4A-2

MIPSソフトプロセッサへの浮動小数点演算ユニットの実装と評価

藤枝 直輝 †　　吉瀬 謙二 †
東京工業大学 大学院情報理工学研究科 †

1 はじめに

FPGA(Field Programmable Gate Array) デバイスの広がりに伴い，ソフトプロセッサ (ソフトマクロのマイクロプロセッサ) の利用が広がっている．ソフトプロセッサで多くのアプリケーションを効率よく動作させるためには，浮動小数点演算ユニット (FPU) の存在が重要である．

我々は，本研究室で開発しているMIPSソフトプロセッサであるMipsCoreに対して，単精度のFPUを追加した．本稿ではその実装を示し，FPGAを用いてその評価をおこなう．

2 MIPSソフトプロセッサ MipsCore

本稿では，MIPSソフトプロセッサMipsCoreを対象とする．MipsCoreは，我々の開発したMIPSシステムシミュレータであるSimMips[1)]をベースとしており，その一部をVerilog HDLに移植することによって構築したものである．コンパクトな記述，可読性の高さ，カスタマイズの容易さが特徴である．その詳細な設計と実装については文献 [1] に述べられている．ここでは，文献 [1] に示す版と，今回の実装のベースとなる版との相違点を述べる．

今回の版ではより多くのメモリを利用するために，メインメモリとしてFPGA内部のブロックRAMではなく，オフチップのSRAMを利用する．このSRAMは8ビットアクセスであるため，MipsCore側もまた8ビット単位でメモリでアクセスするように変更を施している．また，メモリアクセスステージは2つのステージを統合している．この変更により，ステージの構成はfetch，decode，regfetch，execute，memaccess，writebackの6ステージとなり，fetchおよびmemaccessのステージは処理に5サイクルを要する．すなわち，1命令の実行にかかるサイクル数は通常の命令で9サイクル，ロード/ストア命令で14サイクル，乗除算では40サイクルとなっている．

その他，独自OSを搭載するための制御回路などが追加されているため，使用するハードウェア量が増加している．これらの詳細は本稿では扱わない．

3 FPUの実装

FPUはMipsCore本体と並行して動作する．decodeステージで，フェッチした命令が浮動小数点命令であるかを検出する．浮動小数点命令でなければ，以後の処理はMipsCore本体が通常通りおこなう．もし浮動小数点命令であれば，regfetchステージで浮動小数点レジスタから値を読み出し，executeステージで演算をおこなう．演算が終わるまでの間，FPUはビジー信号を送出する．MipsCore本体はビジー信号を検出している間，executeステージに留まりつづける．浮動小数点レジスタとメモリとの間のロード/ストアは，memaccessステージでMipsCore本体がおこなう．最後に，FPUはwritebackステージで計算した，あるいはメモリからロードした値を浮動小数点レジスタへと書き戻す．

浮動小数点演算にはXilinx社の浮動小数点IPコア[2)]を利用する．IPコアには加減乗除，平方根，比較，整数との変換，精度の変換の機能が提供されている．これらを適切に組み合わせることによって，MIPSアーキテクチャで定義されている浮動小数点演算命令を実現する．これらのコアにはレイテンシのサイクル数が規定されているので，コアの組み合わせにより命令の実行ステージのレイテンシが決定される．例えば積和演算命令の実行レイテンシは，乗算コアの6サイクル，加算コアの13サイクルに，演算結果をMipsCore本体の演算結果バッファに格納するための1サイクルとを合わせた，合計20サイクルとなる．

倍精度演算命令はハードウェアコストが相当に大きいため，単精度相当の精度で提供する．すなわち，対象となる全ての値をいったん単精度に変換してから演算をおこない，最後にその結果を倍精度に変換して格納する．この変換にかかる追加のレイテンシは，典型的には5サイクルである．したがって，例えば倍精度の積和演算命令の実行レイテンシは25サイクルとなる．また，倍精度の値のロード/ストア命令は8バイト (64ビット) のメモリアクセスとなるので，通常5サイクルでおこなうmemaccessステージの処理を，9サイクルかけておこなう．

整数への変換は，IPコアが最近接偶数への丸めのみをサポートしている一方，MIPSアーキテクチャには最近接遇数への丸めのほか，0，$+\infty$，$-\infty$ への丸めによる整数への変換命令も含まれている．これらの命令を実現するために，まず値を一度整数に変換してから浮動小数点値に再度変換し，その値と元の値とを比較して，元の値が整数であるかどうかを判定する．整数であれば元の値をそのまま変換し，そうでなければ元の値に0.5を加算 (負数の0への丸め，および $+\infty$ への丸め)，あるいは減算 (正数の0への丸め，および $-\infty$ への丸め) してから整数への変換をおこなう．なお，これらの命令の実行レイ

Implementation and Evaluation of an FPU on a MIPS Soft Processor
Naoki FUJIEDA† and Kenji KISE†
†Graduate School of Information Science and Engineering, Tokyo Institute of Technology

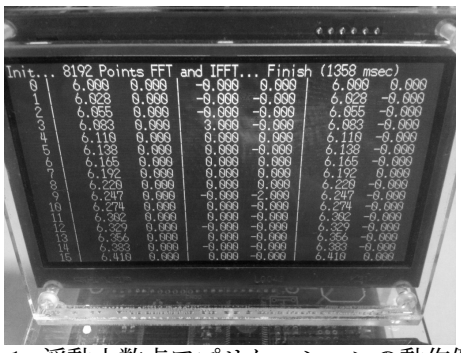

図 1: 浮動小数点アプリケーションの動作例.

