A-02

FPGA ボード上でのマルチ ALU プロセッサの設計と実装

Design and Implementation of a Multi ALU Processer on a FPGA Board

境　直樹†　　山崎　勝弘†
Naoki Sakai　　Katsuhiro Yamazaki

1. はじめに

近年の急速な半導体製造時術の発展により、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の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]～[8]。

図 1 に HSCS の学習フローを示す。学習者は MONI 命令セットを用いて、アセンブリプログラミングを行い、MONI シミュレータで実行する。これによりプロセッサアーキテクチャの知識を習得する。次に、MONI プロセッサを HDL で設計し、FPGA ボード上で動作検証を行い、基本的な設計能力を習得する。まず、基本命令セット MONI を設計し、それらのシングル、マルチ、パイプライン、スーパースカラプロセッサを設計した。次に、MONI を拡張した SOAR、SARIS、STRAD を定義し、それらの設計を行った。FPGA ボード上での実行を支援するプロセッサモニタとプロセッサデバッガも開発した。

本研究では、HSCS を用いて、複数の ALU による並列処理可能なマルチ 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。

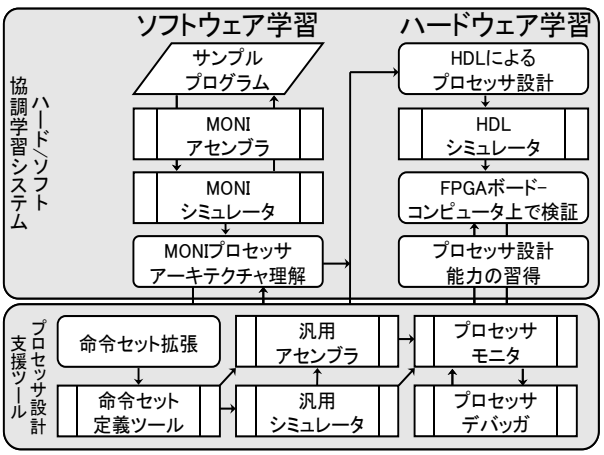

図 1：ハード/ソフト協調学習システム

2. マルチ ALU プロセッサ MAP

2.1 MAP の特徴とデータパス

MAP は複数 ALU を有する 32 ビットのプロセッサである。院生が設計したプロセッサを、FPGA チップ上で実際に動作させ、大学教育でプロセッサ設計能力を習得することが目標である。ALU が 2 個の場合のデータパスを図 2 に示す。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 レジスタファイルは 32 個で、全 ALU で共有
③ ALU で並列演算と連鎖演算を行う

例えば、レジスタ間演算命令では、命令メモリ(IM)から制御ユニット(CU)にオペコードとファンクションコードを送る。また IM から 2 つのオペランドのアドレスと、演算結果を格納するアドレスをレジスタファイル(RF)に出力する。ALU は、CU から演算命令の種類を受け取り、2 つの演算データの演算を行い、指定したアドレスに演算結果を格納する。PC は加算器によって 8 加算される。

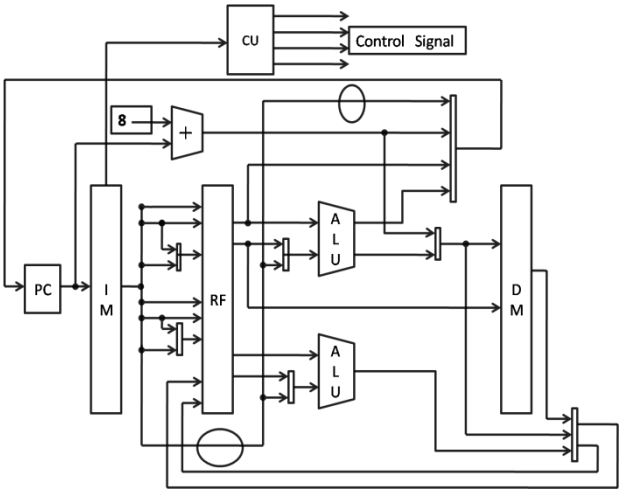

図 2：MAP のデータパス

2.2 MAP の命令セットアーキテクチャ

MAP には、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 つの命令形式を用意した。R 形式にはレジスタ間の演

† 立命館大学大学院　理工学研究科, Graduate School of Science and Engineering, Ritsumeikan University.

