1A-01

RoCC アクセラレータを用いた Rocket Chip による複素数乗算の高速化

津村 雄太† 山下 雄也† 岡田 正純† 丹所 良二† 外山 正勝†
三菱電機株式会社 情報技術総合研究所‡

1. はじめに

近年，オープンな ISA である RISC-V が普及しつつある．RISC-V はモジュール性の高い ISA であり、カスタム化した専用命令を実装することができる．RISC-V を実装した SoC の一つである Rocket Chip [1] は，RoCC(Rocket Custom Coprocessor)と呼ばれるインターフェースを有しており外部アクセラレータを簡易な方式で駆動することができる．この特長を活かして，Rocket Chip を用いたデジタル信号処理，特に高速フーリエ変換(FFT)の高速化を検討している．

本稿では，高速フーリエ変換のバタフライ演算に現れる複素数乗算のアクセラレータを試作し，ソフトウェアのみで実現した場合と比較した高速化可能性の評価について述べる．

2. RoCC の概要

RoCC を介した Rocket Core・アクセラレータ間接続の概略を図 1 に示す． Rocket Chip では，アクセラレータに対応しているカスタム命令が呼び出された場合，RoCC インターフェースを通してアクセラレータが呼び出される．具体的には，まずカスタム命令が呼び出され，RoCC インターフェースの cmd 信号からカスタム命令の引数情報がアクセラレータに渡される．複素数乗算実行後は，resp 信号から Rocket Core に演算結果が渡される．なお，アクセラレータのソースや Rocket Chip は Chisel(Scala ベースのハードウェア記述言語)を用いて実装されている．

3. 設計

3.1 外部設計

アクセラレータを呼び出すための RISC-V カスタム命令の形式 [2]について図 2 に示す．この形式に準ずる RISC-V 命令をソフトウェアから実行することで，RoCC アクセラレータを呼び出す．以下，今回試作した複素数乗算アクセラレータにおける各フィールドの意味を説明する．

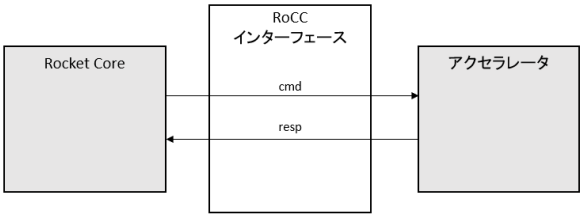

図 1 RoCC を介した Rocket Core・アクセラレータ間接続の概略

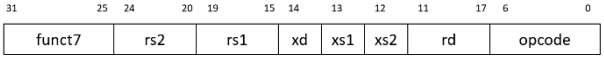

図 2 RISC-V カスタム命令の形式

opcode には，カスタム命令を表す値を指定する．本アクセラレータでは，0001011 を使用した．

rs1,rs2 はアクセラレータの入力データを格納している RISC-V レジスタの番号を指定し，rd はアクセラレータの出力を格納する RISC-V レジスタの番号を指定している．本アクセラレータでは，rs1,rs2 に複素数乗算の入力を指定し，rd に複素数乗算結果が格納される．なお，1 つの命令で受け渡し可能な入力データの最大数は 2 であるため，各レジスタの値は上位 32bit を実部，下位 32bit を虚部として扱うこととした．

xd,xs1,xs2 はカスタム命令に出力データ，入力データが存在する場合に立てるフラグであり，本アクセラレータでは入出力データがそれぞれ存在したためすべてのフラグを立てた．

funct7 はアクセラレータ作成者が任意の目的で指定することができる値である．本アクセラレータでは，この値を使用しないため，命令実行時は任意の値を指定する．

3.2 内部設計

作成した複素数乗算アクセラレータの概要図を図 3 に示す．複素数乗算アクセラレータでは，カスタム命令が呼び出されたときに，cmd 信号から与えられた値を上位 32bit と下位 32bit に

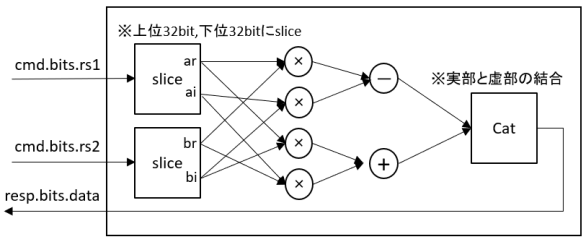

図 3 複素数乗算アクセラレータの内部構成

Acceleration of Complex Multiplication with Rocket Chip using RoCC Accelerator
†Yuta Tsunura, Yuya Yamashita, Masazumi Okada, Ryoji Tandokoro, Masakatsu Toyama
‡Information Technology R&D Center, Mitsubishi Electric Corporation

スライスする．続いてその値で複素数乗算を行う．その後，演算によって求められた実部と虚部を Cat メソッドで結合し，その値を resp 信号から Rocket Core へ渡す．

