社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－ＳＬＤＭ－112 (22)
2003／11／28

自動合成を指向した階層型マルチプロセッサアーキテクチャとその設計

大川 大介† 西門 秀人† 原 武弘† 加藤 俊之† 山内 寛紀†

†立命館大学 VLSI センター
〒525-8577 滋賀県草津市野路東 1-1-1
E-mail: †{re002984,yamauchi}@se.ritsumei.ac.jp

あらまし C 言語で記述されたプログラムからリアルタイム処理向けマルチプロセッサシステムの自動合成を行うシステムを提案している[1] [2]. 本システムでは，様々なマルチプロセッサの規模に柔軟に対応できる，階層的なネットワーク構造を持つ分散メモリ型マルチプロセッサアーキテクチャを採用している．また，分散メモリ型マルチプロセッサでは，PE（プロセッサエレメント）間通信によるオーバーヘッドがマルチプロセッサ全体の処理性能に影響するため，PE 間通信を効率よく行うことが求められる．今回，PE アーキテクチャと PE 間ネットワーク SW の検討とその設計を行い，FFT プログラムを用いたシミュレーションによる性能評価実験を行ったので報告する．

キーワード マルチプロセッサシステム ハードウェア／ソフトウェア自動合成 システムオンチップ

Design of Hierarchical Multi-processor architecture for Automatic Generation

Daisuke OKAWA† Hideto NISHIKADO† Takehiro HARA† Toshiyuki KATO†
and Hironori YAMAUCHI†

†VLSI center, Ritsumeikan University

1-1-1 Nojihigashi, Kusatsu-shi, Shiga, 525-8577 Japan
E-mail: †{re002984,yamauchi}@se.ritsumei.ac.jp

Abstract We proposed a multi-processor system based software and hardware co-design platform intended for generating real-time applications[1] [2]. In this system, we adopt the distributed memory type multi-processor architecture with hierarchical network which can respond to the various scale of multi-processor flexibly. In a distributed memory type multi-processor, since the overhead by communication between PEs (Processor Element) influences the processing performance of multi-processor system, performing communication between PEs efficiently is called for. We have examined and designed a PE architecture and PE network SW, and we have experimented performance comparison by simulation with FFT program. So we are reporting here.

Keyword Multi-processor system, Hardware-Software Co-design, System on Chip

1. はじめに

マルチプロセッサの構成方式は，共有メモリ型と分散メモリ型に分けられる．共有メモリ型では，従来のマイクロプロセッサ技術が応用できるため比較的容易に実現が可能であるが，メモリアクセス制御等の問題から搭載できる PE 数に限界がある．一方，分散メモリ型では，大規模なマルチプロセッサの実現が可能であるが，PE 間の相互接続網や，PE 間通信によるオーバーヘッドなど課題も多い．

提案するマルチプロセッサ自動合成システムにおいては，マルチプロセッサアーキテクチャとして，小規模なマルチプロセッサから大規模なマルチプロセッサまで，柔軟に対応できることが望まれる．そこで，分散メモリ型マルチプロセッサアーキテクチャを採用し，階層型の PE 間ネットワークとすることでマルチプロセッサの規模に柔軟に対応できるようにした．また，PE 間通信は全二重方式と半二重方式について検討を行い，HDL による設計とシミュレーションを行ったので報告する．

2. マルチプロセッサ自動合成システム

自動合成システムは，大きく分けてソフトウェア生成部とハードウェア生成部から構成される．処理の流れを図 1 に示す．ソフトウェア生成部では，C 言語で

書かれたアプリケーションプログラムの並列化とコンパイルを行い，各 PE のバイナリコードを生成する．ハードウェア生成部では，PE の HDL 記述と SW の HDL 記述を基に結線を行ったネットリストの自動生成を行い，マルチプロセッサシステム全体の HDL 記述を生成する．

