C-005

演算レベル並列処理用マルチ ALU プロセッサの設計と実現

Design and Implementation of a Multi-ALU Processor with Operation Level Parallelism

石川　陽章†　　杵川　大智†　　境　直樹†　　孟　林‡　　山崎　勝弘‡
Haruaki Ishikawa　Daichi Kinekawa　Naoki Sakai　Lin Meng　Katsuhiro Yamazaki

1. はじめに

近年の急速な半導体製造技術の発展により、LSI の小型化、高速化が可能となり、低消費電力化が進められている。LSI 開発技術はハードウェアとソフトウェアに密接な関係があり、ハードとソフトの両方の知識が求められる。我々はプロセッサ設計を中心として、ハードとソフトの両方の知識を習得するためのハード/ソフト協調学習システム(HSCS)の研究を進めてきた[1]～[3]。

本研究では、演算レベル並列性の検証が可能な複数の ALU を有するマルチ ALU プロセッサ(MAP)の設計を行い、FPGA ボード上での動作検証と評価を行うことを目的とする。MAP では ALU 同士の並列演算と連鎖演算を可能として、高速化を図る。

本稿では MAP システムの構成、ALU2 個の MAP の設計、MAP の FPGA ボード上への実装とデバッグ、ブース乗算プログラムなどの並列性の評価について述べる。

2. MAP システム

2.1 システムの構成と機能

図 1 に MAP システムの構成を示す。ソフトウェア分野では、まずアセンブリプログラミングを行って機械語を生成し、次にシミュレータで動作を確認する。ハードウェア分野では、HDL で MAP の設計を行い、シミュレーションで動作を確認する。次に、プロセッサデバッガ、MAP 本体、メモリなどをまとめて論理合成し、FPGA ボード上に実装する。ホスト PC 上のプロセッサモニタからプログラムとデータをロードし、各種デバッグコマンドを用いて、ボード上での動作を確認する。

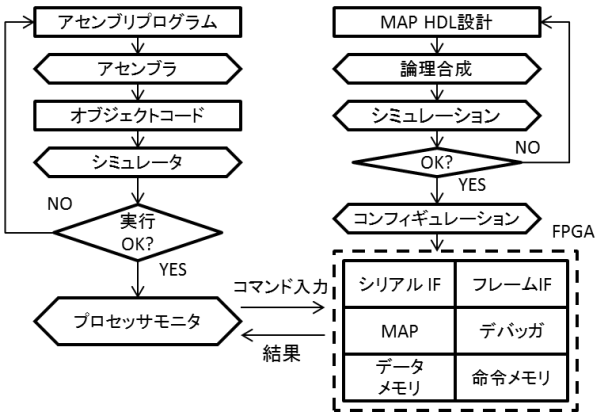

図 1：MAP システムの構成

MAP システムでは、複数 ALU による演算レベル並列性の有効性の検証を目的としている。MAP システムにおいて、二通りの検出方法をとる。一つ目はアセンブラによる静的並列性の検出である。これは、プログラムを上から下まで確認し、プログラム自体にある静的な並列性の可能性を検出する。二つ目はハードウェアによる動的並列性の検証である。これは、実際にプロセッサを動作させた際に並列性があるかどうかを判断するものである。静的、動的の並列性を検出し、複数 ALU による演算レベル並列性を検証する。

2.2 MAP シミュレータ

MAP シミュレータはプログラマが記述したアセンブリプログラムのデバッグ、および命令実行時の並列・連鎖性を調べる。MAP の複数 ALU による並列・連鎖演算、単一実行の有効性を示す役割を持つ。現在作成されたものは 2ALU のシミュレータである。図 2 に MAP シミュレータの動作を示す。オブジェクトコードを IM に、初期データを DM に設定する。命令実行時は IM から 1 命令分のコードを取り出し、命令のオペコードを解析する。取り出した命令の命令形式を判別し、それぞれの演算別に分岐させる。各分岐先で命令を解析し、レジスタ、即値、シフト量、ファンクションを判別し実行する。その後、結果を DM や RF に保存し、それを HALT まで演算を実行し続ける。最終的な演算結果が DM に格納される。