また，RoCC は Rocket Core・アクセラレータ間の同期を実現するための信号（ready と valid）を定義している．ready はスレーブがデータ受け取りの準備ができているか否かの信号であり，valid はマスターのデータが有効か否かの信号である．アクセラレータが呼び出されるまでは，図 4 の演算待機状態となり，Core はデータ受信の準備ができていることから，ready が 1 となる．その後，アクセラレータ内の演算が終了したタイミングで，図 5 の通り valid が 1 になり resp.data を通してデータがアクセラレータから Core へ渡される．今回試作したアクセラレータでは，これら同期制御を適切に実行できるように，valid/ready 信号を制御している．

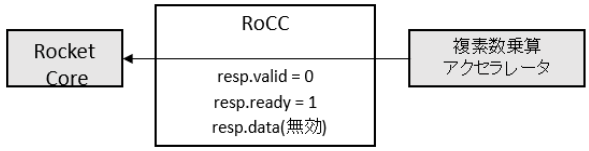

図 4　アクセラレータからの演算結果待機状態

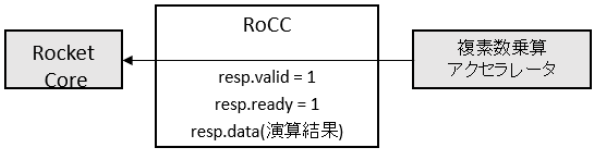

図 5　データ送信可能状態

4. 評価

4.1 評価用プログラム

rocket-rocc-example [3]を参考に Rocket Chip 用の実行バイナリを作成するための C プログラムを作成する．

作成したプログラムの一部を図 6 に示す．これは，アクセラレータを使用せずに複素数乗算を行うプログラム部とアクセラレータを呼び出すプログラム部からなる．計測には gettimeofday 関数を使用し，複素数乗算のループ回転数 LOOP_NUM は 100000 とする．なお，プログラム中の complexMul はアクセラレータを呼び出すためのマクロであり，インラインアセンブラとしてカスタム命令を呼び出す．このマクロの引数がインラインアセンブラに渡され，その情報が RoCC を通してアクセラレータへと渡る．

4.2 性能計測

3 章で作成したアクセラレータを含む Rocket Chip を ZedBoard [4]に書き込み，そのコア上で評価用プログラムを実行し，複素数乗算にかかる実行時間を計測した．最適化オプションとして，-O0 と-O2 の 2 パターンを用いて計測した．

```
//アクセラレータ無しの複素数乗算
  gettimeofday(&start_tv, NULL);
  for(int i=0;i<LOOP_NUM;i++){
    Q_r[i] = (a_r[i] * b_r[i]) - (a_i[i] * b_i[i]);
    Q_i[i] = (a_i[i] * b_r[i]) + (a_r[i] * b_i[i]);
  }
  gettimeofday(&end_tv, NULL);

//アクセラレータありの複素数乗算
  gettimeofday(&start_tv, NULL);
  for(int i=0;i<LOOP_NUM;i++){
    complexMul(rd_data[i], r1_data[i], r2_data[i]);
  }
  gettimeofday(&end_tv, NULL);
```

図 6　評価用プログラム

計測結果を表 1 に示す．この結果より，複素数乗算において RoCC アクセラレータを用いることで，最適化-O0 では 2.34 倍，最適化-O2 では 1.50 倍の速度向上が得られた．

本結果の妥当性検証のために，Rocket Chip 向けエミュレータを用いて最適化-O0 における推定実行時間を求めた．アクセラレータを使用した場合は 3.99ms，アクセラレータ未使用の場合は 12.87ms で，どちらも実行結果との差異が確認された．原因は今後調査予定である．

表 1　評価用プログラムの実行結果

	最適化-O0	最適化-O2
アクセラレータ使用	6.331ms	3.642ms
アクセラレータ未使用	14.832ms	5.481ms

5. まとめ

本稿では，RoCC を活用した FFT の高速化に向けた第一歩として，平易な題材である複素数乗算 RoCC アクセラレータを試作し，速度向上率を確認した．その結果，最適化-O0 では 2.34 倍，最適化-O2 では 1.50 倍の速度向上が達成された．

今後は，本計測結果の考察を行いつつバタフライ演算とビットリバースを RoCC アクセラレータ化し，FFT の高速化を目指す．

参考文献

[1] “rocket-chip,”. https://github.com/chipsalliance/rocket-chip.
[2] “The RISC-V Instruction Set Manual, Volume I: User Level ISA, Version 2.1”. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-118.pdf.
[3] “rocket-rocc-examples,”. https://github.com/seldridge/rocket-rocc-examples.
[4] XILINX. https://japan.xilinx.com/support/download/index.html/content/xilinx/ja/downloadNav/vitis/2021- 1.html.