Vol.2014-ARC-208 No.7
2014/1/23

JAIST23-Pro: FPGA用マルチコアプロセッサの設計

Fengxiang Xie[1,a)]　田中 清史[1,b)]

概要：本プロジェクトでは，第1回ハイパフォーマンスプロセッサ設計コンテストへの参加を目的とし，FPGA（Spartan-6）上で動作するプロセッサ JAIST23-Pro を設計した．JAIST23-Pro は MIPS 命令群のサブセットを実行する 8 つの計算コアを搭載するマルチコア構成である．本稿では JAIST23-Pro の概要を示す．

1. はじめに

FPGA（Field Programmable Gate Array）デバイスの発展は著しく，1980 年代の誕生当時は数千ゲート相当の規模であったが，現在では数千万ゲート相当の論理を搭載可能な集積規模にまで成長している．第 1 回ハイパフォーマンスプロセッサ設計コンテストのターゲットデバイスである Spartan-6 においても，ロジックセル数が 43,661[*1]であり，マルチコアプロセッサを実現することが可能な規模であるといえる．このことから，著者は MIPS 命令セット [2] を実行する計算コア（JAIST23-Core）を設計し，それを複数接続することによりマルチコアプロセッサ JAIST23-Pro を実現した．

2. JAIST23-Pro：マルチコアプロセッサ

図 1 に JAIST23-Pro の構成を示す．本プロセッサは，後述する 8 つの計算コア（JAIST23-Core），各コアに 4KB の命令メモリ（I-MEM，Block RAM を使用）と 16KB（コア 0 のみ 32KB）のデータキャッシュ（D-Cache，Block RAM を使用），コア間でのデータ共有と同期を可能とする共有レジスタファイル（32 ビットレジスタを 8 本と 4 ビットレジスタを 8 本），8 つのポートを持つ DDR2-SDRAM インタフェース，およびシリアル転送モジュール（UART-Tx，UART-Rx）から構成される．シリアル通信（UART-Rx）によってプログラムを各命令メモリにロードすることにより，本プロセッサは MIMD（Multiple Instruction stream，Multiple Data stream）として動作する．コア 1〜7 のデータキャッシュは，ソフトウェアからの設定によりコア 0 が使用するスクラッチパッドメモリ（SPM）として再構成することが可能となっている．

2.1 JAIST23-Core：計算コア

計算コアである JAIST23-Core は，MIPS32 命令セットを実行する．実装した命令は**表 1** に示す 30 命令である．今回のコンテストの対象プログラムは全て int 型整数のみを扱うため，4 バイト以外のデータサイズを対象とした LB（Load byte），LH（Load half word），SB（Store byte），SH（Store half word）は複雑さの軽減のために実装対象外とした．

図 2 に示すように，命令実行方式は 5 段パイプラインであり，遅延分岐（1 命令のディレイスロット）を実現する．構成部品として，プログラムカウンタが 1 つ，32 個のレジスタを持つレジスタファイルが 1 つ，2to1 マルチプレクサが 4 つ，3to1 マルチプレクサが 2 つ，乗算で使用される HI と LO レジスタが各 1 個ずつ，ALU が 1 つ，加算器が 2 つ，フォワーディングユニットが 1 つ，符号拡張ユニットが 1 つ，各コントロールシグナルを生成する制御ユニットがある．ロード命令に関するデータハーザードを検出するユニット [3] も設計したが，利用するコンパイラはロード命令に関するパイプラインストールを

表 1 実装命令

算術論理	メモリ参照	その他
OR, ORI	LW	BEQ, BNE
AND, ANDI	SW	BGTZ, BLEZ
XORI		BGEZ, BLTZ
ADD, ADDI		SLT, SLTU
ADDU, ADDIU		SLTI, SLTIU
SUBU		MFHI, MFLO
SLL		J, JR
SRA		JAL
MULT		

[1] 北陸先端科学技術大学院大学
JAIST, Asahidai 1-1, Nomi, Ishikawa 923–1292, Japan
[a)] s1210028@jaist.ac.jp
[b)] kiyofumi@jaist.ac.jp
[*1] 6 入力 LUT 数の 1.6 倍の比率で換算 [1].