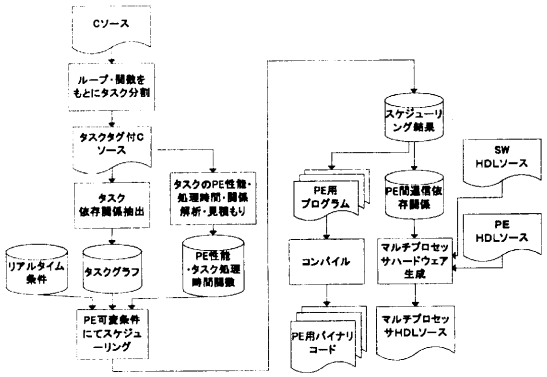

図 1　マルチプロセッサ合成フロー

3. マルチプロセッサアーキテクチャ

3.1. PE アーキテクチャ

マルチプロセッサの PE 内部構造を図 2 に示す．コアプロセッサは，データ長，命令長ともに 32bit の RISC 型プロセッサ[3], [4]を採用した．IF（命令フェッチ），ID（命令デコード），EX（実行），MEM（メモリアクセス），WB（ライトバック）からなる 5 段パイプライン構造としている．また，ハーバードアーキテクチャの採用，および，Forwarding 機能の搭載により，パイプラインの構造ハザードとデータハザードの回避を行う．

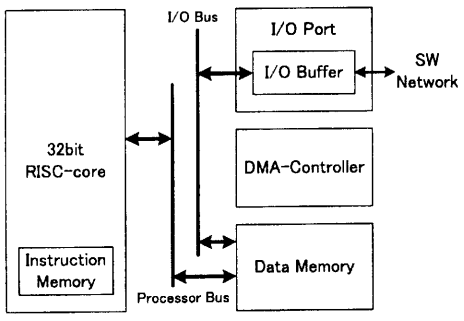

図 2　PE の内部構造

このコアプロセッサに対しマルチプロセッサ用 PE として用いるために，PE 間通信時に通信処理とデータ転送処理を行う I/O ポート，データメモリ（DMEM）と I/O ポート間のデータ転送処理を，コアプロセッサと独立して行う DMAC（DMA Controller）を搭載している．また，DMEM はコアプロセッサと I/O ポートが独立してアクセスできるように Dual-Port-Memory としている．

3.2. PE 間ネットワーク

PE 間ネットワークは図 3 に示すように階層的な構造とする．

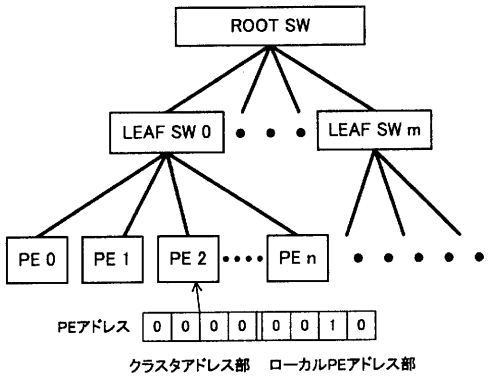

図 3　ネットワーク構造と PE アドレス

複数の PE をスイッチ（LEAF SW）で接続してクラスタを形成し，さらに LEAF SW を上位のスイッチ（ROOT SW）で接続する．なお，ROOT SW には PE を直接接続することも可能である．1 つの SW に接続できる SW あるいは PE は 16 までとし，最大 2 階層，256PE まで接続可能なネットワーク構造とした．

PE 間通信時の各 PE の識別は PE アドレスによって行う．各クラスタに固有に割り振られたクラスタアドレス（4bit）と，クラスタ内の各 PE に固有に割り振られたローカル PE アドレス（4bit）によって決まる．この PE アドレスは，PE 間通信開始時に PE-SW 間のデータバスを利用して送出する．

3.3. SW の基本アーキテクチャ

マルチプロセッサの相互接続網は様々な方式が提案されているが[5][6]，本システムではマトリクス結合網（クロスバ網）を採用した．マトリクス結合網では，同時に複数の通信が可能であり，高い PE 間通信性能を得ることが可能である．しかし，回路規模はノード数の 2 乗に比例して増加することが予想されるため，SW に接続される PE，SW 数を 16 までと制限している．

SW は 3 つのモジュールによって構成される（図 4）．経路接続の制御を行うコントローラ・モジュール，入

力される PE アドレスを解析するアドレスデコーダ・モジュール，マトリクス結合網（クロスバ網）と等価なセレクタの集合体であるネットワーク・モジュールである．

PE-SW 間の信号線は制御信号（req，ack，val）とデータバスからなる．また，データバス幅は，PE-SW 間の配線の複雑度，SW の回路規模を考慮して 8bit とした．これらの制御信号線とデータバス線を，接続 PE，SW 数分用意する．

