Vol.2012-EMB-26 No.5
2012/9/10

ディペンダブルVLSIのための高機能NoCルータアーキテクチャ

吉瀬 謙二[1]

概要：トランジスタの微細化にともないばらつきや宇宙線によって引き起こされるソフトエラー等のエラーが増加する．このため，ディペンダブル VLSI を実現する技術が様々なレイヤーで求められている．本稿では，同じ機能を持つ多くのモジュールが搭載される VLSI の空間冗長性に着目し，2 つのモジュールを利用した DMR(Dual Modular Redundancy) により信頼性を向上させる SmartCore システムについて述べる．本システムでは，高機能ルータと呼ばれる高機能化したネットワークオンチップ (NoC) ルータを活用し，DMR としてペアを組むモジュールの出力パケットを比較することでエラーを検出する．本稿ではまた，SmartCore システムにおいて DMR を実現する鍵となる高機能ルータのマイクロアーキテクチャについて述べ，一般的な NoC ルータと比較してハードウェアの増加がそれほど大きくないことを議論する．

1. はじめに

トランジスタの微細化にともないばらつきや宇宙線によって引き起こされるソフトエラー等のエラーが増加する．このため，ディペンダブル VLSI を実現する技術が様々なレイヤーで求められている．

本稿では，同じ機能を持つ多くのモジュールが搭載される VLSI の空間冗長性に着目し，2 つのモジュールを利用した DMR(Dual Modular Redundancy) により信頼性を向上させる **SmartCore システム** [1] について述べる．

チップに多くのモジュールを搭載する VLSI では，モジュール間の通信のために，スケーラビリティに優れたネットワークオンチップ (**NoC**) を用いることが選択肢の 1 つとなる．SmartCore システムでは，この NoC で用いられるルータに着目する．高機能ルータと呼ばれる高機能化した NoC ルータを活用し，DMR としてペアを組むモジュールの出力パケットを比較することでエラーを検出する．

本稿ではまた，SmartCore システムにおいて DMR を実現する鍵となる**高機能ルータのマイクロアーキテクチャ**について述べ，一般的な NoC ルータと比較してハードウェアの増加がそれほど大きくないことを議論する．

本稿の構成を示す．2 章で SmartCore システムの仕組みについて述べる．3 章でシステムを実現するための高機能ルータのマイクロアーキテクチャについて述べる．4 章で関連研究を述べ，5 章でまとめる．

[1] 東京工業大学 大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology

2. SmartCore システム

SmartCore システムは，高機能ルータと呼ばれる高機能化したルータが核となり，同じ機能を持つ複数のモジュールによる冗長実行によって**信頼性の向上を達成する仕組み**である．ここで，モジュールとは NoC ルータを用いて接続されるハードウェアであり，たとえば，メニーコアプロセッサの場合にはコアをモジュールと見なすことができる．

本稿では，**多数のモジュールがメッシュ接続**された VLSI を前提とする．また，NoC に関しては，ワームホールスイッチングと XY 次元順ルーティングという一般的な構成を前提とする．モジュール間の通信のために**パケット**が用いられる．また，パケットは 1 サイクルにルータ間で転送される**フリット**の集まりとして定義される [2].

2.1 一般的なシステムにおける通信

図 1 に，一般的なシステムにおける通信と SmartCore システムにおける通信との比較を示す．これらは，ノードの物理的な配置を示している訳ではなく，通信とハードウェアとの関係を表している．

図 1(a) は，一般的な NoC におけるパケットの送受信の様子である．点線で示すノードには，1 つのモジュールと 1 つのルータが含まれる．ここでは，ノード A に含まれるモジュール A から，ノード B に含まれるモジュール B への通信を示している．ルータ間の矢印を太線にしているが，