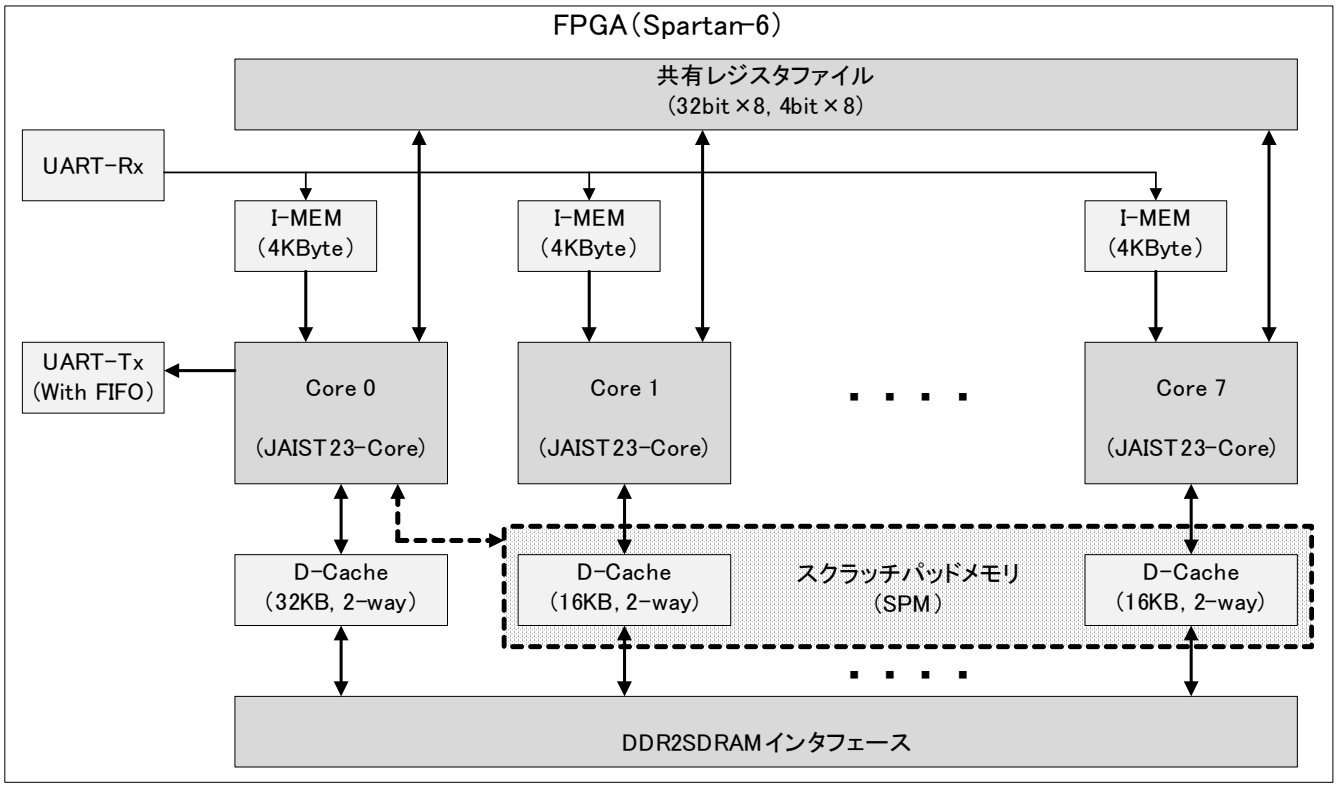

図 1 JAIST23-Pro のブロック図

発生させる依存関係を生成しないため，JAIST23-Core では，複雑さの軽減のために，当該データハーザード検出ユニットを削除した．分岐命令に関する依存関係については（例えば，第 2 ステージの命令が “BEQ　s1,s2,ADDR” 第 3 ステージの命令が “ADD s1,t1,t” の場合，あるいは第 2 ステージが”BEQ s1,s2,ADDR”，第 4 ステージが “LW s1,NUM(t1)” の場合），設計初期段階ではこれに対処するためのフォワーディングユニットを設計したが，これについても複雑さの軽減のために削除し，依存関係が発生した場合には，1 サイクルのパイプラインストールを発生させる選択をした．また，乗算命令の実行について，第 3 ステージで 1 サイクルストールする方式を選択することで乗算器による遅延に対処した．