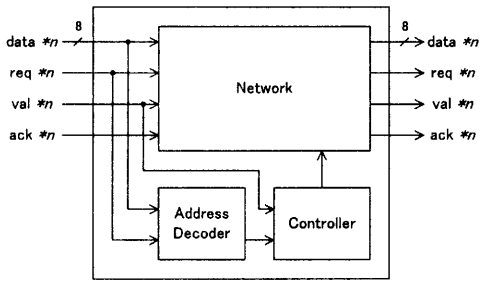

図 4　SW の内部構造

SW の各モジュールの構造を説明する．コントローラ・モジュールはステートマシンにより SW の接続状態の制御を行う．アドレスデコーダ・モジュールのアドレス解析結果と val 信号により，SW の接続制御信号をネットワーク・モジュールに出力する．なお，出力調停用のアービターの機能も内蔵している．また，val 信号を監視し，通信の終了を検出すると回線を切断して接続を終了する．アドレスデコーダ・モジュールは req 信号を監視し，通信が開始されるとデータバスに入力された PE アドレスの解析結果をコントローラ・モジュールに出力する．ネットワーク・モジュールは，セレクタの集合体である．コントローラ・モジュールからの制御信号により経路接続を行う．

3.4. 通信手順

PE 間通信の流れを示す．（図 5）．

1) 送信命令の実行

PE 間通信は，送信側プロセッサの送信命令実行により開始する．送信命令では，PE 間通信に必要なパラメータ（データロードアドレス，データ数，ID 番号，送信先 PE アドレス，送信先 PE におけるデータストアアドレス）を指定し，DMAC へ出力する．これらのパラメータは DMAC 内の送信命令キューに書き込まれる．

2) DMAC の起動

DMAC は，キューからパラメータを取り出し，データロードを開始すると同時に，I/O ポートに対して req 送信要求，PE アドレス，ID 番号を出力する．

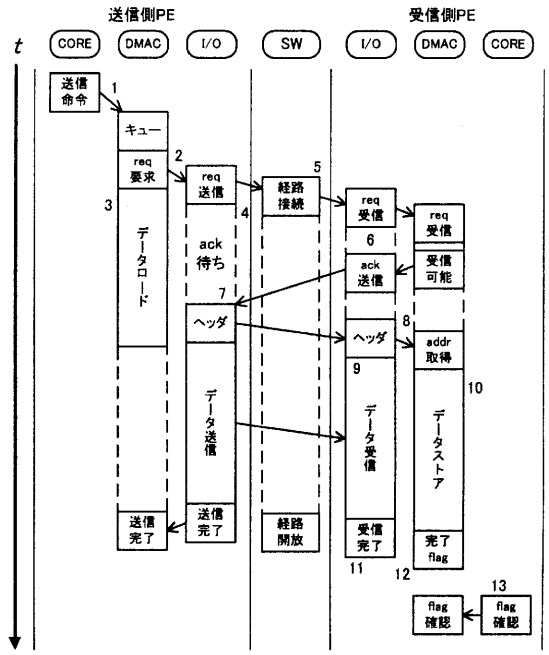

図 5　通信手順

3) データロード

DMAC は，DMEM から I/O ポート内のバッファ（I/O BUF）へ送信データのロードを行う．

4) req 信号と PE アドレスの送出

I/O ポートは SW に対して req 信号と PE アドレスを出力し，ack 受信待ち状態になる．

5) 経路接続

SW は req 信号と PE アドレスを受信すると経路の接続を行い，送信先 PE に req 信号を送信する．

6) req 信号の受信と受信処理の開始

受信側 PE が req 信号を受信すると I/O ポートは受信状態となり，DMAC へ受信開始信号を出力する．DMAC は受信可能な状態であれば I/O ポートへ ack 送信要求を出力し，I/O ポートは ack を送信側 PE に対し送信する．

7) ack の受信とデータ送信の開始

送信側 PE が ack を受信すると I/O ポートはデータ送信を開始する．データのヘッダとして送信先 PE におけるデータストアアドレス，データ数および ID 番号を付加して送信を行う．データ送信時には val 信号を出力することでデータ送信状態を表す．

8) データ受信の開始

受信側 PE は val 信号を監視し，受信するとデータヘッダ部の受信を開始し，DMAC へ出力する．

9) データ受信

I/O ポートはデータを受信し I/O ポート内のバッファに一時的に格納する.