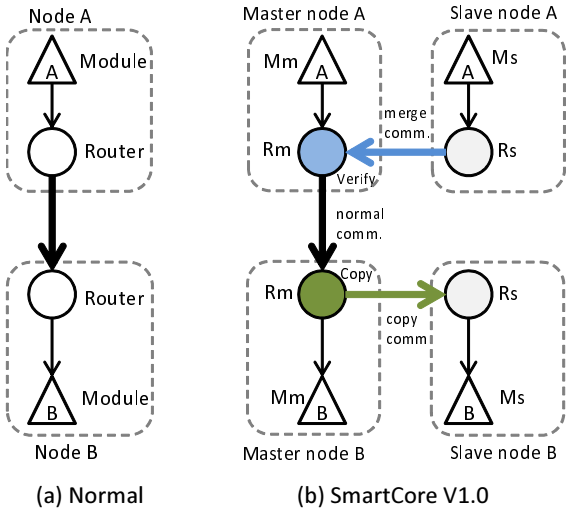

図 1 一般的なシステムにおける通信 (a) と SmartCore V1.0 システムにおける通信 (b) との比較．ノード A に含まれるモジュール A が送信したパケットをノード B に含まれるモジュール B が受信する．

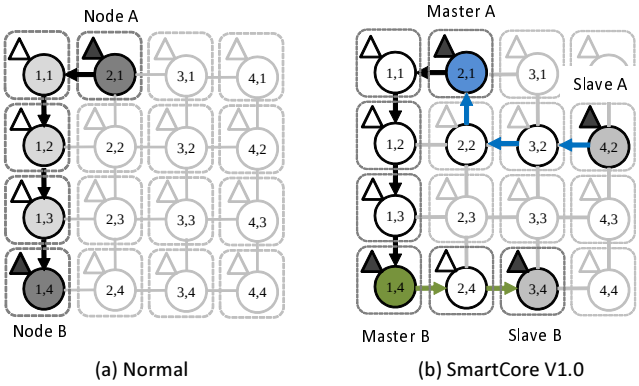

図 2 ノード A のモジュールからノード B のモジュールへの通信．(b) は SmartCore システムにおける通信の様子を示している．

これは，この通信において実際には複数のルータを経由する場合があることを意味している．

図 2(a) に，ノードの物理的な配置例と，その場合におけるモジュール A からモジュール B への通信を示す．ここでは，**論理ノード ID と物理ノード ID を区別することが重要**である．ノード A とノード B という ID は，プログラマから見える論理ノード ID である．一方，それぞれのノードの物理的な座標 (X,Y) で指定される ID が物理ノード ID である．

図 2 においても，三角はモジュールであり，丸はルータである．ルータには物理ノード ID が記入されている．この例では，ノード A という論理ノード ID のスレッドが物理ノード (2,1) に配置され，ノード B という論理ノード ID のスレッドが物理ノード (1,4) に配置されている．この時，ノード A からノード B への通信では，XY 次元順ルーティングを採用していることから，パケットは X 方向に転送され，その後，Y 方向に転送される．このため，(1,1), (1,2), (1,3) という物理ノードを経由してノード B に届くことになる．

ベースとなる VLSI システムが，論理ノード ID から物理ノード ID への変換の仕組みを持つことを前提とする．

2.2 SmartCore システムにおける通信

図 1(b) に，SmartCore システム[*1] における通信を示す．DMR としてペアを組むそれぞれのノードを**マスターノード**，**スレーブノード**と呼ぶ．マスターノードは図 1(a) におけるノードに対応し，スレーブノードは冗長実行のために追加されたノードである．

マスターノード A とスレーブノード A，マスターノード B とスレーブノード B のそれぞれを冗長実行のペアとする．ペアを組む 2 つのノードでは同一のスレッドを実行し，DMR を構成する．この時，ペアを組む 2 つのノードには同じ論理ノード ID が割り当てられ，ある時点におけるそれぞれのスレッドの処理は同じ (これまでに同じパケットを受け取っており，次に送信されるパケットが同じ) とする．ここで，図 1(a) と同様に，ノード A がノード B にパケットを送信することを考える．

図 2(b) に，SmartCore システムを用いた場合の，ノードの物理的な配置例と，その場合におけるモジュール A からモジュール B への通信を示す．マスターノード A が (2,1) に，スレーブノード A が (4,2) に，マスターノード B が (1,4) に，スレーブノード B が (3,4) に配置されている．この時，図 2(a) と比較すると，ルータ (3,2), (2,2) を経由するスレーブルータ A からマスタールータ A への通信が追加される．これを，**マージ通信 (merge comm.)** と呼ぶ．また，ルータ (2,4) を経由するマスタールータ B からスレーブルータ B への通信が追加される．これを，**コピー通信 (copy comm.)** と呼ぶことにする．また，図 2(a) で必要となる一般的な通信を**ノーマル通信 (normal comm.)** と呼ぶことにする．

