Appendix 1　5ステージ動作でベーシックなCPU設計の勉強にもなる

RISC-Vコアの HDL記述の実際

石原 ひでみ Hidemi Ishihara

リスト1　命令フェッチ（IF）部分（fmrv32im.v内）

```
/* ------------------------------------ *
 * Stage.1(IF:Instruction Fetch)        *
 * ------------------------------------ */
 assign I_MEM_ENA  = (cpu_state == S_
          WRITEBACK) | (cpu_state == S_IDLE);
 assign I_MEM_ADDR = pc;
```

● ファイル名の命名ルール

本特集で紹介しているRISC-Vは，今回の特集のためにスクラッチからHDLソースを書き起こしました．まさにFPGAマガジン版ともいうべき実装となっています．なお記述言語としては，筆者が得意なVerilog HDLを採用しました．

FPGA開発ツールのプロジェクト名やHDLソースの基本ファイル名としては，次の命名ルールを考えました．

- FPGAマガジンを示す“fm”
- RISC-Vを示す“rv”
- 32ビットを示す“32”
- 基本命令＋乗除算命令を示す“im”

以上を連結して“fmrv32im”としています．また，RISC-Vコアを構成する各部は“fmrv32im_xxxx”というファイル名を付けました．具体的には，例えば命令デコーダのHDLソースは，“fmrv32im_decode.v”となります．

● RISC-Vコアの各ステージ

第3章で説明したように，今回実装したRISC-Vコアの動作は，次に示す5ステージで構成し，この順で命令を実行していきます．

(1) 命令フェッチ（IF）
(2) 命令デコード（DE）
(3) 実行（EX）
(4) メモリ・アクセス（MA）
(5) 書き戻し（WB）

第3章ではHDLソースを示した各部の説明はしなかったので，ここで各ステージについて具体的な実装例をHDLソースを示しながら説明します．

なお各ステージのうち，(1) 命令フェッチと (4) メモリ・アクセスは，RISC-Vコア外部とのメモリ・アクセスのため，コアの内部に処理の主だったものはありません（fmrv32im_xxx.vという独立したHDLソースはない）．

● 命令フェッチ（IF）

命令フェッチは，単純にプログラム・カウンタが示す命令メモリ（実際はキャッシュ・メモリ）のアドレスから命令を読み込むステージで，命令デコードや命令実行のステージのように，モジュールにするほどの機能はありません．

命令フェッチの部分は**リスト1**のように，fmrv32im.v内で命令メモリに対してプログラム・カウンタを出力しているだけです．

● 命令デコード（DE）

命令デコードは，命令コードを入力して命令コードからrd，rs1，rs2のレジスタ番号とイミディエートの抽出，命令を判断して命令ごとの信号を出力します（**リスト2**）．

リスト2（b）でタイプの判別とイミディエートの生成を行っています．RISC-V仕様書のChapter 19のTable 19.1 RISC-V base opcode mapの中のinst[1:0]=11と，Chapter 2の2.2 Base Instruction Formatで示されている表と同じように構成しています．

リスト2（c）では命令タイプに応じて命令コードからレジスタ番号を抽出します．rs1，rs2のレジスタ番号はレジスタ・ファイルのレジスタを選択して実行ステージへ渡されます．rdは書き戻しのレジスタ番号に使用されます．また，この際に命令コードで存在しないレジスタ番号は0（x0）を示すようにしています．x0レジスタは常にゼロを示すので実行ステージや書き戻しで使用されても問題ありません．

● 実行（EX）

実行ステージは，命令デコードでデコードされた命令の演算部分を実行します．実行ステージは整数演算モジュール，乗算モジュール，除算モジュールの3つで構成されますが，**リスト3**のようにいたって簡単な演算回路になっています．

● メモリ・アクセス（MA）

メモリ・アクセスも命令フェッチと同じように，データ・メモリ（実際はキャッシュ・メモリ）への読み書きになるため，モジュールにするほどの機能はありません．命令フェッチのステージと同じように，fmrv32im.vで**リスト4**のようにデータ・メモリへの制御信号の接続になっています．