10) データストア

DMAC は受信したアドレスに対して受信データのストアを行う.

11) データ受信完了

I/Oポートは val 信号を監視しデータ受信を終了する.

12) データストアの終了

DMAC はデータストアを終了すると,受信した ID 番号に対応するフラグを立て受信処理を終了する.

13) 受信確認

受信データを使用する処理を行う場合は,ID 番号に対応するフラグを確認してから行う.

3.5. 通信遅延時間

SW を経由するごとに 1clock の遅延が生ずる.

PE 間通信による遅延時間(送信命令の実行から受信側 PE でのフラグ確認終了まで)は,LEAF SW のみを経由する場合は,

$t = 4n + 22$ (n:データ数　単位:Word=32bit)

LEAF SW と ROOT SW を経由する場合は,

$t = 4n + 24$

となる.

3.6. 通信同期

PE 間通信同期方式は,従来からバリア同期等の研究が行われているが,本システムは,各 PE が自律して動作する分散メモリ方式であるので,バージョンナンバー手法を基に,各 PE で依存関係のある受信データに対してのみ同期を取る方式とした.各 PE には,DMAC 内に受信 ID 番号とフラグを対応させた ID テーブルを搭載し(図 6),データ受信が終了したら ID 番号の書き込みとフラグのセットを行う.後続タスクはフラグがセットされるまでビジーウェイトし,フラグ確認を行ってから実行を始める.

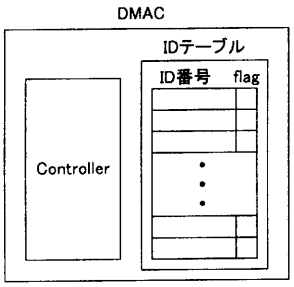

図 6　DMAC と ID テーブル

4. 全二重通信と半二重通信

通信方式は,同時に送受信することが可能な全二重通信と,同時に送信と受信を行うことができない半二重通信に分けられる.今回,前章のアーキテクチャを基に,全二重通信と半二重通信の両方の方式についてハードウェア・アーキテクチャと通信手順の検討を行った.SW,PE 内の I/O ポートと DMAC は各方式において構造および制御方式が異なるので,その違いについて説明する.

4.1. 全二重通信

送受信を同時に行うため,I/O BUF は送信用と受信用を別々に設けている(図 7).また,I/O ポート内のコントローラも送信用と受信用で独立して制御を行う.

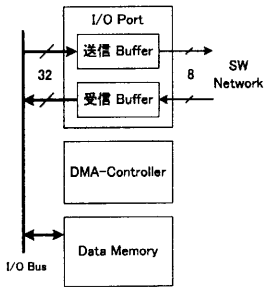

図 7　全二重通信用 PE 内部構造(一部)

送受信を同時に行う場合には,DMEM のバス幅が 32bit, PE-SW 間のバス幅が 8bit と帯域幅が異なるため,DMAC がデータストアとデータロードを切り替えることでデータの送受信を遅滞なく行うことができる.

4.2. 半二重通信とデッドロック回避方法

半二重通信では送受信が同時に起こらないため,I/O BUF は 1 つとした.

半二重通信では,デッドロックが発生する可能性があるため,その回避方法を考慮する必要がある.本システムでは,送信命令が実行され送信手順が開始されると,DMAC と I/O ポートは ack 受信待ち状態となる.しかし,PE が互いに送信を始める,あるいは複数の PE がループ状に req を送信した場合,各 PE は ack 受信状態から処理が進行しなくなる.この状態がデッドロックである.

本システムにおけるデッドロックの回避方法は,SW が優先度の高い PE, SW からの req のみを送信先へ出力することで行う.PE が送信手順の開始後に req を受信した場合は送信手順を中断し,データ受信を行い,受信処理終了後に再び送信手順を開始する.