まず，マージ通信について考える．モジュール B へのパケットはペアを組むそれぞれのモジュール A(マスターモジュール A およびスレーブモジュール A) から送信される．なにも制御しなければ，マスターモジュール A とスレーブモジュール A が送信したパケットがマスターモジュール B に 2 つ届いてしまうため，マスターモジュール B では正しい実行が保証されない．また，スレーブモジュール B にはパケットが 1 つも届かないため，こちらでも正しい実行が保証されない．

マスターモジュール B に同じパケットが 2 回届くこと

[*1] SmartCore システムには，TMR(triple modular redundancy) の実現を目指す Version 1.5 や，高機能ルータが冗長実行しているすべてのノードの情報を保持することで必要となる仮想チャネルの数を抑える Version 2.0 などがあるが，本稿の議論は Version 1.0 に限定している．

を防ぐため，SmartCore システムでは，スレーブモジュールが送信したパケットの宛先を，同じノードのルータにおいて変更する．この時，宛先はペアを組むマスターノード (図ではマスターノード A) に変更される．

こうすることで，マスターモジュール A とスレーブモジュール A が送信したパケットがマスターノード A のルータに届くため，これらをマージして 1 つのパケットとして送信することで，マスターノード B には本来必要な 1 つのパケットのみが届くようになる．また，マスターモジュール A とスレーブモジュール A が送信したパケットがマスタールータ A の届くことを利用して，マスタールータ A でこれらを比較することでモジュール A において発生したエラーを検出する．

この際，スレーブモジュール A がパケットを送信して，スレーブルータ A が宛先を変更して転送してから，そのパケットがマスタールータ A に届くまでの通信がマージ通信である．マージ通信は，エラー検出のために，スレーブルータがマスタールータに送信する通信と捉えることができる．

次に，ノーマル通信について述べる．プログラムを正しく実行するためには，モジュール A が送信したパケットをモジュール B が受信する必要がある．マスターノードは図 1(a) の一般的なシステムにおけるノードに対応する．このため，マスタールータ A においてマージされて送信されたパケットの宛先のノード B は，マスターモジュール B であり，自然に，マスターノード B がパケットを受信する．この通信は，一般的な通信と同じノーマル通信である．

最後に，コピー通信について述べる．ノーマル通信のみではマスターモジュール B にしかパケットが届かない．これではスレーブモジュール B でパケットを受け取ることができずに，ペアを組んでいるマスターモジュール B とスレーブモジュール B が同じ処理を継続できなくなる．

これを回避するために，マスターモジュール B に届くパケットをコピーして，同じ順番で，スレーブモジュール B に届けることにする．これは，マスタールータ B に届いたマスターモジュール B 宛のパケットをコピーして，コピーしたパケットの宛先をスレーブモジュール B に変更することで実現する (図 1(b) 中の Copy)．これにより，マスターモジュール B が受信するパケットは，同じ順番で，スレーブモジュール B に向けて送信される．このようにコピーされてからスレーブモジュール B に届くまでの通信がコピー通信である．

2.3 SmartCore システムにおける挑戦と制約

変換テーブルと制限される送信先

スレーブルータにおいてパケットの宛先を変更するためには，スレーブルータがペアを組んでいるマスタールータの物理ノード ID を知っている必要がある．また，マスタールータでパケットをコピーするためには，マスタールータがペアを組んでいるスレーブルータの物理ノード ID を知っている必要がある．

このような対応関係をどの程度持てばよいかは難しい選択だが，できるだけ少ない情報で実現できることが好ましい．このため，高機能ルータは，ペアを組む相手の物理ノード ID のみを知っているとする．10 ビットの ID を用いることで，1024 ノードを識別できる．高機能ルータでは，マスタールータであればペアを組むスレーブルータの ID を保存するレジスタを持つ．スレーブルータであればペアを組むマスタールータの ID を保持するレジスタを持つ．このために追加されるハードウェア量はたかだか ID のための十数ビットと少ない．