算を行う命令を定義している。I 形式にはレジスタ値と即値演算を行う命令を定義している。L 形式には条件分岐命令とメモリ・レジスタへのデータ転送命令を定義している。図 3 に MAP の命令形式を示す。

命令語長	32					
命令形式	6	5	5	5	5	6
R形式	Op	Rs	Rt	Rd	Shamt	Fn
I形式	Op	Rs	Rd	imm		
L形式	Op	Rs	Rd	address/immediate		
J形式	Op	address				

図 3：MAP の命令形式

ALU の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。

表 1：命令セット内容

R形式	ADD,SUB,AND,OR,XOR,NOT,NOP,SLT SGT,SLE,SGE,SEQ,SNE,SLL,SRL,SRA,JR
I形式	ADDI,SUBI,ANDI,ORI,XORI,SLTI,SGTI, SLEI,SGEI,SEQI,SNEI,LDHI,LDLI
L形式	BEQZ,BNEZ,LD,ST,JAL
J形式	JUMP,HALT

3. MAP プログラミングと並列処理

3.1 MAP プログラミング

MAP には 37 個の命令があり、詳細を表 1 示す。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 並列実行には並列演算と、ALU 間で依存関係があり連鎖させて実行できる連鎖演算がある。MAP は実行時に 2 演算ずつフェッチして並列、連鎖、単一のみを判定する。図 4 に MAP プログラム例を示す。このプログラムは整数同士による掛け算 A×B のプログラムである。例では、並列演算が 2 回、連鎖演算が 2 回、単一実行が 1 回実行される。

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

図 4：MAP プログラム例

3.2 ALU 並列演算

ALU 並列演算は、依存関係のない 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。また、2 演算のオペランドに依存関係がないか判断し、なければ並列演算を行う。

MAP の並列実行において図 4 のプログラムにおける並列関係を示す。並列演算においては、一方の ALU で{ADD $3 $3 $1}のレジスタ演算を行い、もう一方の ALU では{BNEZ $3 LOOP}の分岐命令を行う。連鎖演算においては{SUB $0 $0 $0}で$0 の値を初期化、{LD $1 MEM[$0+#0]}で初期化した$0+#0 のアドレスからロードする。これらの並列実行可能性をハードウェアで検出するが、MAP アセンブラでも演算やロードの順序を変更して、並列性を高めるようにする。

3.3 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。

PPU(Parallel Processing Unit)は並列実行を判断するものである。PPU ではフェッチした 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は 1 演算のみ動作させ、並列実行可能と判断した場合には連鎖判定にフェッチした 2 演算を送る。次に連鎖判定で並列演算と連鎖演算の判定を行う。連鎖判定で命令内のレジスタ指定フィールドから一方の ALU の演算結果を他方の ALU で演算を行おうとしているのか判断する。つまり、演算に使われるレジスタが格納するレジスタと等しければ連鎖演算が可能と判断する。連鎖演算が可能であると判断したとき、ALU の前にあるマルチプレクサ(MUX)で ALU の結果を演算に使用できるように指定する。

例えば、{ADD $1 $2 $3}と{SUBI $4 $1 #1}を同時に計算する場合、$1 を介して連鎖演算を用いて 1 命令で実行する。図 5 に連鎖演算時のデータパスを示す。IM から PPU に 2 つの演算命令を送る。ここで ADD と SUBI の命令同

士で並列実行が可能と判断され連鎖判定に命令を送る。連鎖判定内でアドレスデータから ADD の計算結果を SUBI の計算で使用すると判断し、連鎖演算が可能とされ、ADD の計算結果を使用できるように MUX へ信号を送る。RF から流れてくる演算データを ALU で計算するが、SUBI は連鎖演算なので ADD 計算結果をもらってから出力する。