2.2 命令メモリ

コンテストで対象となる 4 つのプログラムに対し，ソースプログラムを一部変更することにより実行バイナリサイズの縮小を図った。これにより，全プログラムでコードサイズを 4KB 以下にすることができ，これに合わせて命令メモリのサイズを 4KB とした．

2.3 データキャッシュ（スクラッチパッドメモリ）

各コアはコア専用の 16KB（コア 0 のみ 32KB）のデータキャッシュを使用する．ブロックサイズは 16B，連想度は 2 ウェイであり，置換選択方式は LRU に従う．プロセッサのリセット時にハードウェアによりタグの初期化を行う．ライトバック方式を採用したが，スヌープ機能は実装していないため，キャッシュ間でのデータの一貫性については，更新した対象ブロックをソフトウェアからフラッシュすることにより対処する．

コア 1〜7 用のデータキャッシュは，ソフトウェアからの設定により，コア 0 が使用するスクラッチパッドメモリ（SPM）として再構成することが可能となっている．これにより，並列化されていないプログラムに対してコア 0 が 16KB × 7=112KB のチップ内高速メモリを使用することが可能となる．

2.4 共有レジスタファイル

8 つのコア間でデータの授受と同期を実現するために 8 つの 32 ビットレジスタと 8 つの 4 ビットレジスタから構成される共有レジスタファイルを持つ（図 1）．これらのレジスタはメモリアドレス空間上にマップされており，メモリ参照命令（LW，SW）によって読み書きされる．32 ビットレジスタは主にデータの授受を目的としており，4 ビットレジスタは同期機構での使用（フラグとしての使用やバリア同期）を目的としている．