この制約のため，あるスレーブモジュールにパケットを送信できるのはペアを組んでいるマスタールータのみであることに注意して欲しい．図 1(b) において，例えば，スレーブルータ A からスレーブモジュール B にパケットを送信できない．同様にマスタールータ A からスレーブモジュール B にパケットを送信することもできない．この制約を満たしながら多重実行を実現するシステムとして設計しなければならない．

信頼性の向上

信頼性の向上が SmartCore システムの目的であるため，これが達成されなければ意味がない．

モジュールの入力および出力は必ずルータを経由するパケットとして送受信されることを前提とする (ルータにおけるパケット比較でエラーを検出するため，ルータを用いない通信の抜け道があると SmartCore システムはうまく機能しない)．そうであれば，ソフトエラーやタイミング制約違反などによりモジュールの内部でエラーが発生すれば，それはいずれ間違ったパケットとして出力される（または，本来は送信されるべきパケットが出力されなくなる）．モジュール内でエラーが発生しても，その出力に悪影響を引き起こさなければ，そのエラーはマスクされたことを意味しておりモジュールの実行は正しいと考えてよい．このため，SmartCore システムでは，マスクされたエラーを無理に検出せずに，出力されたパケット (厳密にはフリット) を比較してモジュール内でマスクされないエラーを検出する．

エラー検出のユニットでは，タイムアウトの機構も備えており，比較すべき片方のフリットが届いているが，もう一方からフリットが届いていない場合，この待っているサイクル数を覚えておくカウンタをインクリメントし，それがある閾値を超えると，送信されるべきパケットが送信されないエラーとして検出する．

このように，SmartCore システムでは出力パケットを比較して正しさを確認するが，入力パケットに関しては，その正しさを確認していない点に注意してほしい．例えば，コピー通信のネットワークにおいてエラーが発生して，あ

るフリットの一部のビットが反転したとすると，それはスレーブモジュールの入力にエラーが含まれることになる．モジュール内のエラーと同様に，この入力エラーがマスクされて出力パケットに現れなければ，そのエラーは検出する必要がない．一方，そうでなければ，そのスレーブモジュールが間違ったパケットを出力するので（あるいは，正しいパケットが出力されないので），それらは送信時のパケット比較にてエラーとして検出される．ただし，マスターモジュールとスレーブモジュールの両方に，同じエラーを含むフリットが入力されないように，ネットワークおよびルータにおけるエラーに対処する必要がある．

ネットワークおよびルータにおけるエラーについて検討する．ネットワークでは，パケットがフリット単位に分割されて，送信側から受信側に伝えられる．一般的に，ネットワーク上ではパケットのデータ自体に演算が加えられることはない．このため，ネットワークにおけるソフトエラーは，(1) フリット送信のデータパスにおけるビットの反転，(2) ルータの制御に関するエラー，という 2 種類に大別できる．(1) に関しては多くの通信システムで用いられているように，エラー検出あるいはエラー訂正の情報をフリット単位，あるいはパケット単位で追加することで緩和できる．(2) に関しては，制御に関するエラーなのでその検出や訂正は容易ではない．特に，マスタールータでエラーが生じる場合には，それが単一故障点となりうるため問題が深刻となる．これに対する方式を検討しているが，本稿では，これについては取り扱わないこととする．

高機能ルータのマイクロアーキテクチャが複雑になると，その部分が信頼性を低下させる要因となる．このため，本稿では，高機能ルータのマイクロアーキテクチャを示し，複雑さがそれほど増加しないことを議論する．

NoC におけるデッドロックの回避

ネットワークトポロジ，ルーティング，ルータのマイクロアーキテクチャ等に依存するが，SmartCore システムでは，それを用いない場合に発生しなかった通信パターンが生じるため，ネットワークにおいてデッドロックが発生する危険がある．

図 1(b) のスレーブルータ A では，出力されたパケットの宛先がそこで指定されている宛先から，マスターノード A に変更され，その方向にパケットが出力される．