MAP シミュレータの並列・連鎖演算と単一実行判定は、２つの命令を比較し並列・連鎖性を判定する。アドレスの小さい方を上位命令、大きい方を下位命令とする。上位命令のオペコードを解析し、終了命令であれば単一実行とする。上位と下位にデータ依存がある場合は、上位の演算結果を下位でも使用できる連鎖演算とする。上位と下位に依存関係がない場合は同時実行とみなし、並列演算とする。このようにして並列・連鎖演算と単一実行判定を行う。

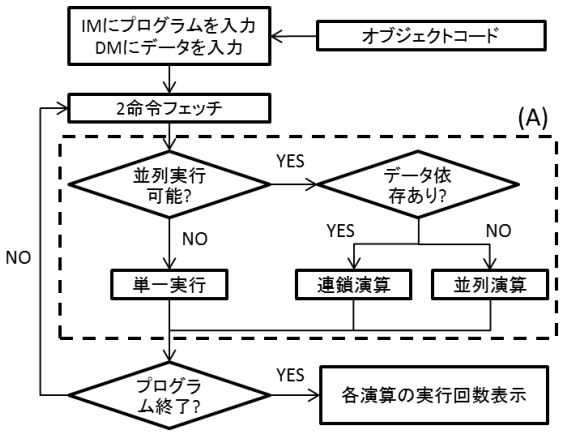

図 2：MAP シミュレータの動作

†立命館大学 大学院理工学研究科, Graduate School of Science and Engineering, Ritsumeikan University
‡立命館大学 理工学部, College of Science and Engineering, Ritsuemikan University

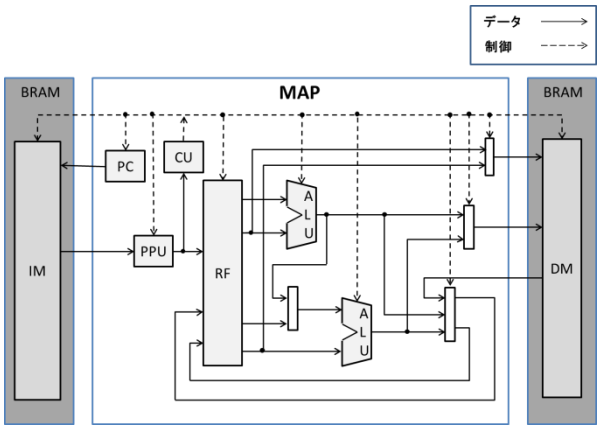

図 3：MAP のデータパス

3. マルチ ALU プロセッサ MAP

3.1 MAP の特徴

図 3 に MAP のデータパスを示す。MAP は複数 ALU を有する 32 ビットのプロセッサである。命令メモリとデータメモリが分離したハーバードアーキテクチャである。MAP の特徴は以下の 3 点に要約できる。

① 複数 ALU による並列処理が可能で、ALU 数は 2,4,8
② レジスタ数は 32 個で、全 ALU で共有
③ ALU で並列演算と連鎖演算を行う

3.2 MAP の命令セットアーキテクチャ

MAP には 37 個の命令があり、Register 形式（R 形式）、Immediate 形式（I 形式）、Long 形式（L 形式）、Jump 形式（J 形式）の 4 つの命令形式がある。R 形式ではレジスタ間の演算を行い、I 形式では即値演算を行う。L 形式は条件分岐命令とメモリ・レジスタのデータ転送命令である。図 4 に MAP の命令形式を示す。

ALU 数を増やすことにより、並列性を高めることができる。Op で命令を R 形式、I 形式、L 形式、J 形式と判断する。R 形式では、フィールド Fn で命令を詳細に識別しているため、Op1 つに対して 64 種類まで拡張が行える。J 形式のアドレスフィールドは 26 ビットである。

命令形式	32						命令種類
	6	5	5	5	5	6	
R形式	Op	Rs	Rt	Rd	Sha mt	Fn	ADD,SUB,AND,OR,XOR,NOT,NOP,SLT, SGT,SLE,SGE,SEQ,SNE,SLL,SRL,SRA,JR
I形式	Op	Rs	Rd	imm			ADDI,SUBI,ANDI,ORI,XORI,SLTI,SGTI, SLEI,SGEI,SEQI,SNEI,LDHI,LDLI
L形式	Op	Rs	Rd	address/immedi ate			BEQZ,BNEZ,LD,ST,JAL
J形式	Op	address					JUMP,HALT