2.5 DDR2-SDRAM インタフェース

ボード上に搭載されている DDR2-SDRAM[4] に対して，16 ビット × 8 バーストアクセスを利用することにより，

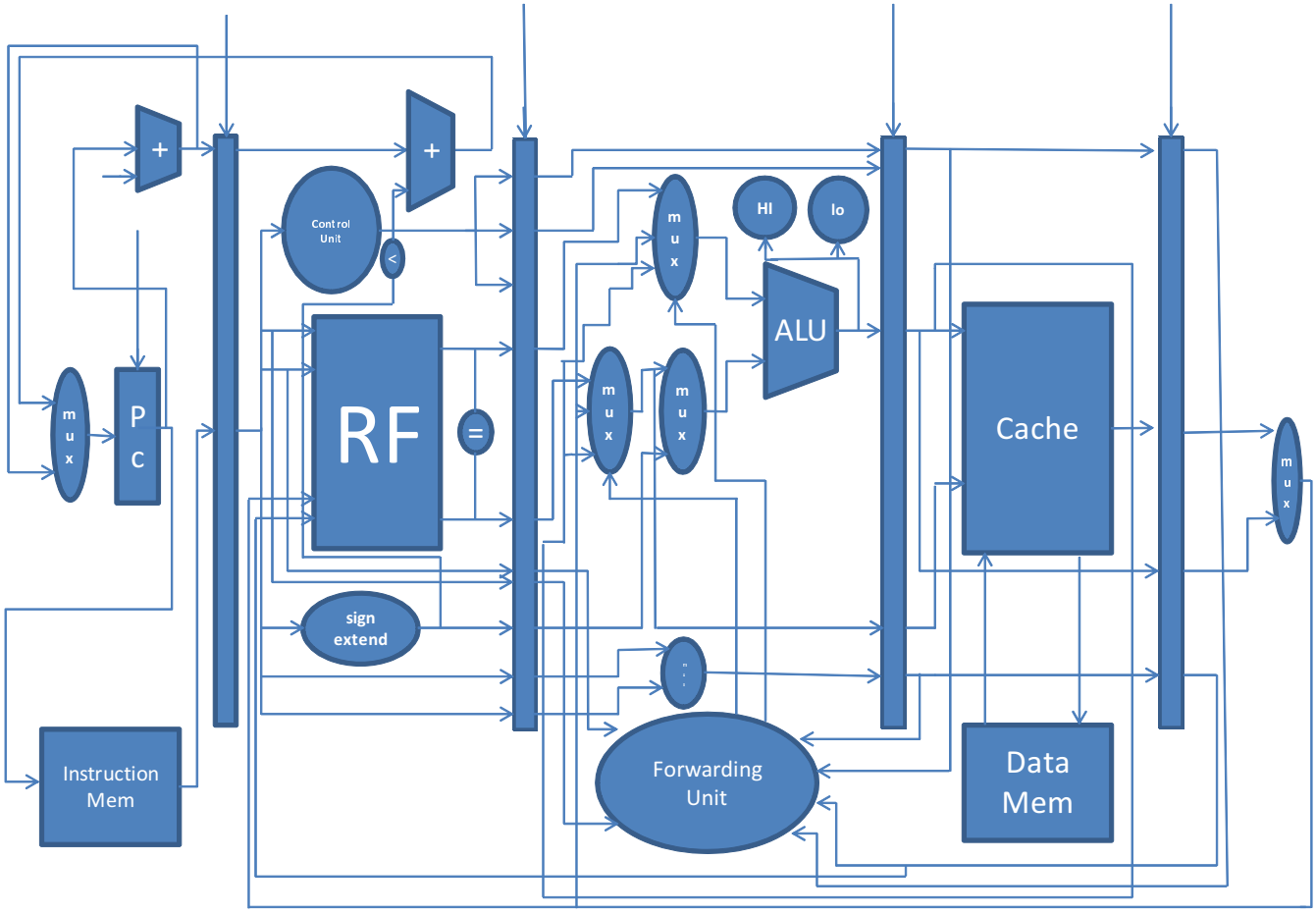

図 2 JAIST23-Core のブロック図
Fig. 2 Block Diagram of JAIST23-Core.

データキャッシュのブロックサイズでの転送を可能としている．複数のコアからのアクセス[*2]が競合した場合はラウンドロビン調停を行う．

2.6 シリアル転送モジュール（UART-Tx/Rx）

UART-Rx は 8 つコアの命令メモリへのプログラムロードに使用される．ホストからの転送データの最初の 4KB が各命令メモリに書き込まれる．UART-Tx はコア 0 による出力用で，ハードウェアによるフロー制御を行う．また，コア 0 からの書き込み用の FIFO を持つことにより，書き込み時のストールを緩和する．

3. FPGA 内部リソース使用

Spartan-6 をターゲットとしてインプリメントした JAIST23-Pro の FPGA 内部リソース使用（スライスレジスタ，スライス LUT，RAMB16BWER，RAMB8BWER，および DSP48A1）について**表 2** に示す．

表 2 内部リソース使用

リソース	使用数（総数）	使用率
スライスレジスタ	7,724（54,576）	14.2 %
スライス LUT	15,387（27,288）	56.4 %
RAMB16BWER	90 （116）	77.6 %
RAMB8BWER	23 （232）	9.9 %
DSP48A1	58 （58）	100.0 %

4. アプリケーションプログラム

プロセッサ設計コンテストで使用される，C 言語で記述された 4 つのプログラムである 310_sort，320_mm，330_stencil，340_spath のそれぞれについて，適用した高速化方式を示す．

4.1 310_sort：ソート

8 つの計算コアがそれぞれソート対象データ配列の 1/8 ずつに対してクイックソートを行い，8 つのソート済みデータ列が生成される．その後，4 つのコアがそれぞれ 2 つのデータ列に対してマージを行い，4 つのソート済みデータ列が生成される．続いて，同様に 2 つのコアによるマージ実行により 2 つのデータ列が生成され、最後に 1 つのコア