具体的なデッドロック回避手順を図 8 の場合を例として説明する．req 信号の送受信が同時に起こる場合，図の左の PE からの受信を優先するものとする．

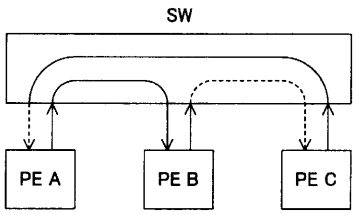

図 8　デッドロック回避方法の例

1. PE A は送信の優先度が最も高いため，PE C からの req 信号は SW 内で中断される．
2. PE B は PE A からのデータ受信を優先するため，SW は PE A からの req 信号を転送し，送信手順を中断して受信処理を開始する．
3. PE B から PE C に対する req 信号は PE B が req 信号の受信を行うため SW 内で中断される．
4. PE C から PE A に対する req 信号は，PE A からの送信を優先するため SW 内で中断される．

以上の手順で，デッドロックの回避を行う．

5. 設計とシミュレーション実験

5.1. 設計環境

設計は Verilog-HDL にて行った．シミュレータは Cadence 社の Verilog-XL，論理合成は Synopsys 社の Design Compiler を使用した．ターゲットライブラリはローム社提供の 0.35um プロセスとした．

5.2. 各モジュールの回路規模

論理合成を行い，回路規模の見積もりを行った．PE のモジュール（コアプロセッサ，I/O ポート，DMAC）ごとの回路規模と PE の回路規模は表 1 の通りである（メモリを除く）．プロセッサコアは全二重通信と半二重通信で同一のものを使用する．

表 1　PE の回路規模

	全二重	半二重
RISC-core	20,627	全二重と同じ
I/O ポート	6,188	5,026
DMAC	15,787	15,593
PE	43,428	42,096

単位：ゲート　（2 入力 NAND 換算）

SW の接続 PE，SW 数による回路規模は図 9（ROOT SW），図 10（LEAF SW）の通りである．

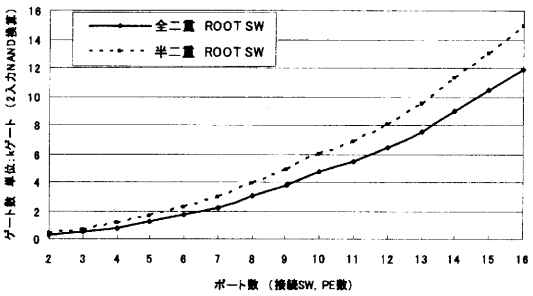

図 9　ROOT SW の回路規模

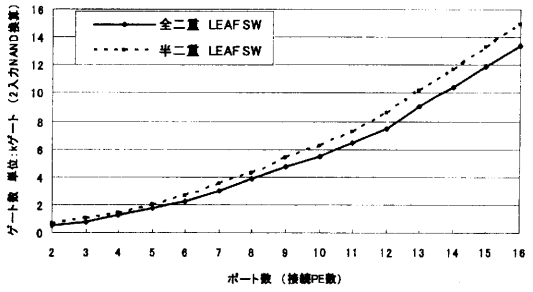

図 10　LEAF SW の回路規模

マトリクス結合網では，接点の数は

$$n(n-1) \quad （n：接続 PE，SW 数）$$

（自 PE に対する PE 間通信は定義しない）で与えられ，回路規模も接続 PE，SW 数の 2 乗に比例して増加することが，図 9，10 からわかる．

ROOT SW，LEAF SW ともに半二重通信用の方が，回路規模が大きい．これは，半二重通信用 SW にはデッドロック回避用に機能を追加しているためである．

なお，次節のシミュレーション実験で用いた 17PE 構成のマルチプロセッサの回路規模は，約 750,000 ゲートとなる．

5.3. シミュレーション実験

性能比較用プログラムとして，128 点 FFT をマニュアルにて並列化を行い，シミュレーションを行った．

FFT プログラムの概略とデータフローについて説明する（図 11）．元データは ROOT SW に接続された PE 16 のメモリ内に格納されており，データの並べ替え処理（ビットリバース）を PE 16 のみで行う．並べ替えが終わると他の 16PE に PE 間通信によりデータ転送を行う．DFT 処理用 PE は，データ転送を行いながら DFT