項目	Original	w/ FPU
コード行数 (含コメント)	2,603 行	3,441 行
使用スライス数	2,264 個	4,654 個
使用ブロック RAM 数	5 個	7 個
使用乗算器数	0 個	2 個
最高動作周波数	54 MHz	54 MHz

表 1: FPU 追加前後のコード行数・ハードウェア量などの変化.

テンシは，整数との変換 6 サイクルを 3 回，比較 2 サイクル，加算 13 サイクル，演算結果格納の 1 サイクルの合計 34 サイクルとなる．

FPU のコード構成は MipsCore 本体の構成を踏襲している．すなわち，各ステージに関連する信号を，命令が通過する順番で記述している．またそれぞれのステージにおける記述も，MipsCore 本体との類似性に注意して記述している．これにより，コードの見通しの良さが確保され，理解やカスタマイズを容易にする．

4 FPU の評価

まず，FPU を追加したことによるコード行数・ハードウェア量などへの影響について評価をおこなう．MipsCore および FPU を，我々が開発している FPGA を用いた計算機システムに組み込み，これら全体について論理合成をおこない，ハードウェア量ついて評価をおこなう．また，動作周波数を変化させながら動作の可否をチェックし，最高動作周波数の評価をおこなう．この計算機システムは，SRAM，PS/2 キーボード，マルチメディアカード，出力用 LCD とのシリアル通信などのコントローラを含む．図 1 に，この計算機システムの上で，実際に浮動小数点命令を含むアプリケーションを実行させた例を示す．FPGA デバイスには Spartan-3E XC3S500E(50 万ゲート) を利用する．論理合成ツールには Xilinx 社の ISE 12.2 を用いる．

表 1 に，FPU 追加前後のコード行数・ハードウェア量などの変化をまとめる．“Original” が FPU を追加する前，“w/ FPU” が FPU を追加した後を表す．FPU を追加したことにより，Xilinx 社 FPGA の回路規模の尺度であるスライス数は 2 倍強となった．これは，XC3S500E で利用可能なスライスをほぼ全て使い切っていることになる．また，ブロック RAM を 2 個，乗算器を 2 個多く利用していることが確認できた．これらは浮動小数点レジスタと，浮動小数点の乗算 IP コアとに，それぞれ使用されたと考えられる．そして，動作周波数については FPU の追加前後で変化はなく 54MHz であった．これは，FPU の有無にかかわらず，SRAM へのアクセスがクリティカルパスとなっているのが原因であると考えられる．

項目	soft-float	hard-float
命令数	159.9M	7.79M
CPI	10.39	12.78
サイクル数	1661M	99.6M

表 2: アプリケーション (FFT) の実行結果.

次に，シミュレータを用いて FPU を追加した MipsCore がアプリケーションの実行に必要なサイクル数を計算する．シミュレーションには SimMips を利用する．命令の実行された回数を測定し，それに命令ごとのレイテンシを掛け合わせ，それらを合計することでアプリケーションの実行に要するサイクル数を求める．アプリケーションには高速フーリエ変換 (FFT) を用い，浮動小数点の処理を FPU でおこなうものと，ソフトウェアエミュレーションによりおこなうものとを比較する．コンパイラとして GCC 4.3.3 を利用する．

アプリケーションの実行結果を表 2 に示す．“soft-float” がソフトウェアエミュレーションで処理したもの，“hard-float” が FPU で処理したものである．浮動小数点命令は実行ステージのレイテンシが大きいために，CPI(Cycle Per Instruction) の値は約 23%増加した．しかし，浮動小数点命令の利用が可能であることで，実行命令数が約 1/20 に削減され，これらを掛け合わせた実行サイクル数は約 1/17 へと大きく減少していることが確認できた．

5 まとめ

ソフトプロセッサで多くのアプリケーションを効率よく動作させるため，我々は独自開発のソフトプロセッサ MipsCore に FPU を追加した．本稿では追加した FPU の実装について述べ，FPGA を用いた計算機システムに組み込んで評価をおこなった．今後は，並行して進められているその他の MipsCore 拡張とのマージを進めるとともに，実際のプロセッサアーキテクチャ研究にこれらのモジュールを活用していく．

参考文献

1) 藤枝直輝, 渡邉伸平, 吉瀬謙二. 研究・教育に有用な MIPS システムシミュレータ SimMips. 情報処理学会論文誌, Vol. 50, No. 11, pp. 2665–2676, 2009.

2) Xilinx Inc. *Floating-Point Operator v5.0.* Product Specification DS335, 2009.