図 4：MAP の命令形式

```
        SUB   $0   $0   $0   } 連鎖演算 ①
        LD    $1   0[$0]
        SUB   $3   $3   $3   } 並列演算 ②
        LD    $2   4[$0]
LOOP:   SUBI  $2   $2   1    } 連鎖演算 ③
        SGTI  $4   $2   0
        ADD   $3   $3   $1   } 並列演算 ④
        BNEZ  $4   LOOP
        ST    $3   8[$0]     ・・・単一実行
        HALT
```

図 5：乗算プログラム

3.3 MAP プログラミング

MAP では演算、ロード、ストア、分岐などを順に並べて逐次プログラムをユーザが記述する。プログラマは並列演算や連鎖演算を考慮せずに 1 演算ずつ記述していく。MAP は実行時に 2 演算ずつフェッチして並列、連鎖、単一を判定する。図 5 に MAP の乗算プログラムと、並列演算、連鎖演算、単一実行の例を示す。

3.4 ALU 並列演算

ALU 並列演算は、データ依存のない 2 つの演算を、1 命令サイクルで同時実行する。命令メモリからフェッチしてきた 2 演算で命令の判別を行い、並列実行できるか判断する。図 5 の②では SUB 命令と LD 命令、④では ADD 命令と BNEZ 命令が並列実行される。

3.5 ALU 連鎖演算

ALU 連鎖演算では、ある ALU での演算結果を他の ALU の入力とすることにより、依存関係のある 2 つの演算を 1 命令サイクルで実行する。図 5 の①では SUB 命令と LD 命令、③では SUBI 命令と SGTI 命令が並列実行される。

4. 2ALUMAP の設計

複数 ALU における演算レベル並列性をハードウェアで検証するため 2 個の ALU を有する MAP の設計を Verilog-HDL で行った。MAP の設計構成は、機能ごとにモジュールを作成し、設計した各モジュールをトップで統合した。モジュールは MAP のトップ、及び、PC、RF、2 個の ALU、CU、符号拡張、PPU、MUX、IM と DM である。IM と DM は BRAM を用いる。CU は MAP の動作制御と BRAM の制御を行う。

(1) レジスタファイル(RF)

MAP において複数 ALU が一つのレジスタファイルを共有しているのは重要な項目である。このレジスタファイルでは 4 ポート同時読み出しと 2 ポート同時書き込みが行われる。出力ポートはどれも独立しており、複数ポートが同じ読み出しアドレスを指定しても、動作に影響は出ない。演算が終了し、データが格納されるときは元のデータを上書きする。図 6 に示すように、まず、オペランドから読み出しアドレスを参照し、格納データを出力する。演算終了後、書き込みアドレスに演算結果を格納する。

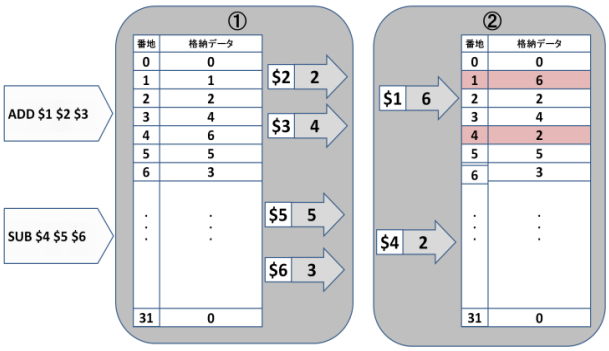

図 6：レジスタファイルの同時アクセス

(2) 並列検出ユニット(PPU)

PPU(Parallel Processing Unit)は並列性を検出するもので、図 2(A)の判定をハードウェアで行う。フェッチした 2 つの演算のオペコードを読みとり、2 演算が単一実行しか動作できないと判断した場合、一方の命令を NOP に変更し、1 命令のみ実行する。また、PC のアドレスが変わるため、その制御信号を送る。並列実行可能と判断した場合、並列演算と連鎖演算の判定を行う。一方の演算の出力レジスタが他方の演算の入力レジスタと等しい場合、連鎖演算を行う。データ依存がなければ並列演算が可能である。ALU 前のマルチプレクサで入力データを選択する。

(3) 2 ポート BRAM

FPGA ボードに実装する際、プロセッサの IM と DM は 2 ポート BRAM を用いている。BRAM の設計は ISE Core Generator を使用して、生成する。2 ポート BRAM では 2 命令同時読み出しが可能である。また、2 ポートを用いる利点として、各ポートが独立していることである。設計の際に制御しやすく、読み書きを同時に行える。これにより、DM においてロード命令とストア命令が同時に実現でき、並列性が向上する。さらに、従来のプロセッサデバッガにも対応でき、ポートごとに制御を振り分けることで容易に実現できる。

