ご購入はこちら

オープンソースCPU「RISC-V」の研究

最終回
第13回 RISC-V と共に育った言語「Chisel」

@msyksphinz

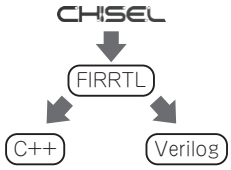

図1　Chisel で回路を記述することのメリット…
Chisel で記述することで論理合成のための Verilog コードやシミュレータ用の C++ ファイルを同時に生成できる．FIRRTL についてはコラム（Chisel → Verilog の間に生成される「.fir ってなにもの？」）を参照

Rocket Chip は Chisel という言語を使って記述されています．Chisel は RISC-V 実装の開発にあたり，一緒に開発されました．Chisel は Scala という関数型言語を拡張する形で定義されており，Rocket Chip は Verilog-HDL や VHDL を一切使わず，Chisel だけを使って記述されています．

つまり，Chisel を習得すれば，Rocket Chip の実装の中身をある程度読み解きながら解析を行うこともできますし，自分で拡張することもできるようになります．今回は RISC-V のために作られたハードウェア記述言語「Chisel」について見ていきましょう．

Chisel が生まれたきっかけ

ハードウェア記述言語 Chisel は，RISC-V のハードウェア実装 Rocket，BOOM やその周辺回路などを開発するために作られた言語です．

Chisel は，これまでのハードウェア記述言語を使って開発する際の問題を解決するために作られました．ハードウェア記述言語の問題とは，

- ハードウェア設計はソフトウェア設計に比べて時間がかかる
- 論理合成ツールなどが高価
- ハードウェア・シミュレーション・ツールが遅い

などです．Chisel はこれらのすべてを解決できるわけではありませんが，Scala というソフトウェア開発で利用される言語を利用することで，ある程度上記の問題に対する解決策を提示できます．

具体的には，

- シミュレーションなどは Scala 言語の体系のまま実行することで，ハードウェア・シミュレーション・ツールと比較して実行時間を短縮できる．
- ハードウェアの部分と，ソフトウェアの部分を同じ記述から生成することにより，仕様のアンマッチを防げる．

例えば，Chisel で記述された RISC-V の実装では，Chisel の記述から Rocket のハードウェア実装とともに，命令セット・シミュレータである Spike の実装（つまり C++ の実装）も出力できます．これにより，例えば新しい命令を実装したいとなると，Chisel でその仕様を記述すると，シミュレータからハードウェア実装まで作成できるようになります（**図1**）．

今回は，Chisel の基本的な使い方を見ていき，まずは簡単な回路を記述できるようになるまでを見ていきましょう．

Chisel 事始め

● 複数クロックの定義もできるので非同期 FIFO などの記述も行える

Chisel を体験するために，まずは簡単な例を見ていきましょう．**リスト1**は 32 ビットの符号なし数を 16 個受け取り，それらを 1 ステップずつ加算して総和を求めるプログラムです．

この reduction プログラムの実現したいことを**図2**に示します．16 種類のデータを受け取り，レジスタに格納します．

1 サイクルごとにそのレジスタをシフトしていき，先頭のレジスタにある値と，現在の総和を格納しているレジスタの値を加算し，もう一度総和レジスタに格

第1回　登場の背景（2018年2月号）
第2回　アーキテクチャの基礎知識（2018年3月号）
第3回　ハード＆ソフト開発環境あれこれ（2018年4月号）