用したハードウェア同期ユニットの設計および評価を行った. SCIC ネットワークボードに搭載するコントロール chip の概要を説明し, そのボードによって構成される SCIC ネットワークについて述べた. また, SCIC ネットワークを利用したシャーシ内のバリア同期制御部の構成について述べ, そのブロックで行われるバリア変数管理手法について述べた.

性能評価として, シャーシ内バリア同期の処理見積もり時間および 2 ノードを利用した SCIC_Barrier の予備評価を行った. その結果, 2 ノードの SCIC_Barrier が 3.86μs で処理できた. この結果は, イーサーネット使用した MPI_Barrier の実行時間の約 30 分の 1 に相当する.

また, クラスタ全体バリアにおいても, send/Receive の実行時間が 60μs を超えなければ, イーサーネットのみを利用した MPI_Barrier よりよい結果になると見積もることができた.

今後の課題として, 実機 3 台で SCIC_Barrier を実行させ, クラスタ全体のバリア同期の実機測定を行うことと, このバリア同期を実アプリケーションに有効的に使うための方法を検討することが挙げられる.

参考文献

[1] M.Warren, E.Weigle, W.Feng, "High-Density Computing : A 240-Node Beowulf in One Cube Meter ",Super Computing 2002, Nov. 2002.

[2]中島,中村, 佐藤, 朴, 松岡, 高橋, 堀田,"高性能計算のための低電力・高密度クラスタ MegaProto",情報処理学会論文誌:コンピューティングシステム,Vol.46,No.SIG12 (ACS11), pp.46-61, Aug. 2005.

[3]IBM and Lawrence Livermore National Laboratory,"An Overview of the Blue Gene/L Supercomputer",Super Computing 2002, Nov. 2002.

[4]北村, 濱田, 宮部, 伊澤, 宮代, 田邊, 中條, 天野, "DIMMnet-2 ネットワークインタフェースコントローラの設計と実装", 先進的計算基盤システムシンポジウム SACSIS2005, pp.293-300, May.2005.

[5]Kiyoshi Hayakawa, "SCCB CLUSTER SYSTEM -SYNCHRONIZATION AND PSEUDO GLOBAL CLOCK COUNTER SYTEM-", 23rd IASTED International Conference on PDCN2005, pp.216-221, Feb.2005.

[6]堀田義彦,佐藤三久,朴泰祐,高橋大介,中島佳宏,高橋睦史,中村宏,"プロセッサの消費電力測定と低消費電力プロセッサによるクラスタの検討"先進的計算基盤システムシンポ SACSIS2004,pp.19-26, Mar.2004.

[7]鯉淵道紘, 渡邊幸之介, 大塚智宏, 天野英晴,"RHiNET-2 クラスタを用いたシステムエリアネットワーク向けトポロジの実機評価", 先進的計算基盤システムシンポジウム SACSIS2004,pp.381-388, Mar.2004.

[8]住元真司,成瀬彰,久門耕一,細江広治,清水俊幸,"PM/InfiniBand を用いた大規模 PC クラスタ向け高性能通信機構の設計", 先進的計算基盤システムシンポジウム SACSIS2004,pp.373-380, Mar.2004.

[9]Steven L. Scott, "Synchronization and Communication in the T3E Multiprocessor", ASPLOS VII, pp.26-36, Oct.1996.

[10]Kiyoshi Hayakawa, Satoshi Sekiguchi``Design

and Implementation of a Synchronization and Communication Controller for Cluster Computing Systems," Proc. 4th Intel. Conf. High Performance Computing in Asia-Pacific Region, vol.I, pp76-81, May.2000.
[11]Kiyoshi Hayakawa, Masahiko Iwane, Satoshi Sekiguchi,"SCC Beowulf-Cluster System for Accurate Performance Analysis", 5TH Int'l. Conf. High Performance Computing in Asia-Pacific Region, CD-ROM, Sep.2001.
[12]Steve Sistare, Rolf vande Vaart, Eugene Loh, "Optimization of MPI Collectives on Clusters of Large-Scale SMP's", Super Computing'99, CD-ROM (1999).
[13]Sathish S. Vadhiyar, Graham E. Fagg, Jack Dongarra,"Automatically tuned Collective Communications", Super Computing 2000, CD-ROM(2000).
[14]S.Shang, K.Hwang,"Distributed Hardware Barrier Synchronization for Scalable Multiprocessor Clusters",IEEE, Trans, Parallel Distib. Syst., vol.6, pp591-605, June.1995.
[15]OPENCORES.ORG (http://www.opencores.org)
[16]堀田義彦, 佐藤三久, 木村英明, 松岡聡, 朴泰祐, 高橋大輔, "PC クラスタにおける電力実行プロファイル情報を用いた DVS 制御による電力性能の最適化", 情報処理学会研究報告, 2006-HPC-105, pp.139-144, Mar.2006.