スレーブルータ A にとって，マスタールータ A の配置は任意の方向となり得るので，これが本来禁止されている (XY 次元順でない) 通信パターンとなることがある．このような事態が生じたとしてもネットワークでデッドロックが生じないことを保証しなければならない．

SmartCore システムでは，これを保証するために複数の仮想チャネルを用意して，本来禁止されている通信パターンが生じるかもしれない場合に異なる仮想チャネルを割り当てる．すなわち，3 本の仮想チャネル (VC0, VC1, VC2) を用意して，ノーマル通信では VC0 を用い，マージ通信では VC1 を用い，コピー通信では VC2 を用いることでデッドロックを回避する．

エラー訂正ではなく，エラー検出で良いのか？

DMR では，エラーを検出できるが訂正はできない．リアルタイム性を必要としないバッチ処理の計算機システムではエラーが検出された場合にはそのプログラム全体を中断して，それらの結果を破棄して，最初から再実行すればよい．ソフトエラーの多くはマスクされることが知られている．また，SmartCore システムを用いてエラーが検出される頻度は数日に 1 回程度の低いことを想定している．1 つのプログラムの実行時間が数分程度であれば，数日に 1 回程度の頻度でそれを再実行したとしても全体のスループットに与える影響は非常に少ない．

システムソフトウェアが定期的にチェックポイントをとり，その時点からの再実行を可能にする場合には，それを活用してオーバーヘッドを削減できる．

エラー訂正を必要とするアプリケーションに関しては TMR の仕組みが必要となる．本稿で述べるシステムをベースに，SmartCore システムによる TMR を実現できる．ただし，これに関する議論は本稿の範囲外とする．

プログラミングモデルへの制約

メニーコアプロセッサにおいて，複数のコアに対する初期状態が一致しており，全く同じタイミングで同一のパケットの系列を入力として与えれば，出力されるパケットの系列もまた等しくなることが期待できる．

しかしながら，SmartCore システムにおいて冗長実行を行う 2 つのモジュールは，パケット複製により全く同じ系列のパケットを受信するものの，その時刻については差が生じる．SmartCore システムでは，そのような場合においても同一のパケット系列を出力することを，モジュールのアーキテクチャおよびプログラミングモデルに要求する．

3. 高機能ルータのマイクロアーキテクチャ

高機能ルータのマイクロアーキテクチャとしては様々な構成が考えられるが，一般的なルータからの変更とハードウェア量の増加を抑えた版 (Version 1.2) のマイクロアーキテクチャについて述べる．

図 3 に，高機能ルータのマイクロアーキテクチャ (Version 1.2) のデータパスを示す．一般的なインプットルータからの変更点は図の下部にまとめられている．

高機能ルータは，状況によって一般的な入力バッファルータとしても動作する．すなわち，信頼性がそれほど重要ではない状況では，冗長化実行を用いない構成とすることもできる．**図 4** に，一般的な入力バッファルータとして用いる高機能ルータの構成を示す．図 3 に示した構成において，利用しない部分を灰色に変更している．また，クロスバスイッチの後のデマルチプレクサの出力を固定させ

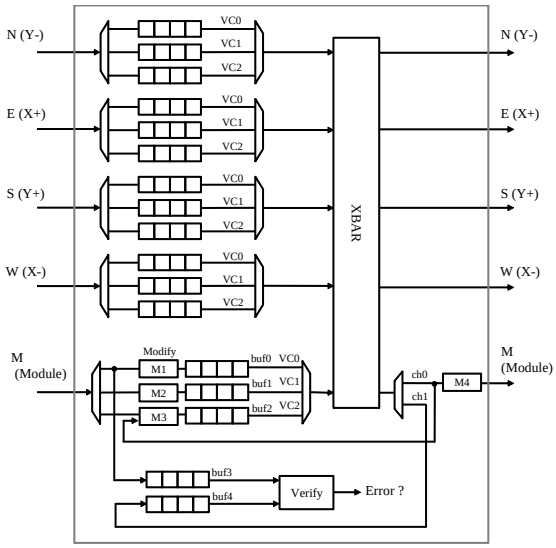

図 3 高機能ルータのマイクロアーキテクチャ (V1.2) のデータパス.

