4K-6

PISA ベース VLIW プロセッサの FPGA による試作

三村 貴志 † 中島 伸吾 † 横田 隆史 † 大津 金光 † 馬場 敬信 †
† 宇都宮大学工学部情報工学科

1 はじめに

プログラムの命令レベル並立実行を行なうアーキテクチャの１つとして, VLIW(Very Long Instruction Word) アーキテクチャがある. VLIW アーキテクチャは, プログラムの命令コードを静的に解析して命令間の依存関係を調べ, 同時に実行可能な命令を 1 つの VLIW 命令としてまとめて実行する事で命令レベル並列性を抽出する. スーパースカラアーキテクチャと比べ, 動的な命令スケジューラを持たないため, 簡素な HW 構成で実現する事が出来る. そのため, VLIW アーキテクチャを採用したプロセッサは動作周波数の向上や省電力化を実現する事が容易である.

本稿では VLIW アーキテクチャを採用し, 命令セットには簡素且つ拡張性のある MIPS ベースの RISC 命令セットである PISA(Portable Instruction Set Architecture) 命令セット [1] を用いた PISA-based VLIW Processor(PVP) の設計について述べる. また, PVP を FPGA に実装して実際にプログラムを動作させるために必要となる周辺環境の構築について述べる.

2 PVP の仕様検討

まず, 1 つの VLIW 命令としてまとめられる PISA 命令数の検討を行なった。まとめる命令数が少ないとプログラムの並列性があまり抽出できないが, 多すぎても HW 資源を無駄遣いしてしまうという欠点がある. それを考慮し, PVP では最大で 4 つの命令を 1 つの VLIW 命令として扱うこととした. また, 動作周波数を向上させるため, パイプライン構造で実現することとした. パイプラインステージ数は PISA 命令セットの各命令のデータパスを考慮し, 命令実行の過程を 6 つに分割することで, 6 ステージとした。各ステージは, 命令をフェッチする IF(Instruction Fetch) ステージ, 命令をデコードする ID(Instruction Decode) ステージ, レジスタファイルからデータを読み出す RR(Register Read) ステージ, 演算を実行する EX(EXcution) ステージ, メモリアクセスを行なう MA(Memory Access) ステージ, レジスタファイルへの書き戻しを行なう WB(Write Back) ステージである.

実行対象プログラムのアセンブリコードは単一 VLIW プロセッサシミュレーション環境である CHAMEN の言語処理系 [2] を利用して取得することとした. VLIIW アーキテクチャでは, 連続した命令コードの中のどの命令を同時実行可能な命令としてまとめるのかをプロセッサが判断できなければならないのであるが, PVP では 1 つの VLIW 命令としてまとめる PISA 命令数の指定に, VLIW 命令の先頭 2 ビットを利用する. 先頭 2 ビットが"00"ならば 1 命令, "01"ならば 2 命令, "10"ならば 3 命令, "11"ならば 4 命令で 1 つの VLIW 命令を構成する.

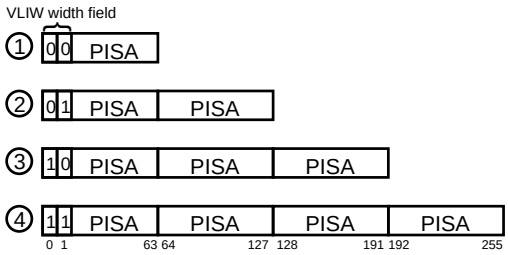

図 1: VLIW 命令形式

また, 入出力機能は HW の簡素化を考慮し, 専用の HW 機構を必要としないメモリマップト I/O で実現することとした.

さらに, PVP ではデバッグ支援のために以下の 6 つの例外に対応することとした.

- 未定義命令の実行：未定義命令をデコードした際に発生.
- VLIW 命令バンドル違反：同時に発行してはならない命令が 1 つの VLIW 命令中に含まれていた際に発生.
- オーバフロー発生：加減算でオーバフローが起こった際に発生.
- 0 による除算：除数が"0"である除算を実行しようとした際に発生.
- 分岐アドレスの語境界違反：分岐アドレスが語境界に整列していなかった際に発生.
- 保護領域違反：主記憶領域の読み出し専用領域に対して store 命令が発行された際に発生

3 PVP の回路設計

PVP の回路設計について述べる. 拡張性の向上や仕様変更の容易性を考慮し, それぞれのパイプラインステージにおいて主要な機能を独立したモジュールで実現することとした. 設計した PVP の回路概略図を図 2 に示す.

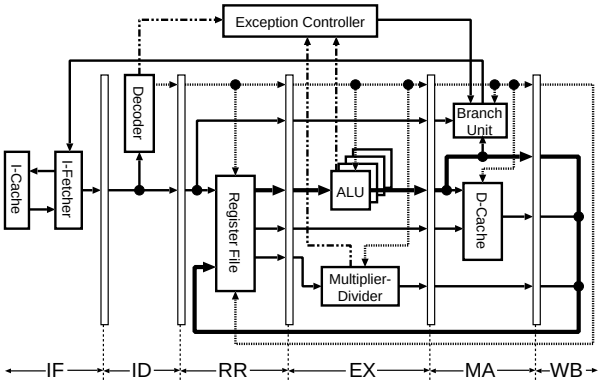

図 2: PVP の回路概略

図中の各コンポーネントの機能は以下の通りである.

- I-Cache：命令キャッシュ. 実行すべき PISA 命令コードが格納される.

