応用編　第4章

RISC-V CPUコアの搭載とマトリクスLEDの制御

井田 健太

FPGAにRISC-Vコアを実装する

Tang Nano 9K（Sipeed）に搭載されているFPGA GW1NR-LV9（GOWIN Semiconductor）は，4入力ルックアップ・テーブルを8640個，フリップフロップを6480個持ちます．これは，小型のCPUコアを実装するのに十分なリソースです．実際に，オープンソースの省リソースなRISC-V CPUコアであるPicoRV32を組み込むサンプル・デザインがボードの製造元であるSipeedから公開されています．

ここでは，単にサンプル・デザインを動かすのではなく，PicoRV32を組み込んだシステムを作成する手順を説明します．

● 実装したCPUからマトリクスLEDを制御

本章では，共にFPGA内に実装したCPUと周辺回路とを組み合わせることによって，マトリクスLEDを制御します．マトリクスLEDは基礎編第4章で作成したものと同じ回路を使います．

マトリクスLEDの制御そのものは一般的なマイコンを使ってCPUからGPIOを操作する方法などでも実装可能です．しかし，その場合はマトリクスLEDの点灯状態を制御するためにCPUの処理能力をある程度使うことになります．

本章ではFPGAらしく実装します．CPUからはレジスタ経由でマトリクスLED制御回路の設定変更だけを行います．それを受けてマトリクスLED制御回路がLEDの点灯制御を行います．そのためマトリクスLEDの点灯制御中でもCPUは別の処理を行えます．

● FPGAにCPUを組み込む理由

そもそも，「全て専用の論理回路で構成するのではなく，CPUのコアを組み込んでシステムを実現する理由は何か」と疑問に思うでしょう．

本章では，マトリクスLEDの制御を行いますが，表示する内容として2パターン程度を交互に切り替えるくらいであれば，実際のところCPUは必要ありません．

では，表示パターンの内容をUART経由で受信し，PCからのコマンドによって切り替える処理を実装する場合はどうでしょうか．その場合でも，専用の論理回路を構成して実現することは可能です．しかし，モジュール間の状態の管理などがかなり面倒です．

こういった場合に有用なのが，PicoRV32などのFPGA実装可能なCPUコアです．FPGAにCPUコアを組み込み，複雑な状態管理をCPU上のソフトウェアで実装することによって，ハードウェアで実装するよりも格段に簡単になります．

実際のところ，GOWIN以外のFPGAベンダも，それぞれ自社FPGA向けに何かしらのソフトCPUコアと，その開発環境を用意しています．GOWINでは，今回使用するPicoRV32に対してGOWIN独自の変更を加えたものを提供しており，開発環境も用意されています．

また，PicoRV32はRISC-Vの命令セット規格に従ったCPUコアであり，CPUコアの実装自体に加えてコンパイラなどのツールチェーンを無償で利用可能です．

環境構築

● RISC-Vツールチェーンの導入

本章で作るシステムでは，FPGAに実装したCPU上で動作させるソフトウェアが必要です．ソースコードをコンパイルするために，RISC-Vのツールチェーン一式をPCに導入します．

▶ツールチェーンのダウンロード

リスト1の①のコマンドでRISC-VのGCCツールチェーンを公開しているGitHubリポジトリから，ビルド済みのRISC-Vツールチェーンをダウンロードして展開します．

▶正しく展開されたかを確認

`~/riscv`ディレクトリにいる状態で**リスト1**の②のコマンドを実行して，`bin`，`include`などのディレクトリが見えれば成功です．

最後にコンパイラのパスを通して，コンパイラのコマンド`riscv64-unknown-elf-gcc`を呼び出せるかどうかを確認しておきます（**リスト1**の③）．

以上でRISC-V CPU上で動作するソフトウェアをビルドする環境の構築が完了しました．

● RISC-VコアPicoRV32の取得

PicoRV32はGitHubリポジトリ上で公開されていま