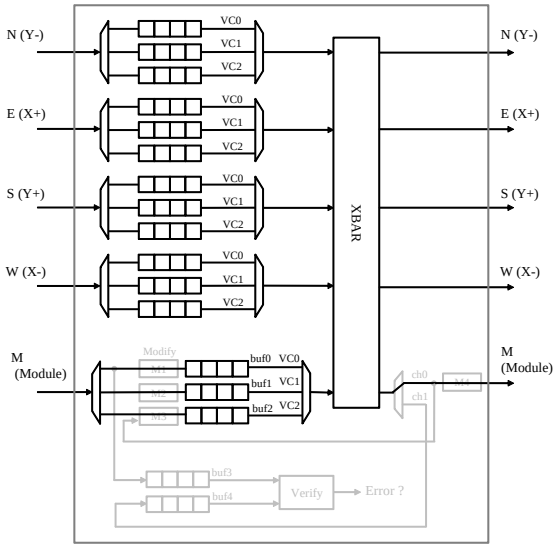

図 4 高機能ルータを一般的な入力バッファルータとして利用する構成.

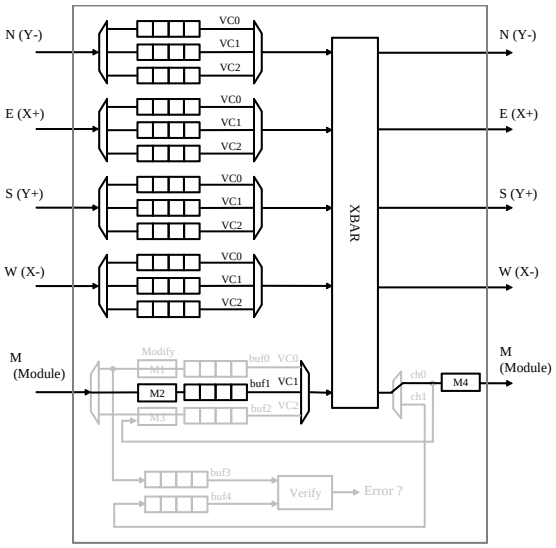

図 5 高機能ルータをスレーブルータとして利用する構成.

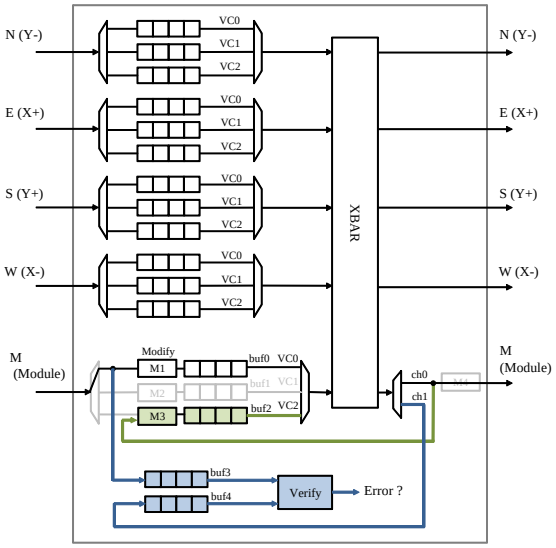

図 6 高機能ルータをマスタールータとして利用する構成.

る．この構成は，自明に，5 入力，5 出力，3 本の仮想チャネルを持つ一般的な入力バッファのデータパスである．この図の灰色で示したユニットが，高機能ルータにおいて追加される本質的なものである．

次に，高機能ルータをスレーブノードとして利用する構成を**図 5** に示す．スレーブルータにおいて，図左下の入力ポートのモジュールからパケットが入力される場合，そのパケットの宛先をマスターモジュールに変更する．これは，図の M2 と書かれた変更ユニット (Modify Unit) にておこなう．また，同時に，ヘッダフリットに書かれている利用する仮想チャネル番号を 0 から 1 に変更する．スレーブモジュールがパケットを送信する時は仮想チャネル 0 が指定されている．これを強制的に仮想チャネル 1 に変更するために，モジュールからの入力を強制的に VC1 に対応する入力バッファ buf1 に格納する．これにより，仮想チャネル 1 を用いて，パケットはマスタールータに届けられる．