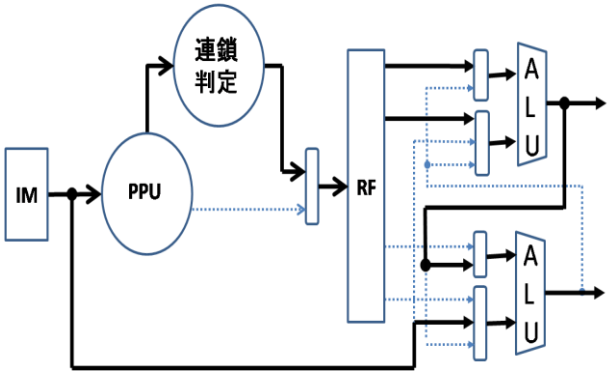

図 5：連鎖演算部のデータパス

4. FPGA ボード上での実装

4.1 プロセッサデバッガ・モニタ

FPGA ボードに実装する際に、HSCS で開発されたプロセッサデバッガ・モニタを用いる。プロセッサデバッガは、HSCS を利用して設計したプロセッサを、実機上で動作検証するためのハードウェア IP である。本モジュールと接続して論理合成をかけることで様々なプロセッサが FPGA に実装でき、ホスト PC と通信することで実行が可能である。プロセッサデバッガの実装対象として Xillinx 社の Spartan-3 & 3E Starter Kit ボードを使用している。本研究では、Spartan-3A Starter Kit に実装できるように環境を整え、学習者が自由に設計できる環境を作る。

図 6 にプロセッサデバッガ・モニタの構成を示す。本モジュールが担当する処理は、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ことである。プロセッサデバッガの利点は、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。HSCS 上で学生が作成したプロセッサを FPGA ボードに搭載するために欠かせない周辺モジュールを提供する。FPGA やホスト PC の環境によって発生する問題ではなく、プロセッサに関わる問題に集中し、自作プロセッサの実装と動作検証に専念できる。

表 2：デバッグコマンド

コマンド	ターゲット	意味
send	dm/im/rf	メモリ、レジスタの書き込み
read	dm/im/rf/pc	メモリ、レジスタの読み出し
save	dm/im/rf/pc/bp	メモリ、レジスタの内容を保存
load	dm/im/rf/bp	ファイルからロード
set	pc/bp	PC、ブレイクポイントの設定
del	bp	ブレイクポイントの削除
list/init	dm/im/rf/pc/bp	メモリ、レジスタの表示/初期化
run		通常実行
run clk N		Nクロック実行
run bp		ブレイク実行

dm:データメモリ　im:命令メモリ　rf:レジスタファイル
pc:プログラムカウンタ　bp:ブレイクポイント

また学習者は、デバッグコマンドをプロセッサモニタから入力し、プロセッサデバッガへ指示を与える。表 2 にデバッグコマンドを示す。表 2 の他に停止(halt)、リセット(rst)、終了(exit)、help のコマンドがある。テストの流れとしては、load でファイルをロードし、send でファイルを書き込む。これにより命令メモリやデータメモリにデータを書き込むことができる。set でプログラムカウンタの値を 0 にし、run で命令メモリに書き込まれたプログラムを実行する。read により値が正しく書きこまれているか、プログラムが正常に実行したか確認することもできる。

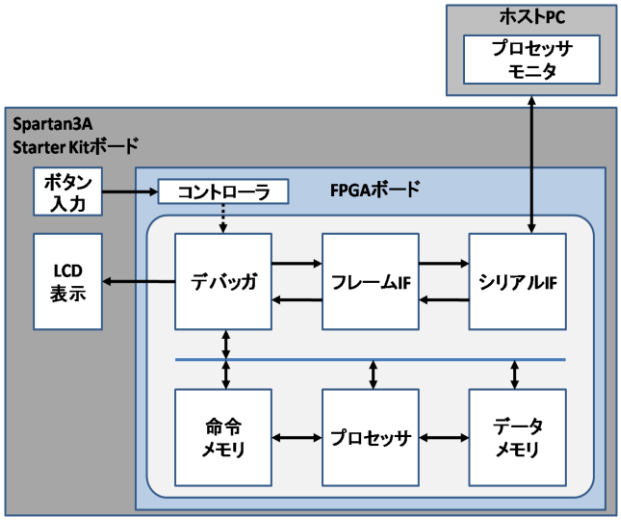

図 6：プロセッサデバッガ・モニタの構成

プロセッサデバッガにデバッグ対象のプロセッサを搭載するためには、インタフェースに準拠しなければならない。プロセッサデバッガで用意しているプロセッサの接続先は、データメモリ（DM）、命令メモリ（IM）、レジスタファイル（RF）、プログラムカウンタ（PC）である。DM と IM はプロセッサの外部との接続であり、必ず接続しなければならない。