BRAM では読み出しのタイミングが 1 クロック分遅延する。ストア命令の際、誤ったデータが格納される場合がある。それを回避するため、CU の書き込みをクロックの立下りで行っている。

5. FPGA ボード上での実装

5.1 プロセッサデバッガ

MAP を FPGA ボードに実装する際に、プロセッサデバッガ・モニタを用いる。MAP、デバッガ、IM、DM、フレーム IF、シリアル IF をまとめて論理合成し、FPGA チップ上に実装する。プロセッサデバッガを Spartan-3A Starter Kit 上に実装した。図 7 にプロセッサデバッガ・モニタの構成を示す。

プロセッサデバッガは、ホスト PC 上のプロセッサモニタから要求される様々なデバッグコマンドを FPGA 内部で処理し、必要に応じてデータの送受信を行う。その利点は、FPGA ボードの LED や LCD、スイッチを使用せず、パソコン操作によって実機のデバッグを進められることである。デバッグコマンド一覧を表 1 に示す。

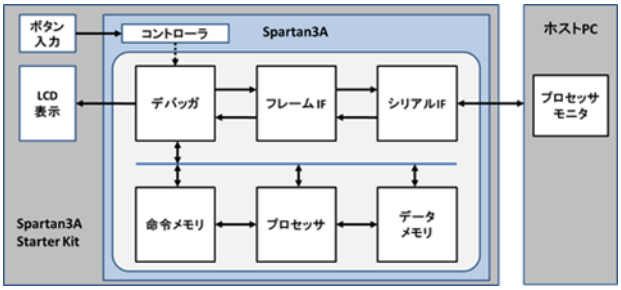

図 7：プロセッサデバッガ・モニタの構成

表 1：デバッグコマンド一覧

コマンド	ターゲット	入力	意味
reset	–	–	RF、PCの初期化
load	dm/im/rf/pc/bp	ファイルパス	ファイルからプロセッサモニタにロード
send	dm/im/rf	開始番地/終了番地	メモリ、レジスタ書き込み
set	dm/rf/pc/bp	対象番地/値	DM、RF、PC、BPの設定
read	dm/im/rf/pc	開始番地/終了番地	メモリ、レジスタ読み出し
save	dm/im/rf/pc/bp	ファイルパス	メモリ、レジスタの内容を保存
delete	bp	–	BPの削除
run	–	–	通常実行
run clk N	–	–	Nクロック実行
run bp	–	–	ブレイク実行

5.2 プロセッサモニタ

プロセッサモニタは、設計したプロセッサを FPGA 上で動作検証する際のツールとして利用する。ユーザからのコマンドを解析し、プロセッサの実行、メモリ・レジスタの読み書き等を実行する。

プロセッサモニタは CUI ベースのアプリケーションであったため、コマンド入力が煩雑で結果が見にくいという問題があった。そこで、GUI を設計することにより、これらの問題を解決した。図 8 に GUI 画面構成を示す。

今回は GUI を起動すると同時に CUI をバックグラウンドで起動させることによって、GUI の画面上から CUI に文字を入力できるプログラムを作成することで実現した。まず、GUI で実行するコマンドをラジオボタンやボックスに値を打ち込む。入力ボタンをクリックすると GUI は選択されたボタンと値を読み取り、それをもとに従来の CUI で使用されていた命令コマンドを文字列で作成し、CUI 側に送信する。CUI 側で受信した文字列が入力され、コマンドが実行される。図 9 にコマンドの出力画面を示す。ブ