スレーブノードにおけるパケット受信については，基本的に，スレーブノード宛に届いたパケットをモジュールに出力すればよい．ただし，このパケットはコピー通信を用いているため，仮想チャネルは 2 が利用されている．一方，モジュールの入力としては仮想チャネル 0 の利用が前提となっている．このため，M4 のモジュールにてヘッダフリットに書かれている仮想チャネル番号を 2 から 0 に変更して，モジュールの入力とする．

高機能ルータをマスタールータとして利用する構成を**図 6** に示す．高機能ルータが真価を発揮するのはこの構成である．高機能ルータは一般的なルータと同様に，(1) パケットを宛先に送信する，(2) 送られてきたパケットを受信する，に加えて，送信側のマスタールータでは，(3) マスターモジュールとスレーブモジュールが出力したフリットの比較によるエラー検出，(4) 受信側のマスタールータに

おけるパケットのコピー，の機能を正しく実現する必要がある．それぞれを順番に見ていく．

(1) マスターモジュールが送信したパケットは buf0 に格納され，通常のパケットとして，適切な方向に出力される．

(2) 受信側のマスターモジュールを宛先とする通常のパケットは仮想チャネル 0 を用いているため，このチャネルを用いているパケットはそのままモジュールの入力となる．仮想チャネル 0 ではないマージ通信によるパケットを入力としてはいけないので，図右下のデマルチプレクサを用いてこれを実現する．

(3) パケットの比較のために，マスターモジュールが送信したパケットは，buf0 に加えて，buf3 にも格納される．また，ペアを組むスレーブモジュールが送信したパケットは仮想チャネル 1 を用いて，マスタールータに届くので，そのパケットは図右下のデマルチプレクサにて選択され，buf4 に格納される．エラー検出ユニット (図の Verify) は，buf3 と buf4 とにフリットが格納されている場合に，それらを取り出して比較する．フリットが一致しない場合には，モジュールが間違ったフリットを送信したと考えられるので，エラーとして検出する．

ただし，比較に関して，ヘッダフリットの扱いに注意する必要がある．これは，スレーブルータにおいてフリットの宛先が変更されているからである．このため，単純な手法では，スレーブモジュールが送信先を間違えたパケットを送信したとしても，この宛先の情報を排除しているため，このエラーは検出できない．運良くヘッダフリットにもう 1 つの宛先を格納する領域が余っている場合には，そちらに，オリジナルの宛先を格納しておき，比較の際にはそのフィールドを用いることもできる．

エラー検出ユニットにはタイムアウトを検出する機構も含まれており，buf3 と buf4 のどちらかにフリットが含まれている場合にカウンタをインクリメントする．このカウンタは，比較がおこなわれた場合にゼロに初期化し，その値が閾値を超えた場合にエラーとする．本システムでは，エラー検出の頻度が非常に小さいことを前提としているため，このカウンタの閾値としては，通常の通信遅延などで発生するカウンタの値と自明に区別できる値 (例えば 10 万サイクル程度) を用いればよい．

(4) パケットのコピーについては，コピーすべきパケットがマスターモジュールに入力される仮想チャネル 0 のものである，という性質を利用する．このため，図右下にて，モジュールへ入力されるフリットを仮想チャネル 2 のバッファである buf2 に挿入する．また，この時，ヘッダフリットに関してはペアを構成するスレーブモジュールになるように宛先を，また用いる仮想チャネルが 2 になるように M3 にて変更する．これにより，buf2 に格納されたフリットが仮想チャネル 2 を用いてスレーブルータに届く．

高機能ルータのハードウェア量を議論する．モジュール以外の入力バッファ，入出力ポート，クロスバスイッチ (XBAR) の構成は一般的なルータと同じである．仮想チャネルアロケータ，スイッチアロケータ，バックプレッシャーの制御では幾つかの条件の追加が必要となるが，それは少しの変更点である．ハードウェアの増加の多くは，エラー検出のためのバッファ (buf3, buf4) の追加が原因である．単純にすべてのバッファのエントリ数が同じだとすると，一般的なルータで 15 本利用されているバッファが高機能ルータでは 17 本に増加し，その増加率は 14%である．その他，変更ユニットやエラー検出ユニットによる増加を考慮しても，全体で 20%以下のハードウェア量の増加に抑えることができると考えている．

