オープンソースCPU「RISC-V」の研究

第11回 Myカスタム命令用演算器をFPGAに載せる

@msyksphinz

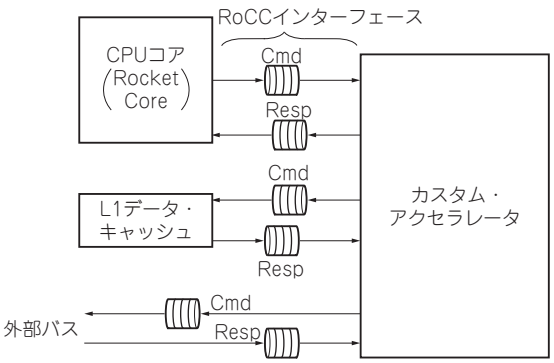

図1 RISC-VのRocket Chip実装でカスタム命令を実行するRoCCアクセラレータを追加する

前回はCPU自作の醍醐味としてMyカスタム命令の追加を行いました．今回はこれを踏まえ，Myカスタム命令用演算器をFPGAに実装します．（編集部）

行列積アクセラレータの機能の整理

いよいよChiselでRoCCアクセラレータ（**図1**）を開発します[注1]．RoCCアクセラレータは，大きく分けて3つの機能を持っています．

(1) 行列Aの列数＝行列Bの行数（＝M）を設定する．
(2) 行列Bの行数Kを設定する．
(3) 行列Aを行方向にM個分フェッチする．行列Bを列方向にM個分フェッチし，その結果を内積する．

(1)と(2)は命令のFunct値を使って区別しました．

(3)の実現方法ですが，2つのステート・マシンを用意して実装しました．それぞれのステート・マシンは以下のような仕様にしました．

● リクエスト発行ステート・マシン（cmd_state）

計算リクエストが渡されるとIdle状態から動作状態に移行します．$M \times 1.5$個のリクエストを発行します．発行した回数を記憶している以外に，0～2をカウント・アップするカウンタr_cmd_count_3を定義します．r_cmd_count_3はリクエストが発行されるたびにカウント・アップし，r_cmd_count_3==0の場合は行列Aの現在のアドレス，r_cmd_count_3==1, r_cmd_count_3==2のときは行列Bの現在のアドレスに対してリード・リクエストを，L1データ・キャッシュ・インターフェースを通じて発行します．

行列Aのリクエスト・アドレスは，+8バイトずつ，行列Bのリクエスト・アドレスは$+K \times 8$（行列Bの行方向にシークするため）ずつ加算していきます．

全てのメモリ・アドレスをリクエストすると，ステート・マシンはアイドル（Idle）状態に戻ります．

● メモリ・データ受信＆計算ステート・マシン（rcv_state）

L1データ・キャッシュ・インターフェースからデータを受信し，内積を計算するステート・マシンです．リクエスト発行ステート・マシンが$M \times 1.5$個のリクエストを発行しているわけですから，$M \times 1.5$個のデータが，行列A，行列B，行列Bの順番で返ってきます．受信した数をカウントする（r_recv_countレジスタを定義する）と同時に，0～2をカウント・アップするカウンタr_recv_count_3を定義します．r_recv_count_3==0の場合は行列Aの要素ですのでその値をレジスタに保存しておきます．r_recv_

注1：最新版のfpga-zynqリポジトリは2018年7月現在Rocket-Chipが正常にブートできなくなっています．筆者はfpga-zynqのリビジョンf03982e，内包しているRocket Chipリポジトリはリビジョンf3299aeで実験を行っています．

注：本稿の内容は執筆時点のもので，随時更新されていく可能性があります．
編集注：本誌2018年2月号特集2「新時代プロセッサ システム作り」ではバークレイ以外のさまざまなコアも紹介しています．