図 8：GUI 画面構成

図 9：コマンドの出力画面

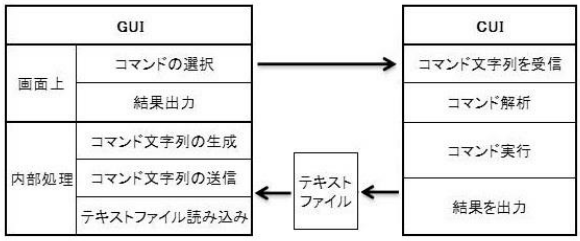

図 10：GUI と CUI の関係

ース乗算プログラムを送信し、初期値 2 と 3 を DM にセットし、実行して、結果 6 を読み出している。

実行結果はテキストファイルとして CUI から GUI に渡され、出力される。図 10 に GUI と CUI の関係を示す。

5.3 MAP 実装

表 2 に MAP の設計規模と動作周波数を示す。プロセッサの FPGA 使用率はスライス数が 78%、LUT 数が 74%、FF 数が 8%となった。最高動作周波数は 31.1MHz である。このプロセッサをプロセッサデバッガに接続し、実行を行う。MAP をプロセッサデバッガに接続するにあたり、Spartan-3A Starter Kit へ対応させるための pin 配置の変更、2 ポート BRAM の入出力の制御の変更、命令語長の違いによるプロセッサデバッガ内のバス幅の調節、プロセッサモニタの通信で、必要なデータの幅が変わるため、命令幅と RF 数の変更を行った。

連鎖演算ありの回路となしの回路の動作周波数を比較すると、前者は後者の 0.62 倍であり、連鎖演算を高速化するための改善が必要である。

表 2：MAP の設計規模と動作周波数

	スライス数	LUT数	FF数	動作周波数
連鎖あり	4593	8821	1057	31.1(MHz)
連鎖なし	4319	8250	1057	49.9(MHz)

表 3：2ALU の並列性

プログラム		連鎖あり												連鎖なし			
		命令数(個)						並列性(%)						命令数(個)		並列性(%)	
		動的			静的			動的			静的			動的		動的	
		並列	連鎖	単一	並列	連鎖	単一	並列	連鎖	単一	並列	連鎖	単一	並列	単一	並列	単一
三角形の判別		18	15	12	9	13	22	40	33	26	20	30	50	18	42	30	70
直線の方程式		5	10	9	33	17	6	20	42	38	40	31	29	5	29	15	85
ブース乗算	1次	5	14	12	2	7	6	16	45	39	13	47	40	5	40	11	89
	2次	7	13	4	4	13	6	10	61	29	17	56	26	7	30	19	81
平均		9	13	9	12	13	10	22	45	33	23	41	36	9	35	19	81

6. 並列性の評価

表 3 に 2ALU の並列性を示す。ブース乗算、直線方程式など複数の MAP プログラムを実行して、並列性の評価を行った。静的な並列性では、並列演算が平均して 23%を占めており、連鎖演算は 41%を占めている。合計で 64%を占めており、演算レベル並列性が有効であるといえる。また動的な並列性では、並列演算が平均して 22%、連鎖演算が 45%である。プログラムの実行時における演算レベル並列性は平均して 67%あり、2ALU による並列性が有効である。動的並列性では、特に連鎖演算が有効であると言える。

連鎖ありで実行した場合、実行命令数は 0.7 倍に減少するが、動作周波数が 0.62 倍に低下するので性能は低下してしまう。これを改善するために、連鎖演算とそれ以外の演算で周波数を変える、連鎖演算時のみクロック数を増やす、アセンブル段階で命令の順序を並べ替えるなどの機能を追加する必要がある。

また、2ALU で並列性を検証した場合、データ依存のある命令が多いことが判明した。ALU 数を増やし、命令間の依存関係を拡大して連鎖演算の有効性を検証する必要がある。

7. おわりに

本研究では、2ALU プロセッサ MAP を HDL で設計し、FPGA ボード上で実動作させた。そのために、従来のプロセッサデバッガを修正して Spartan3A ボード上に実装し、ブース乗算などの複数プログラムを実ボード上で動作させて、演算レベル並列性の有効性を確認した。

今後の課題として、連鎖演算実装の工夫、実行時のボード上での並列・連鎖演算の計測、アセンブラの命令移動による並列性の向上、4ALUMAP の設計などがある。

参考文献

[1] 中村、池田、小柳、山崎：プロセッサアーキテクチャ教育用ボードコンピュータシステムの開発、FIT2004、情報処理科学技術レターズ、LC-008、pp.51-52、2004.

[2] 難波、志水、山崎、小柳：プロセッサ設計支援ツールの設計・実装とハード/ソフト協調学習システムの評価、FIT2007、情報科学技術レターズ、LC-002、pp.39-42、2007.

[3] 境、山崎：FPGA ボード上でのマルチ ALU プロセッサの設計と実装、情報処理学会関西支部大会　ものづくり基盤コンピューティングシステム研究会、A-02、2011.