演算を行い，演算が終了するとPE 16に結果を送信しデータストアが終了するとFFTプログラムは終了する．

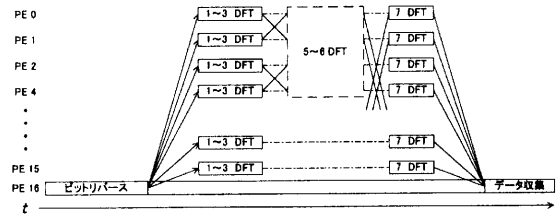

図 11　FFTプログラムのデータフロー（一部省略）
（実線はPE間通信を表す）

FFT処理用の16PEの接続構成は，以下の3通りの場合について同一のプログラムにてシミュレーションを行った．

1. 2クラスタ，クラスタ内PE数8
2. 4クラスタ，クラスタ内PE数4（図12）
2. 8クラスタ，クラスタ内PE数2

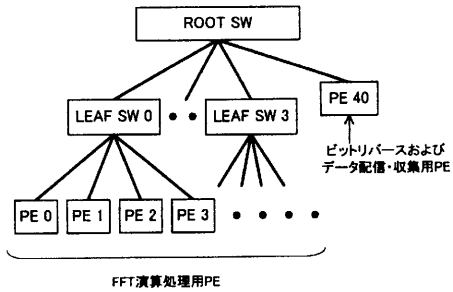

図 12　FFTプログラム用マルチプロセッサ構成

128点FFTのシミュレーション結果を表2に示す．各プログラムにおけるPE間通信回数は128回，1回当たりの転送データ量は，データ配信，収集時には16Word（1Word=32bit），バタフライ演算処理時には16Wordである．

表 2　128点FFTの処理時間（単位：clock）

	1	2	3
シングル	245,137		
全二重	43,593	43,593	43,593
半二重	43,593	43,593	43,593

（ビットリバース処理時間：29,483 clock）

5.4. 考察

各マルチプロセッサ構成で処理時間に違いがないのは，ROOT SWを経由するPE間通信がFFT演算処理のクリティカルパスにはなっていないためである．

また，このプログラムでは同一の通信路で送受信が行われることがなかったため，全二重通信と半二重通信による性能差はない．PE間通信が頻発する，あるいは転送データ量の多いPE間通信が発生するプログラムでは，全二重通信と半二重通信の性能差が出る可能性があるため，今後さらに各種プログラムについて実験を行う必要がある．

128点FFTプログラムでは，全体での処理時間はシングルプロセッサ時に比べ約5.6倍，ビットリバース後のDFT処理部では約15.3倍向上しており，非常に高い並列処理性能が得られることがわかる．

6. まとめ

本稿では，筆者らの提案するマルチプロセッサ自動合成システムのマルチプロセッサアーキテクチャの設計とその評価実験について述べた．評価実験では，非常に高い並列処理性能が得られることを確認した．

今後の予定としては，JPEG2000等のC言語で記述された大規模な動画像応用プログラムからマルチプロセッサを自動生成し，LSI化を行って性能評価を行う．

文　献

[1] 池田哲崇，西門秀人，瀬戸与志孝，加藤俊之，山内寛紀，“大規模マルチプロセッサ自動合成システムの基本検討”，信学技報[VLD2002-97]，pp.91-96，Nov 28. 2002.

[2] 西門秀人，瀬戸与志孝，池田哲崇，加藤俊之，山内寛紀，“GAスケジューリングによるリアルタイムマルチプロセッサ自動合成システム”，信学論（C），vol.J85-C，no.12，pp.1216-1228，Dec. 2002.

[3] ヘネシー，パターソン，“コンピュータ・アーキテクチャ　設計・実現・評価の定量的アプローチ”，日経BP社，1992.

[4] パターソン，ヘネシー，“コンピュータの構成と設計　第2版　ハードウエアとソフトウエアのインタフェース”，日経BP社，1999.

[5] 森村知弘，田中健介，岩井啓輔，天野英晴，“スケジューリングを考慮した多段結合網スイッチチップの実装”，信学技報[VLD2001] vol.101，no.46，pp.43-50，May 18. 2001.

[6] 藥袋俊也，緑川隆，田辺靖貴，茂野真義，天野英晴，“オンチップマルチプロセッサ向け内部接続網の検討”，情報処理学会[2003-ARC-153] no.153-1，pp.1-6，May 8. 2003.