4. 関連研究

SmartCore システムは，多数のモジュールを搭載する VLSI システムにおける空間的冗長性を利用して，複数のモジュールを冗長実行させることで信頼性を向上させる仕組みである．信頼性向上のためのエラー検出を，高機能ルータにおけるパケットの比較でおこなっているところに特徴がある．

ここでは，多数のモジュールを搭載する VLSI システムの例として，マルチコアプロセッサおよびマルチプロセッサに着目して関連研究をまとめる．

複数の CPU に同一の動作をさせることで故障を検出する技術として，PowerPC 750GX に採用されている Lock-step[3] がある．チェック用のプロセッサを追加し，プロセッサの入出力のレベルでエラー検出する．2 つのプロセッサは，システムの起動時からクロックレベルで完全に同期する．冗長実行を行うコアの組を自由に決定できる SmartCore システムと比べ，この手法は，冗長実行を行うプロセッサの組が固定されていて柔軟性に欠ける．

マルチコアプロセッサの空間的冗長性を利用し，エラー検出および回復をおこなう手法に Loose Lock-stepped システム [4], [5] がある．この手法では，コア，キャッシュおよび主記憶をいくつかのグループに分割し，冗長実行させる．エラー検出はチップ外のレベルでおこなう．この手法は，オンチップの通信で発生するエラーを検出できない．

マルチコアプロセッサの空間的冗長性を利用し，高性能化とディペンダビリティ向上を達成する方式に Slipstream プロセッサ [6] がある．通常の命令列を 1 つのコアで動作させ，同時に，同様の挙動を示す短い命令列を異なるコアで動作させることで，高性能化とディペンダビリティの向上を達成する．この方式は，コアの同期のために低レイテンシの特別なコア間通信の仕組みを必要とするため，任意のペアを利用した冗長実行はできない．

5. まとめ

本稿では，同じ機能を持つ多くのモジュールが搭載される VLSI の空間冗長性に着目し，複数のモジュールを利用した DMR(Dual Modular Redundancy) により信頼性を向上させる SmartCore システムについて述べた．本システムでは，高機能ルータと呼ばれる高機能化したネットワークオンチップ (NoC) ルータを活用し，DMR としてペアを組むモジュールの出力パケットを比較することでエラーを検出する．

本稿ではまた，SmartCore システムにおいて DMR を実現する鍵となる高機能ルータのマイクロアーキテクチャについて述べ，一般的な NoC ルータと比較してハードウェアの増加がそれほど大きくないことを議論した．

謝辞 本研究の一部は，科学技術振興機構・戦略的創造研究推進事業 (CREST) 「アーキテクチャと形式的検証の協調による超ディペンダブル VLSI」の支援による．本研究の実施に関しては，その中心的な役割を果たしている佐藤真平氏，藤枝直輝氏，高前田伸也氏，笹河良介氏，小林諒平氏，池田貴一氏に感謝します．

参考文献

[1] 吉瀬謙二，植原昂，佐藤真平： メニーコアプロセッサのディペンダビリティ向上と高性能を目指す SmartCore システム，情報処理学会研究報告 2008-ARC-180，pp. 49–52 (2008).
[2] Dally, W. J. and Towles, B.: *Principles and Practices of Interconnection Networks*, Morgan Kaufmann (2004).
[3] IBM Application Note: *PowerPC 750GX Lockstep Facility* (2008).
[4] Aggarwal, N., Ranganathan, P., Jouppi, N. P. and Smith, J. E.: Configurable Isolation: Building High Availability Systems with Commodity Multi-core Processors, *Proc. 34th Annual International Symposium on Computer Architecture*, pp. 470–481 (2007).
[5] Aggarwal, N., Jouppi, N. P., Smith, J. E., Ranganathan, P. and Saluja, K. K.: Implementing High Availability Memory with a Duplication Cache, *Proc. 41st Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 71–82 (2008).
[6] Sundaramoorthy, K., Purser, Z. and Rotenberg, E.: Slipstream processors: Improving both Performance and Fault Tolerance, *Proc. 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 257–268 (2000).