4.2 MAP 用アセンブラ

図 7 に MAP アセンブラによる並列化を示す。

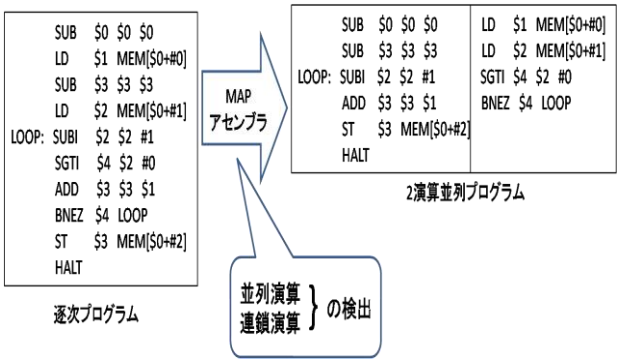

図 7：MAP アセンブラによる並列化

MAP アセンブラはプログラマが記述した逐次プログラムを並列化する並列化アセンブラである。並列演算と逐次演算の検出を行い、検出すれば並列化した機械語命令に変換する。また、検出時に演算順序や LD、ST 命令の順序を入れかえて、連鎖演算と並列演算の可能性を高められるように最適化する。ハードウェアによる並列演算の検出方法と、アセンブラによる検出方法を比較し評価を行う。

MAP アセンブラはアセンブリ言語で記述されたテキストファイルを入力とし、字句解析と構文解析を行い、機械語プログラムを出力する。まず Flex で字句解析を行い、ファイルのアセンブリ命令をトークンの並びに変換する。次に、Bison で構文解析を行い、トークン間の意味を解析していき、それに従って機械語を生成する。

MAP のアーキテクチャに従って定義した構文解析の定義ルールを説明する。MAP には 4 つの命令形式があり、アセンブリプログラムの記述から、命令形式を読み取るように構文解析の解析定義ルールの作成を行う。例えば、R 形式の「ADD $1 $2 $3」というコードは、「op REG REG REG」というパターンにあてはまる。あてはまった構文をその形式に従って機械語にしていく。解析中の段階では、分岐先のアドレスは未解決のため、分岐先アドレスは未定のまま機械語を作成していく。このとき、解析中に発見したラベルと、アドレス解決のためのラベルをそれぞれリストに追加していく。解決終了後に発見したラベルリストを参考にして、アドレス解決のリストのラベルの分岐先を訂正していく。最後に、機械語のプログラムのファイルを出力する。

5. おわりに

本稿ではマルチ ALU プロセッサ MAP のデータパス、命令セット、プログラミング、並列実行について述べた。MAP では、ユーザは演算の逐次プログラムを記述し、実行時に 2 命令ずつフェッチして、並列演算、連鎖演算を判定して実行する。また、MAP アセンブラにも並列性検出機能をもたせ、並列性検出をハードとソフトいずれで行うのが良いかを評価する予定である。

FPGA ボード上で動作させるためのプロセッサデバッガについても述べた。現在までに MAP の HDL 記述と論理合成、論理シミュレーションを行った。現在、Spartan3A Starter Kit へのデバッガの移植を進めている。

2ALU の MAP を Spartan 3A 上で実動作させ、プロセッサデバッガを用いてプログラムの検証を行える環境を整えることが当面の目標である。4ALU の MAP プロセッサの設計、MAP アセンブラの開発がさらなる課題である。

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、中村、小柳、山崎：マイクロプロセッサの設計と検証に基づいたハード/ソフト・コラーニングシステムの拡張、FIT2005、情報処理科学技術レターズ、C-001、pp.33-36、2005.

[3] Hoang Anh Tuan, K. Nakamura, S. Namba, K. Yamazaki and S. Oyanagi:A FPGA Based hardware/soufware Co-learning System, 信学技報、RECONF2005-48、2005.

[4] 難波、中村、Tuan、山崎、小柳：ハード/ソフト協調学習のための命令セット定義ツールとプロセッサデバッガの開発, FIT2006, N-009, 2006.

[5] 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、LC002、pp.39-42、2007.

[6] 井手、志水、山崎、小柳：学生によるプロセッサ設計実験に基づいたハード/ソフト協調学習システムの評価、FIT2008、C-009、2008.

[7] 井手、志水、山崎：ハード/ソフト協調学習のためのコンパイラ開発の検討、情報処理学会第 71 回全国大会論文集、2K-3、2009.

[8] PISHVA JOHN CYRUS P、井手、山崎：ハードソフト協調設計のためのコンパイラ学習システムの設計と実現、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-10、2010.