[*2] コアのパイプラインから直接メモリをアクセスすることはなく，実際にはキャッシュがミスした場合（ライトバックを含む）にメモリがアクセスされる．

が 2 つのデータ列に対してマージを行い，計算を完了する．各フェーズの終了時には，各コアは計算結果をキャッシュ上でフラッシュし，レジスタファイルを利用したバリア同期を行う．

4.2 320_mm：行列積

行列 a と行列 b の積を求めるプログラムである．行列 a を（行方向に）8 つのブロックに分割し，各計算コアがそれぞれのブロックを使用して行列 b との積を計算する．各コアは計算終了時に計算結果をキャッシュ上でフラッシュする．（このフラッシュはコア 0 による結果表示のためである．）本プログラムは計算結果の検査のために，結果の行列 c に対して各要素の総和を計算するが，この総和計算部分も分割し，各コアで分割部分に対して総和計算を行い，レジスタファイルを利用してコア 0 に計算結果を渡し，コア 0 が最後に最終結果を求める．

4.3 330_stencil：ステンシル計算

行列データを持つ buf1，buf2 を行方向でブロック分割し，各コアに割当てる．各コアはイタレーション毎に担当する計算を行った後，計算されたブロックの最上位の行データ，最下位の行データをキャッシュ上でフラッシュすることにより，左右のコアとの計算結果の授受を可能とする．イタレーション毎にレジスタファイルを利用したバリア同期を行う．全イタレーション終了時に，各コアは計算結果をキャッシュ上でフラッシュする．（このフラッシュはコア 0 による結果表示のためである．）計算結果の検査用の総和計算は，各コアで分割部分に対して行い，レジスタファイルを利用してコア 0 に結果を渡す．

4.4 340_spath：最短路探索

本プログラムはコア 0 のみで実行する．データキャッシュの再構成機構を利用し，コア 1〜7 用の全データキャッシュ領域をスクラッチパッドメモリ（SPM）として使用する．対象グラフのエッジデータの全てを SPM にロードした後，エッジデータ参照は SPM 内で行われる．コンテストのルールにより，エッジ数の上限は 8192 であり，各エッジデータは 3 つの整数から成るため，全エッジデータのサイズは 8192 × 4B × 3=96KB であり，SPM の最大サイズ 16KB × 7=112KB 以下となる．一方，ノード数の上限は 2048 であり，各ノードデータは同様に 3 つの整数から成るため，全ノードデータのサイズは 2048 × 4B × 3=24KB であり，コア 0 のデータキャッシュサイズ（32KB）以下となる．したがって本プログラム実行は，全てのデータの参照に関して，初期参照時を除いてはチップ内で行うことが可能となる．

5. おわりに

本稿では，ハイパフォーマンスプロセッサ設計コンテストへの参加を目的として設計したマルチコアプロセッサ JAIST23-Pro の概要を示した．本プロセッサは MIPS サブセット命令群を実行するコア（JAIST23-Core）を 8 つ内蔵し，共有レジスタファイルによるデータ通信・同期が可能であり，データキャッシュの再構成によりスクラッチパッドメモリを提供することが特徴である．全体設計を通して，コアが実行する命令の選択や命令メモリのサイズ等，コンテストで使用されるプログラムに特化した各種の最適化を行った．

参考文献

[1] Spartan-6 FPGA コンフィギャブルロジックブロック・ユーザーガイド．XILINX.
[2] MIPS32 Architecture For Programmers Volume II: The MIPS32 Instruction Set. MIPS Technologies, Inc.
[3] D.A.Patterson, J.L.Hennessy,“Computer Organization and Design – The Hardware/Software Interface”, 4th Edition, Morgan Kaufmann Pub., 2008.
[4] http://www.trenz-electronic.de/fileadmin/docs/Digilent/ATLYS/P3R1GE4JGF.PDF (2013/12/11 アクセス)