Prototype Design of a PISA-based VLIW Processor using FPGA

†Takashi Mimura, Shingo Nakajima, Takashi Yokota, Kanemitsu Ootsu and Takanobu Baba
Department of Information Science, Faculty of Engineering, Utsunomiya University (†)

- I-Fetcher：I-Cache から PISA 命令コードをフェッチし, VLIW 命令として発行する.
- Decoder：I-Fetcher から発行された VLIW 命令をデコードし, 各種制御信号を生成する.
- Register File：0~ 31 までの汎用レジスタの読み書きを行なう.
- ALU：加減算・論理演算・シフト演算を行なう. 同機能のものが 4 つ並べられる.
- Multiplier-Divider：乗除算を行なう.
- Branch Unit：分岐アドレスの生成と分岐成立判定を行なう.
- D-Cache：データキャッシュ. プログラムで用いられるデータが格納される.
- Exception Controller：例外発生を検知し, 発生した例外に対応する処理コードの先頭アドレスを生成する.

4 周辺環境の仕様検討

PVP を FPGA に実装し, 実際にプログラムを実行するためには, 主記憶や入出力機構等の周辺環境を構築する必要がある. しかし, 周辺環境は実装対象の FPGA ボードの仕様に依存してしまうため, FPGA ボードの仕様を考慮した上で構築しなければならない.

本稿では Dini Group 社の DN3000K10[3] を対象 FPGA ボードとして周辺環境の構築を行なった. DN3000K10 は Xilinx 社の FPGA である XC2V6000 が 5 個搭載され, FPGA 間は共有バスで接続されている. また, PCI バスインターフェースと RS232C によるシリアル通信インターフェースを有している. さらに 168 ピン DIMM コネクタがあり, 本研究では 1GB の PC100 SDR-SDRAM を使用している.

PVP で実際にプログラムを実行するために最低限必要となるのは, 主記憶と入出力機能である. そのため, 主記憶を SDRAM が担当し, 入出力機能をシリアルポートで実現することとした. また, 実行するプログラムのバイナリコードを主記憶にロードする機能が必要である. これは PCI バスを利用し, ホストとなる PC で用意したバイナリコードを PCI バス経由で直接 SDRAM に書き込むことで実現することとした. また, PVP 上でのプログラム実行の開始/中断をホスト側から制御する機能を実装することとし, これはプログラムのロードと同様に, PCI バス経由で行なえることとした. そして, 高い拡張性を持たせるために, これらの機能は各々独立したコンポーネントで実装し, それらをバスで結合することとした. バスは WISHBONE バスを使用し, 結合方式は FPGA 間の共有バスを有効利用するために共有バス結合方式とした.

5 周辺環境の設計

仕様が決定した後, 必要となる周辺回路コンポーネントの設計を行った. PVP と周辺回路を含んだ PVP 動作実験環境の概略図を図 3 に示す.

図中の各コンポーネントの機能は以下の通りである. なお, PVP・I-Cache・D-Cache については既に説明済みのため省略する.

- Host PC：ホストとなるマシン. PCI スロットを持っているものを使用する.
- PCI Bridge：PCI バスと 32 ビット WISHBONE バスを繋ぐバスブリッジ. OPENCORES[4] の IP を使用する.

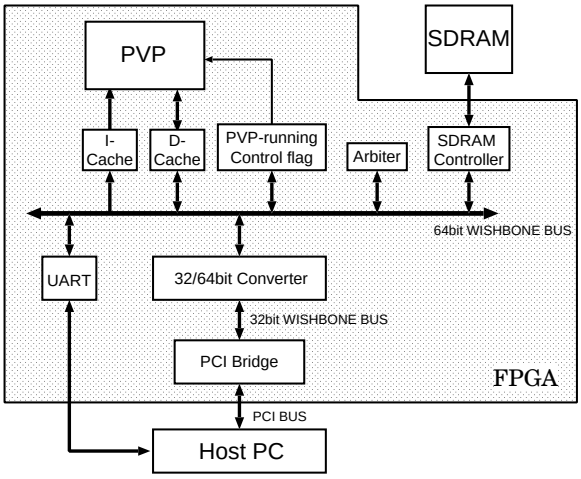

図 3: PVP 動作実験環境

- 32/64bit Converter：PCI Bridge 出力の 32 ビット WISHBONE バスをメインバスである 64 ビット WISHBONE バスに接続するための変換器.
- UART：シリアル通信の制御を行なう.
- PVP running Controller：PVP でのプログラム実行の開始/中断を制御するための機構. PCI バス経由で"1"を書き込むことで実行が開始され, "0"を書き込むことで中断する.
- Arbiter：メインバスである 64 ビット WISHBONE バスの調停を行なう.
- SDRAM Controller：バスからの要求に応えて SDRAM の読み書きを行なう.

6 おわりに

本稿では PISA ベース VLIW プロセッサの設計を行った. さらに, 設計したプロセッサを FPGA に実装して実際にプログラムを動作させるために必要となる周辺環境の構築を行なった.

今後の課題として, 本稿で設計した PVP と周辺環境を実際に FPGA に実装してプログラムを動作させることや, PVP を複数使用したマルチプロセッサの開発が挙げられる.

謝辞 本研究は, 一部日本学術振興会科学研究費補助金（基盤研究 (B)18300014, 同 (C)16500023, 若手研究 (B)17700047）および宇都宮大学重点推進研究プロジェクトの援助による.

参考文献

[1] D. Burger, T.Austin: "The SimpleScalar Tool Set, Version 2.0," Univ. of Wisconsin-Madison Comuter Sciences Department Technical Repot # 1324(1997.6)

[2] 月川 淳ほか: "メタレベル最適化計算機システム YAWARA のシミュレーション環境 —PISA をベースとした VLIW アセンブラの開発—", 情処第 67 回全国大会, 2005.3

[3] "DN3000k10 User's Manual VERSION 1.65," The Dini Group, 2003.6.11

[4] OPENCORES: http://www.opencores.org/