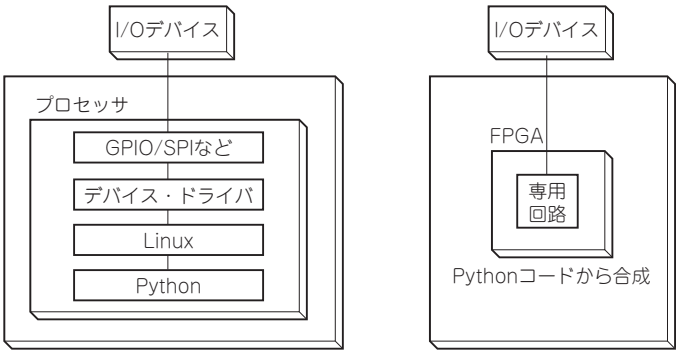
PythonでFPGAを作る

鈴木 量三朗



(a) Pythonシステムの場合　(b) 小規模FPGAの場合

図1　簡単なI/O制御はLinuxのような大げさなシステムではなく専用回路化できた方がよい

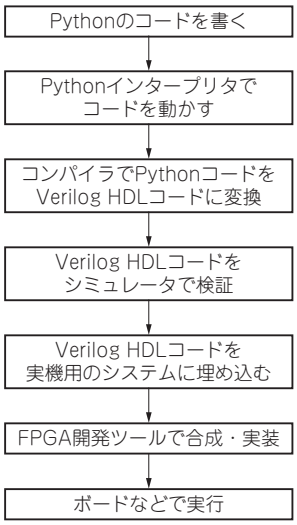


図2　Python記述をFPGA回路として動かす手順
Polyphonyの場合

前章まで使用してきたPython専用コンピュータでは，GPIOやSPIをPythonのライブラリ経由で扱いました．このライブラリは，ARMプロセッサで動作するLinuxからデバイス・ドライバ経由でSoCの内部バスをアクセスし，バスで接続されたFPGA部のCPU機能とデータのやり取りをしながら，GPIOやSPIをアクセスするというものでした［**図1**(a)］．LEDの点滅のような，本来簡単なI/Oデバイスの制御であっても，複雑な仕組みで実現しています．また，このような制御にCortex-Aクラスのプロセッサ やLinuxといったOSは必要ありません．

もっと簡単にI/O制御を実現する方法として，Pythonのコードからハードウェア(回路)を合成する技術があります．合成されたハードウェアは，小規模FPGAでも動作させることができます［**図1**(b)］．

Pythonコードからハードウェアを合成(高位合成)するイメージを**図2**に示します．

まずは体験

高位合成では，HDL(ハードウェア記述言語)でコードを書かずとも，ソフトウェアが得意なシーケンス処理をコンパイルしてハードウェア・モジュールを作成しFPGAの中に組み込むことができます．

● Pythonからハードウェアを合成するツールを使う

ここでは，Pythonのコードを Verilog HDLにコンパイルしてくれるオープンソースのツール(高位合成)「Polyphony(ポリフォニー)」を使い，Pythonコードのハードウェア化を試してみることにしましょう．

`https://github.com/ktok07b6/polyphony`

比較のため前章までと同じArty Z7をターゲットにします．

● サンプルは定番のLED点滅回路

LED点滅回路をPythonで記述したのが**リスト1**です．生成されるハードウェア・モジュールを**図3**に示します．クロックとリセットを入力とし，LEDを点滅させるための出力ポートを持つ回路が出来上がります．

クロックとリセットは，コンパイラによって自動的